Министерство образования и науки Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Институт естественных и точных наук
Факультет математики, механики и компьютерных технологий
Кафедра прикладной математики и программирования
Направление подготовки Программная инженерия

РАБОТА ПРОВЕРЕНА
Рецензент,

_____
«____»_____2018 г.

ДОПУСТИТЬ К ЗАЩИТЕ
Заведующий кафедрой, д.ф.-м.н.,
доцент
_____А.А.Замышляева
«____»_____2018 г.

Разработка сайта для детского сада

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ
ЮУрГУ–09.03.04.2018.299.ПЗ ВКР

Руководитель работы, доцент, к. т. н.
_____/М.Ю. Катаргин
«____»_____2018 г.
Автор работы
Студент группы ЕТ-414
_____/ А.Н. Курочкин
«____»_____2018 г.
Нормоконтролер, доцент
_____/Т.Ю. Оленчикова
«____»_____2018 г.

Челябинск 2018

АННОТАЦИЯ

Курочкин А. Н. Разработка сайта для детского сада. – Челябинск: ЮУрГУ, ЕТ-414, 70 с., 53 ил., 15 табл., библиогр. список – 16 наим., 2 прил.

Цель данной работы – разработать веб-сайт для дошкольного образовательного учреждения. В рамках дипломной работы были разработаны основные алгоритмы решения, спроектированы архитектура и интерфейс. На основании чего была выполнена программная реализация системы, произведена проверка работоспособности.

Первый раздел работы посвящен описанию предметной области, анализу требований к программному обеспечению, обзору и сравнительному анализу похожих решений. Также здесь приводится выбор средств для разработки веб-сайта и его обоснование.

Во втором и третьем разделах приводится описание архитектуры и пользовательского интерфейса веб-сайта.

Четвертый раздел посвящен разработке программного кода веб-сайта: здесь описываются основные принципы создания программного кода, которые были использованы при создании сайта.

Пятый раздел посвящен описанию разработанного сайта.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Актуальность темы. На сегодняшний день почти все сферы деятельности человека неразрывно связаны с интернетом, практически у каждого есть дома подключение к нему. Использование интернета в образовательных целях является очевидным способом повышения качества образовательного процесса.

Веб-сайт, расположенный в сети интернет, является важнейшим элементом информационной политики современного образовательного учреждения. Посредством него предоставляется актуальная информация об учреждении и налаживается контакт между сотрудниками и родителями, дети которых посещают данное учреждение, что в целом плодотворно сказывается на образовательном процессе.

Целью данной работы является разработка веб-сайта для дошкольного образовательного учреждения. Для достижения поставленной цели необходимо решить следующие *задачи*:

- выполнить анализ требований к программному обеспечению;
- провести обзор существующих решений, осуществить их сравнительный анализ;
- выбрать платформу, средства и инструменты для создания веб-сайта;
- спроектировать архитектуру и пользовательский интерфейс;
- разработать алгоритмы работы веб-сайта;
- разработать программную реализацию;
- проверить работоспособность.

1 АНАЛИЗ ТРЕБОВАНИЙ. ОБЗОР СУЩЕСТВУЮЩИХ РЕШЕНИЙ. ВЫБОР СРЕДСТВ ДЛЯ СОЗДАНИЯ САЙТА

1.1 Постановка задачи

Целью данной работы является разработка web-сайта для дошкольного образовательного учреждения, который будет размешен в сети интернет и будет использоваться родителями и сотрудниками учреждения.

Главной задачей данного сайта будет предоставление информации об учереждении и налаживание контактов между родителями и сотрудниками детского садика, что плодотворно скажется на образовательном процессе и работе учреждения в целом.

Особенности разрабатываемого сайта:

- у каждой группы в учреждении присутствует отдельная страница на сайте;
- страница группы может быть закрытой, доступ к ней может предоставляется по желанию администраторов;
- в каждой группе присутствует форум для общения родителей и сотрудников;
- наличие блога, где предоставляется информация о событиях в жизни учреждения;
- возможность создания информационных страниц, в которых размещена информация об учреждении;
- возможность регистрации и авторизации пользователей;
- разбитие пользователей на группы по уровню доступа;
- двухфакторная аутентификация через телефон или почту по желанию пользователя.

1.2 Описание предметной области

1.2.1 Сервер

Сервер (от англ. server) – компьютер или программа, который предоставляет определенные ресурсы или услуги другим компьютерам или программам, называемые клиентами (от англ. client) по их запросу [1]. Клиенты и серверы подключены к определенной общей сети, это может быть локальная сеть или глобальная сеть интернет.

Сервер может выполнять разные функции, например:

- хранение файлов;
- размещение сайтов в сети интернет;
- размещение базы данных;
- сложные научные вычисления;

- предоставление доступа к другим серверам;

В общем случае клиент отправляет определённый запрос серверу, после чего сервер принимает этот запрос, интерпретирует, выполняет поставленную задачу и отправляет ответ назад клиенту.

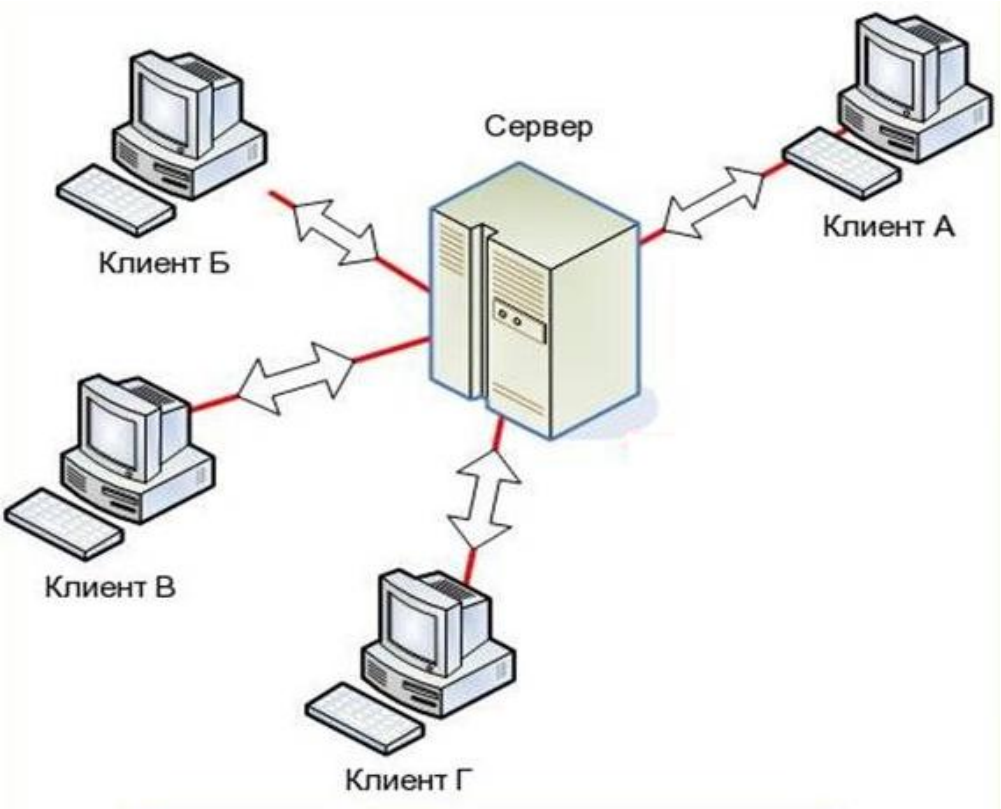


Рисунок 1.1 – Пример клиент-серверного взаимодействия

1.2.2 Web-сервер

Web-сервер – сервер, принимающий HTTP-запросы на получение ресурсов от клиентов, обозначенные URI-адресами, и выдающий им HTTP-ответы, где ответ может представлять из себя:

- HTML страница;
- изображение;
- медиа поток;
- файл.

Клиентом web-сервера обычно является веб-браузер – программа для просмотра веб-страниц.

URI (англ. Uniform Resource Identifier) – символьная строка, которая позволяет однозначно идентифицировать какой либо ресурс в сети интернет, будь то файл или HTML страница.

HTTP (англ. HyperText Transfer Protocol) – широко распространённый протокол передачи произвольных данных, который изначально предназначался для передачи документов в формате HTML.

HTML (англ. HyperText Markup Language) – стандартизированный язык разметки документов в глобальной сети интернет, который интерпретируется веб-браузерами. Полученный в результате интерпретации документ отображается на экране монитора компьютера.

1.2.3 Web-сайт

Web-сайт – набор логически взаимосвязанных веб-страниц, обычно представляющих собой HTML страницы, расположенные на web-сервере.

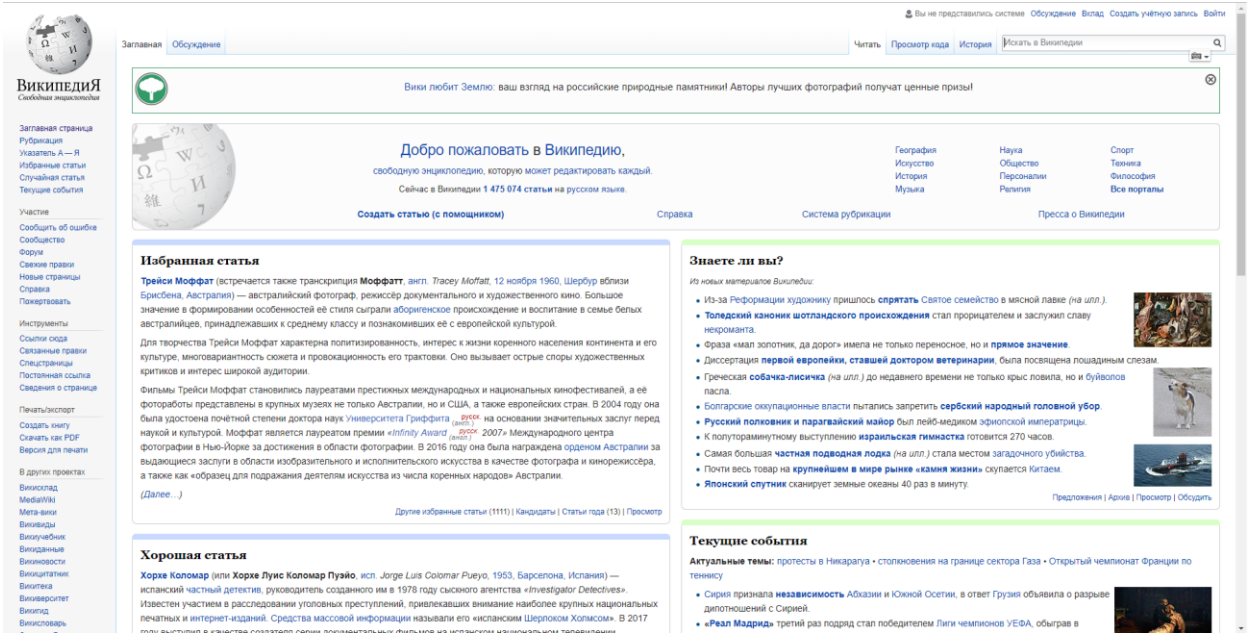


Рисунок 1.2 – Пример веб-сайта (Wikipedia.org)

Все web-сайты можно разделить на две группы:

- статические web-сайты – содержат в себе информацию, которую может изменить только владелец сайта, через изменение HTML разметки страниц. Сервер отправляет клиенту уже заранее готовые страницы. Как правило, это простые сайты, цель которых – предоставить информацию пользователям. Типичный пример – сайт визитка с контактными данными;
- динамические web-сайты – это сайт, который состоит из изменяемых страниц. Пользователи могут изменять информацию на сайте через интерфейс самого сайта. Страницы такого сайта, как правило, генерируются на при запросе их клиентом, а информация берется из баз данных.

1.2.4 База данных

База данных (БД) в самом общем смысле представляет собой организованный набор данных. Более конкретно, база данных представляет собой электронную систему, которая позволяет легко получать доступ к

данным, манипулировать ими и их обновлять. Другими словами, база данных используется организацией как способ хранения, управления и получения информации. Современные базы данных управляются с использованием системы управления базами данных (СУБД).

Основой любой базы данных является модель представления данных, которая описывает структуры данных и способ их организации, или другими словами это представление объектов и способ их взаимосвязи, которые обычно имеют следующую классификацию:

- один к одному, где каждому объекту соответствует какой-либо другой объект. Например, у каждого гражданина есть паспорт, причем каждый паспорт уникальный и относится только к одному человеку;
- один ко многим, где одному объекту соответствует множество других объектов. Самая распространённая связь. Типичный пример – родители-дети, когда у одного родителя могут быть несколько детей;
- многие ко многим, связь, в которой множество объектов связано с другим множеством объектов. Пример – сотрудник в организации занимает определенную должность. С другой стороны, у одной должности может быть много сотрудников.

1.2.5 Реляционная база данных

Реляционная база данных – это модель базы данных, которая хранит данные в таблицах [3]. Подавляющее большинство баз данных, используемых в современных приложениях, являются реляционными, поэтому термины «база данных» и «реляционная база данных» часто используются синонимом. Аналогичным образом, большинство систем управления базами данных (СУБД) являются системами управления реляционными базами данных. Другие модели баз данных включают файловые и иерархические базы данных, но они редко используются.

Термины реляционной модели представлены в таблице 1.1

Таблица 1.1

Термины реляционной модели

Термин	Определение
Отношение	Таблица с фиксированным числом столбцов и неограниченным числом строк
Кортеж	Строка в таблице
Атрибут	Столбец в таблице
Домен	Множество всех допустимых значений атрибута
Первичный ключ, ключевой атрибут	Уникальный идентификатор – множество столбцов, в общем случае один столбец, которые определяют уникальность строки

Продолжение таблицы 1.1

Кардинальность	Количество строк
Степень	Количество столбцов
Функциональная зависимость	Бинарное отношение между множествами атрибутов данного отношения

Управление реализационной базой данных осуществляется с помощью языка SQL (от англ. Structured Query Language, рус: язык структурированных запросов), использующийся для добавления, удаления и изменения данных в базе.

Рисунок 1.3 – Схема взаимодействия с базой данных

1.2.5 Нормальные формы реляционных баз данных

Нормальные формы – набор принципов для проектирования таблиц и структур данных в реляционной базе данных [4]. Нормальные формы предотвращают избыточность данных, повышают эффективность базы данных и уменьшают ошибки согласованности. База данных, как говорят, находится в первой нормальной форме (1НФ), второй нормальной форме (2НФ), третьей нормальной форме (3НФ) и т. д. На практике обычно используется 3НФ, более высокие уровни используются редко.

Описание нормальных форм представлено в таблице 1.2.

Таблица 1.2

Нормальные формы реляционных баз данных

Нормальная форма	Определение
Первая нормальная форма (1НФ)	Отношение находится в 1НФ тогда, когда множество всех используемых значений в доменах являются атомарными
Вторая нормальная форма (2НФ)	Отношения находится во второй нормальной форме тогда и только тогда, когда она находится в первой нормальной форме и каждый не ключевой атрибут неприводимо (функционально полно) зависит от её потенциального ключа

Продолжение таблицы 1.2

Третья нормальная форма (3НФ)	Отношения находится в третьей нормальной форме тогда и только тогда, когда она находится во второй нормальной форме, и отсутствуют транзитивные функциональные зависимости не ключевых атрибутов от ключевых
Нормальная форма Бойса-Кодда (НФБК)	Отношения находится в нормальной форме Бойса — Кодда (иначе — в усиленной третьей нормальной форме) тогда и только тогда, когда каждая её нетривиальная и неприводимая слева функциональная зависимость имеет в качестве своего детерминанта некоторый потенциальный ключ
Четвертая нормальная форма (4НФ)	Отношения находится в четвёртой нормальной форме, если она находится в НФБК и все нетривиальные многозначные зависимости фактически являются функциональными зависимостями от её потенциальных ключей

Для корректности данных, предоставляемых базой данных, следует как можно сильнее избегать любой избыточности хранимой информации. В противном случае наполнение и извлечение данных может сопровождаться логическими ошибками.

Использование нормализации позволяет спроектировать корректную базу данных, при заполнении которой легче избежать дублирования информации, что может привести к несогласованности между копиями одной и той же информации.

1.2.6 NoSQL база данных

NoSQL – это подход к дизайну базы данных, который может сочетать в себе множество моделей данных, включая ключевые значения, документы, столбцы и графические форматы. NoSQL, который обозначает «не только SQL», является альтернативой традиционным реляционным базам данных, в которых данные размещаются в таблицах, а схема данных тщательно разработана до создания базы данных. Базы данных NoSQL особенно полезны для работы с большими наборами распределенных данных.

Термин NoSQL можно применять к некоторым базам данных, которые были предшественниками реляционных баз данных, но чаще это относится к не реляционным базам данных, созданные в последнее десятилетия. В приложениях, которые используют этот тип баз данных, требования к производительности и масштабируемости перевешивают необходимость непосредственной и жесткой согласованности данных, которые РСУБД предоставляют транзакционным корпоративным приложениям.

1.3 Анализ требований к программе

1.3.1 Общие требования к программе

Поставлена цель разработки web-сайта для дошкольного образовательного учреждения. Целями создания этого сайта являются:

- организация взаимодействия сотрудников дошкольного учреждения и родителей, для лучшего проведения образовательного процесса;
- информирование родителей о прошедших и предстоящих мероприятиях в детском садике;
- получение родителями актуальной и достоверной информации об образовательном учреждении.

1.3.2 Функциональные требования

Разрабатываемый сайт должен обладать следующими функциями:

- регистрация пользователей;
- авторизация пользователей;
- разделение пользователей по ролям;
- разделение сайта на группы;
- возможность устанавливать список пользователей, которые имеют доступ к группам;
- возможность создавать, изменять и редактировать группы;
- полноценный форум в каждой группе;
- разделения форумов на смысловые секции;
- возможность создавать, изменять и удалять статические информационные страницы;
- блог учреждения;
- возможность отправки пользователями личные сообщения;
- панель администратора.

1.3.3 Требования к функционированию сайта

Требуется, чтобы сайт был доступен из web-браузера в глобальной сети интернет. Основная бизнес логика должна содержаться на сервере. Клиент же обрабатывает дополнительную вспомогательную логику, связанную с графическим интерфейсом.

1.3.4 Требования к внешнему оформлению

Взаимодействие пользователя с сайтом должно происходить посредством графического интерфейса, который предоставляет доступ ко всем функциям сайта. Он не должен быть переусложнен лишними графическими элементами и избыточной информацией. Текст на сайте должен быть легко читаемым и быть различимым на заднем фоне сайта.

Должна присутствовать удобная навигация по сайту, которая не будет вводить в заблуждение пользователя. На каждой странице должна быть информация о текущем положении пользователя на сайте, для удобной его навигации между страницами сайта.

Дизайн должен адаптироваться под изменение размеров окна браузера. Также Сайт должен отображаться одинаково на всех основных современных браузерах на сколько это возможно.

1.3.5 Требования к информационной безопасности

Сайт должен быть защищён от несанкционированного доступа посредством авторизации через логин и пароль пользователя. Доступ к отдельным участкам сайта, для каждого пользователя, регламентируется администрацией сайта.

1.3.6 Обработка отказов и аварийных ситуаций

Действия пользователя не должны вызывать отказ какой-либо части сайта. Должна присутствовать обработка нештатных ситуаций, которые могут быть вызваны действиями пользователя или администратора. В случаи возникновения ошибки, система должна вернутся в состояние до ошибки в автоматическом режиме, а информация об ошибки должна быть занесена в журнал, для последующего выяснения причин.1.4.3 Условия эксплуатации

1.3.7 Требования к квалификации персонала

Предполагается, что пользователь имеет опыт работы с любым современным веб-браузером: умеет заходить на сайт, вводить данные в формы и т.д.

1.3.8 Требования к составу и параметрам технических средств

Для функционирования сайта требуется:

- операционная система семейства Windows или UNIX/LINUX;
- свободное дисковое пространство не менее 200мб;

1.3.9 Перспективы модернизации и развития

При разработке сайта следует учесть возможность последующей модернизации и улучшения функционала сайта, например, возможность функционирования с другими базами данных (MySQL, PostgreSQL).

1.4 Анализ существующих средств реализации

Проведем анализ существующих решений, которые обладают схожим функционалом разрабатываемого сайта. Рассмотрим самые популярные решения.

1.4.1 Система управления контентом WordPress

WordPress – CMS (англ. Content Management System, рус: Система управления содержимым) с открытым исходным кодом, выпущенная под лицензией GNU GPL 2+ [2]. Написана на php и JavaScript, использует базу данных MySQL. Одна из самых распространённых систем, для создания сайтов, начиная от простых блогов и заканчивая сложными новостными ресурсами.

Можно выделить основные достоинства данной CMS:

- наличие встроенных функций, таких как блог, статичные страницы, комментарии и т. д;
- большой выбор дополнительных плагинов – дополнительных модулей, которые расширяют функциональность системы;
- бесплатная модель распространения как самой CMS, так и базы данных MySQL;
- постоянные обновления, которые добавляют новые функции и устраняют ошибки;
- возможность кастомизации дизайна сайта с помощью шаблонов;
- простая и не дорогая аренда web-сервера - большинство сервисов, которые предоставляют хостинг, поддерживают php;

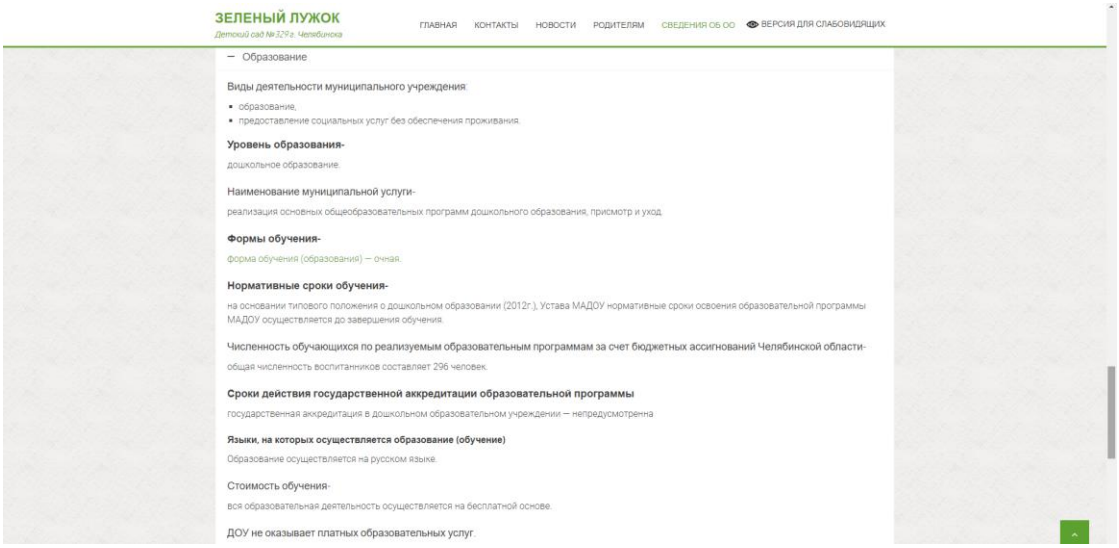


Рисунок 1.4 – Пример WordPress сайта

Во многих случаях выбор этой CMS является самым оптимальным решением с точки зрения затрат времени на создания готового сайта, что подтверждается моим опытом работы с данной системой. Но с другой стороны, она так же обладает некоторыми существенными недостатками:

- наличие конфликтов между плагинами. Многие плагины не могут работать одновременно в одной системе, что приводит к ошибкам и уменьшению скорости работы сайта в целом;
- хотя сама CMS бесплатная, многие качественные плагины для нее стоят не малых денег. Бесплатные же плагины, создаваемые разработчиками-энтузиастами, не всегда бывают качественными и могут содержать серьезные ошибки, в том числе и ошибки безопасности;
- ещё хуже дела обстоят с шаблонами для сайта, в подавляющем большинстве они платные. Бесплатные же шаблоны обычно не качественные и обладают дефектами, такими, например, как плохая масштабируемость в зависимости от разрешения экрана;
- Как показывает индекс TIOBE, оценивающий популярность языков программирования, на основе подсчёта результатов поисковых запросов (см. рисунок 1.5), язык, на котором написана CMS – php, постепенно теряет популярность. И с каждым годом эта тенденция только усиливается. Многие компании отказываются от php в пользу других, более современных языков.

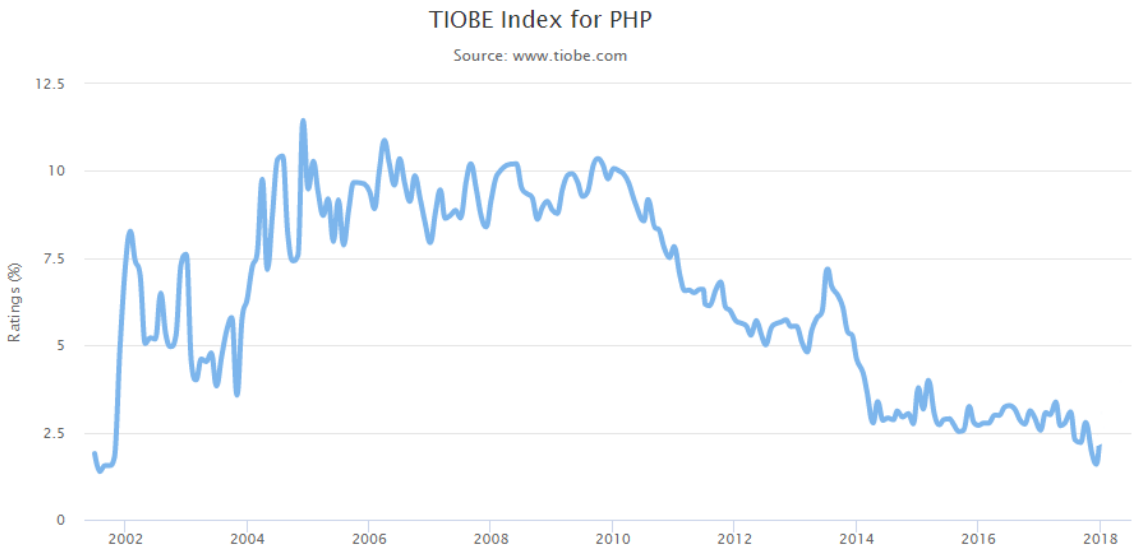


Рисунок 1.5 – Индекс популярности php по версии TIOBE

Хотя данная CMS обладает в некоторой степени подходящим нам функционалом, она не обладает всеми необходимыми функциями, которые должны быть в разрабатываемом сайте. Дополнительно разрабатывать модули на языке php считаю бесперспективным.

1.4.2 Конструктор сайтов uCoz

uCoz – бесплатный конструктор и хостинг сайтов. Позволяет разрабатывать сайты даже тем пользователям, которые не знаю не одного языка программирования и не разу до этого не создавали сайтов. Одна из самых распространенных систем создания сайтов в русскоязычной среде интернета.

Основные достоинства данной системы:

- богатый функционал, возможность создания блога, форума, информационных страниц и т.д.;
- наличие большого количества бесплатных готовых шаблонов, которые позволяют изменять дизайн сайта;
- хостинг, за который не надо платить.

Так же можно выделить и недостатки:

- по условиям пользовательского соглашения, веб-сайт созданный в данной системе принадлежит не создателю, а компании Юкоз медиа, следовательно, веб-сайт может быть изменен или удален по воле администраторов системы;
- наличие рекламы на веб-сайте, которую можно выключить, если только ежемесячно за это платить;
- весь функционал доступен за дополнительную плату.

Даже с учетом платных функций, предоставляемых данной системой, их все равно недостаточно для разрабатываемого сайта. Дополнительные

функции пишутся на языке php, но их использование так же стоит дополнительных денег. С учетом выше перечисленного, тот же WordPress выглядит куда более подходящим решением.

Рисунок 1.6 – Пример сайта, созданного в системе uCoz

1.4.3 Сайт Vk.com

Vk.com – самая популярная социальная сеть на территории России и стран СНГ. У подавляющего большинства жителей Российской федерации есть аккаунт в этой социальной сети. Позволяет создавать так называемые группы – аналоги сайтов с ограниченным функционалом.

Основные достоинства групп Vk.com:

- бесплатное создание групп, за них не надо платить;
- группы могут включать в себя: блог, некоторое подобие форума, хранилище документов, фотоальбомы и т.д.;
- удобное администрирование группами, можно сделать закрытую группу, в которую будет доступ только у приглашенных пользователей;
- записи в блог группы отображаются в новостной ленте у всех вступивших в группу пользователей.

Существуют и недостатки:

- наличие постоянной рекламы, которую нельзя выключить даже за деньги;
- привязка к аккаунту – в случаи потери или блокировки аккаунта вы можете решиться доступа к администрированию группы;
- необходимость создания и управления группами для каждой группы в учреждении, что достаточно неудобно. Например, если необходимо

заблокировать пользователя, придется блокировать его в каждой группе, которых может насчитываться десятки;

- некоторые сотрудники могут не захотеть использовать свои аккаунты в социальной сети для работы из-за соображений анонимности.

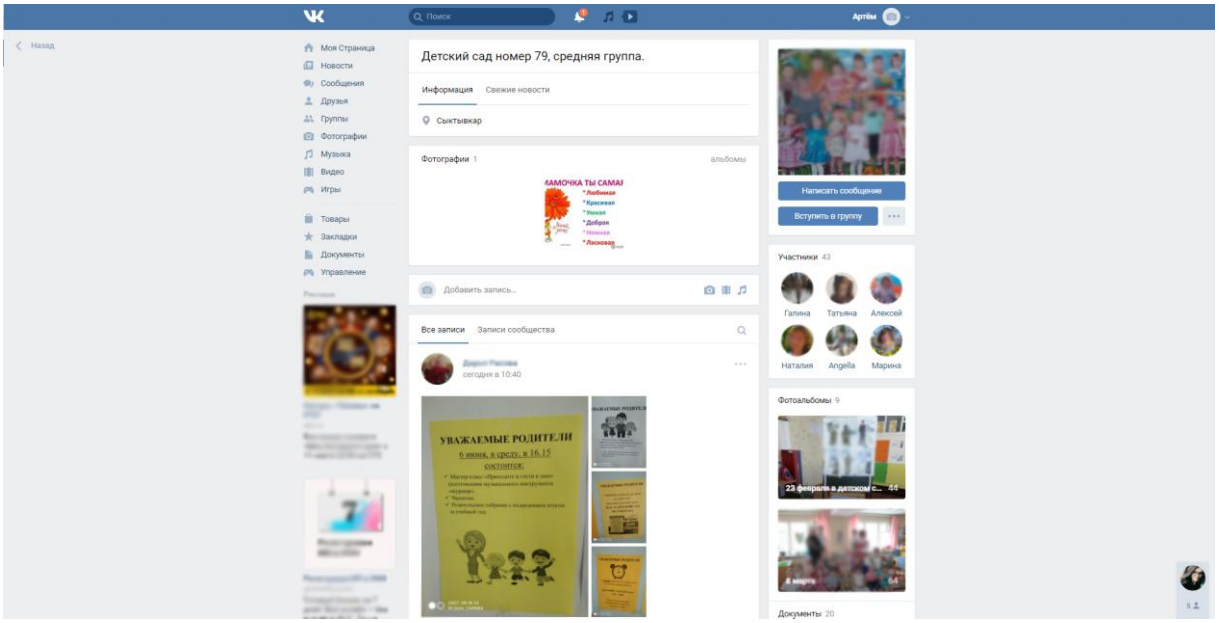


Рисунок 1.7 – Пример группы в социальной сети Vk.com

Vk.com не подходит для решения нашей задачи, из-за тех недостатков, которые были перечислены выше. Хотя, как дополнительный ресурс к основному сайту, группа в социальной сети выглядит идеально. Это обычная практика у многих компаний – иметь собственный сайт и группу в социальной сети.

1.5 Обоснование выбора платформы, средств и инструментов для создания ПО

При выборе средства разработки, следует отталкиваться от тех возможностей, которые предоставляет то или иное средство разработки, наличия качественной русскоязычной документации, стоимости реализации, актуальности и перспективности выбранного средства разработки.

1.5.1 Фреймворк ASP .NET Core

Было рассмотрено несколько вариантов для разработки серверной части, в том числе: Spring MVC вместе с языком программирования Java, ASP .Net и ASP .Net Core с языком программирования C#, Ruby on Rails использующий язык Ruby, php и т. д.

Для разработки был выбран ASP .NET Core framework, который является следующим этапом в эволюции классического ASP .NET. Данный фреймворк

хорошо подходит по всем критериям. C# - язык, который использует данный фреймворк – один из самых современных и постоянно обновляемых. Он содержит один из самых больших наборов библиотек, которые достаточно сильно ускоряют разработку. На этом языке разрабатывается большое количество программного обеспечения, начиная от приложений для смартфонов и компьютеров, заканчивая сложными сервисами для банковского сектора. Разработчики этого языка вдохновлялись языком Java, от которого C#, в плане возможностей, давно ушел вперед. Так же у этого языка есть большое сообщество разработчиков, которые могут помочь в случае возникновения проблем, хорошая русскоязычная документация и учебные материалы, что является большой проблемой для многих выше перечисленных языков программирования.

Ещё одним весомым достоинством C# является среда разработки – Visual Studio от компании Microsoft, которая является одной из самых удобных и продуманных сред разработки. Является бесплатной в версии Community.

ASP .Net Core – достаточно новый фреймворк, который в некоторых случаях, в плане возможностей, немного уступает классическому ASP .NET, но имеет одно существенное достоинство – он разработан на платформе .Net Core, который, в отличии от .Net Framework, является кроссплатформенным. Из этого следует отсутствие зависимости как от операционной системы, так и от IIS (англ. Internet Information Server) – веб-сервер от компании Microsoft, который работает только на операционных системах семейства Windows. Сайты, разработанные с помощью ASP .Net Core можно запускать не только под операционной системой Windows, но и в других, например, в операционных системах семейства GNU/LINUX, которые более распространены как операционные системы для серверов.

1.5.2 СУБД MSSQL

Для нашей работы вполне подходят реляционные системы управления базами данных. При разработке можно было использовать NoSQL базы данных, но потенциал таких баз данных лучше всего раскрывается при огромных объемах слабоструктурированных данных. Данные же в базе данных, которую мы будем использовать будут структурированы, при этом структура не будет подвержена каким-либо изменениям.

Использование реляционных СУБД – классический подход в создании web-сайтов.

Так как мы будем использовать ORM, выбор реляционной базы данных не играет большой роли. Мы можем использовать: SQLite, MySQL, PostgreSQL или MSSQL Server.

Для разработки был выбран MSSQL Server от компании Microsoft. Даная СУБД хорошо себя зарекомендовала в предыдущих проектах, так же она

является бесплатной в версии express. В дальнейшем, при желании мы можем без труда поменять базу данных.

1.5.3 Фреймворк Entity Framework Core

ORM (от англ. Object-Relational Mapping) – технология программирования, которая связывает базы данных с концепциями объектно-ориентированного программирования [5].

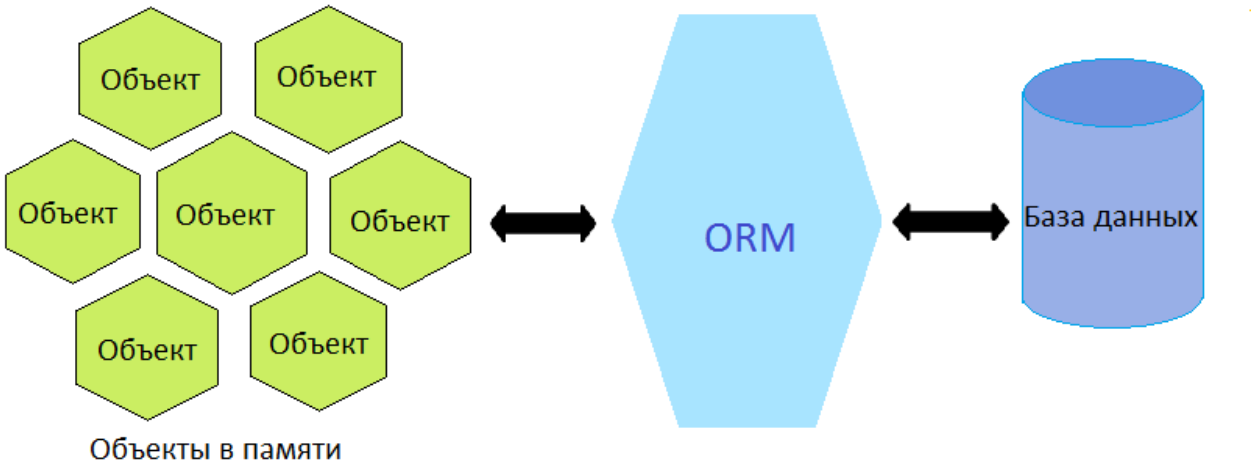


Рисунок 1.8 – Принцип работы ORM

ORM позволяет взаимодействовать с данными в базе данных как с классами, без необходимости взаимодействия напрямую с базой данных. Это позволяет разработать программу, у которой не будет зависимости от конкретной реализации базы данных, отсутствует необходимость в использовании языка SQL, необходимо лишь описать нужные классы и привязать их к ORM. В конечном счете это ускоряет разработку и облегчает поддержку программы в будущем.

Из недостатков можно выделить то, что доступ к данным через ORM зачастую получается более медленным, чем при использовании языка SQL. Чтобы увеличить скорость работы, можно использовать язык SQL в особо тяжелых задачах, которые отнимают много времени, а для всех остальных использовать ORM.

Использование ORM в связке с ASP .Net или ASP .Net Core является де факто стандартом при разработке сайтов и веб-приложений.

На данный момент для ASP .Net Core существует две ORM системы:

- Entity Framework Core;
- NHibernate.

Для разработки была выбрана ORM Entity Framework Core, так как он поддерживает больше возможностей, чем конкурент, а в частности поддерживает Code First – поход, при котором программисту необходимо

лишь описать нужные классы, после чего фреймворк сам сгенерирует базу данных. Данный подход очень удобен тем, что отсутствует необходимость описания привязки классов к базе данных, что ускоряет скорость разработки и позволяет избежать ошибок, при изменении класса, в обратном же случае помимо изменения класса, программисту необходимо изменить саму базу данных и описание привязки классов к этой базе данных. Так же для Entity Framework существует большое количество документации и учебных материалов, в отличии от NHibernate.

1.6 Выводы по разделу

В данном разделе были проанализированы требования к разрабатываемому сайту, в том числе функциональные требования, требования к безопасности и условия эксплуатации. Так же были описаны перспективы последующей модернизации сайта.

Был проведен обзор существующих решений для создания, отвечающего нашим требованиям сайта.

Было произведено обоснование выбора средств и инструментов для разработки сайта, а именно:

- для разработки был выбран ASP .Net Core framework, язык программирования C# и ORM Entity Framework Core;
- для хранения данных будет использоваться MSSQL Server в версии Express.

2 РАЗРАБОТКА АРХИТЕКТУРЫ

2.1 Схема Model-View-Controller

MVC (от англ. Model-View-Controller) – схема разделения данных, интерфейса и бизнес логики приложения, таким образом, что каждая часть может изменяться независимо.

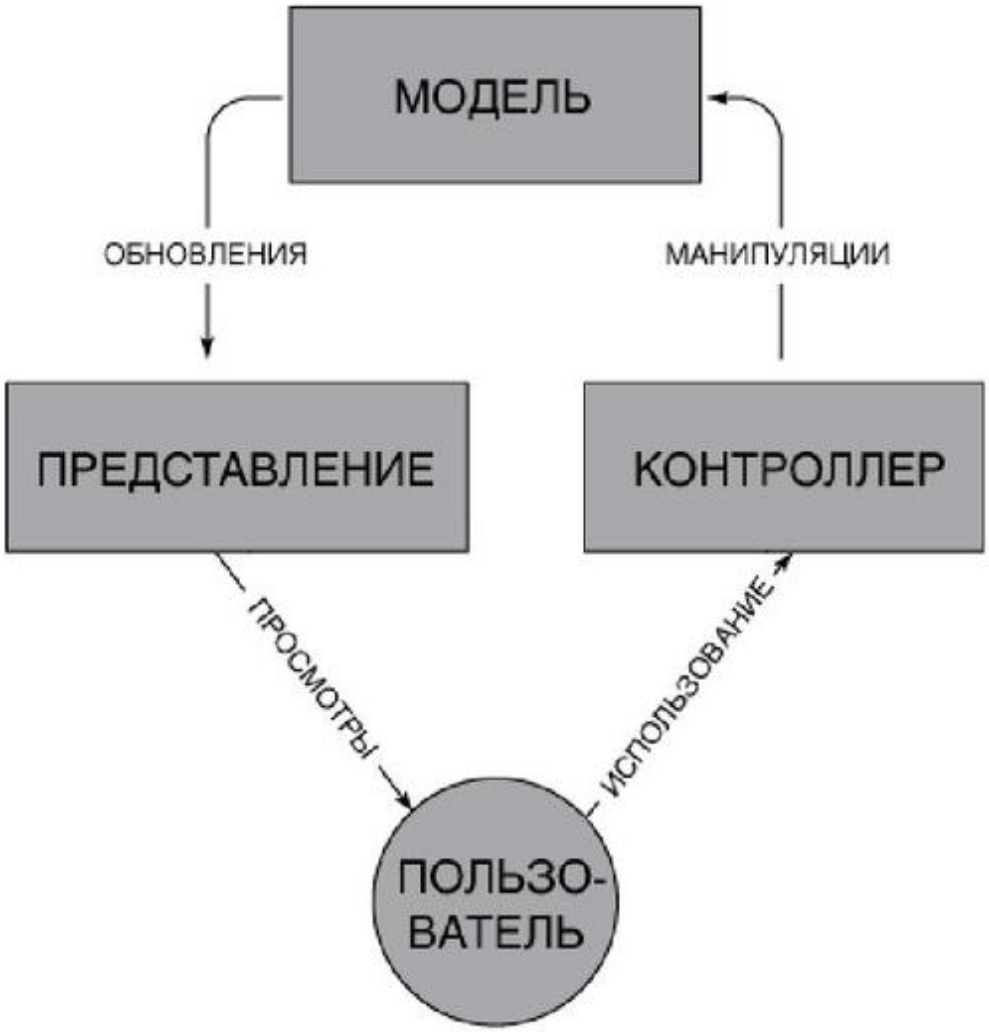


Рисунок 2.1 – Схема MVC

Данная схема Разделена на 3 части (рисунок 2.1):

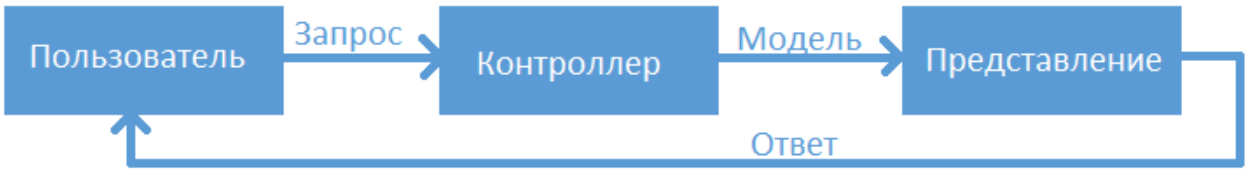
- модель (от англ. Model) – данные программы. Реагирует на команды контролера, изменяя свое состояние;
- представление (от англ. View) – интерфейс программы. Отображает модель пользователю;
- контроллер (от англ. Controller) – реагирует на действия пользователя, сообщая модели о необходимости изменений.

Основная цель данной схемы состоит в том, чтобы отделить бизнес-логику (Controller) от интерфейса (View) [6]. Например, не изменяя интерфейс, можно изменить реакцию на действие пользователя, для этого

достаточно изменить контролер. Так же для одних и тех-же данных можно создать отдельные интерфейсы, например, данные можно представить, как в таблице, так и гистограмме.

Одно из ключевых достоинств данного шаблона является увеличение возможности повторного использования частей кода программы.

Подход с использованием данной схемы является основным при разработке веб-сайтов с использованием технологии ASP .Net и ASP .Net Core. На рисунке 2.2 представлена схема реализации MVC в ASP .Net Core.



Рисунков 2.2 – Схема MVC в ASP .Net Core

2.2 Архитектурная модель

Архитектурная модель – принципы проектирования и эволюции системы, способ ее организации всех компонентов и правил их взаимодействия как со средой, так и между друг другом. Выбор архитектурной модели – одна из основ всей разработки не только сайтов, но и многих других клиент-серверных приложений. Правильный выбор архитектуры позволит в будущем избежать многих глобальных проблем, связанных с разработкой, дальнейшей поддержкой программного обеспечения и повторным использованием кода.

2.2.1 Монолитная архитектура

Самая простая архитектура. Реализация этой архитектуры представляет собой цельное приложения, без разбития на какие-либо модули. Довольно распространенная архитектура, хорошо подходящая для простых приложений.

С точки зрения схемы MVC, это классический подход, в котором вся бизнес логика приложения содержится в контролерах. В маленьких проектах это приемлемо, так как само количество контролеров не высоко, так и логики в них содержится не много. Но при увеличении проекта, возрастает связанность отдельных частей системы, что сильно ограничивает разработчика в последующей разработки приложения. Добавление и редактирование отдельных частей системы становится тяжелой задачей, так как изменения в одной части, может вызвать изменение поведения в другой части программы, что может спровоцировать появление новых ошибок.

Из достоинств данной архитектуры можно выделить намного более высокую скорость разработки приложений, по сравнению с другими архитектурами.

2.2.2 Многоуровневая архитектура

Многоуровневая архитектура, суть которой состоит в том, чтобы разбить программу на отдельные уровни, называемые слоями. Каждый слой может взаимодействовать только с соседними слоями. Компоненты такой архитектуры должны быть слабосвязанными. Неотъемлемой частью этой архитектуры является внедрение зависимостей.

Самой распространённой вариацией многоуровневой архитектуры является трехуровневая архитектура, которая разбита на три уровня:

- Presentation Layer (рус: уровень представления) – уровень взаимодействия пользователя с системой. Данный уровень включает в себя пользовательский интерфейс, механизмы получения и отправки информации пользователю.
- Business Layer (рус: уровень бизнес логики) – содержит всю бизнес логику приложения, разбитую на отдельные модули, называемые сервисами. Сервисы не зависят друг от друга и могут быть заменены или изменены без влияния на другие сервисы.
- Data Access Layer (рус: уровень доступа к данным) – здесь размещается вся логика связанная с данными. Управление данными происходит с помощью отдельных модулей. Через эти модули происходит создание, извлечение, изменение и удаление данных.

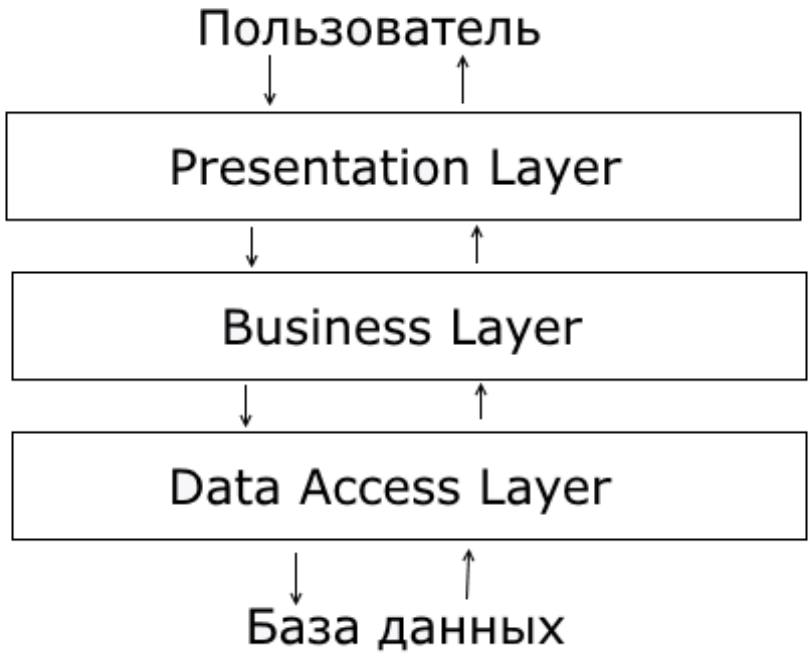


Рисунок 2.3 – Трёхуровневая архитектура

Скорость разработки приложений с использованием данной архитектуры значительно медленнее, по сравнению с монолитной архитектурой, так как приходится разбивать программу на части и организовывать взаимодействие слоев, но трудоемкость последующей разработки и поддержки значительно меньше, из-за независимости отдельных частей программы друг от друга.

При разработке больших приложений, первичный выигрыш времени из-за выбора монолитной архитектуры нивелируется затраченным временем при последующей разработке и поддержке приложения.

2.2.3 Луковичная архитектура

Термин луковичная архитектура был предложен Джеффри Палермо 2008 году. В данный момент это одна из самых распространенных архитектур для построения веб-приложений, уступая по популярности, возможно, только трехуровневой архитектуре.

Как и в трехуровневой архитектуре, в данной архитектуре происходит разбиение программы на уровни, с той разницей, что каждый последующий уровень может завесить только от уровней, которые располагается выше. Образно, это можно представить, как луковицу, где есть один независимый слой – сердцевина, а во круг него наслаиваются остальные зависимые уровни – слои.

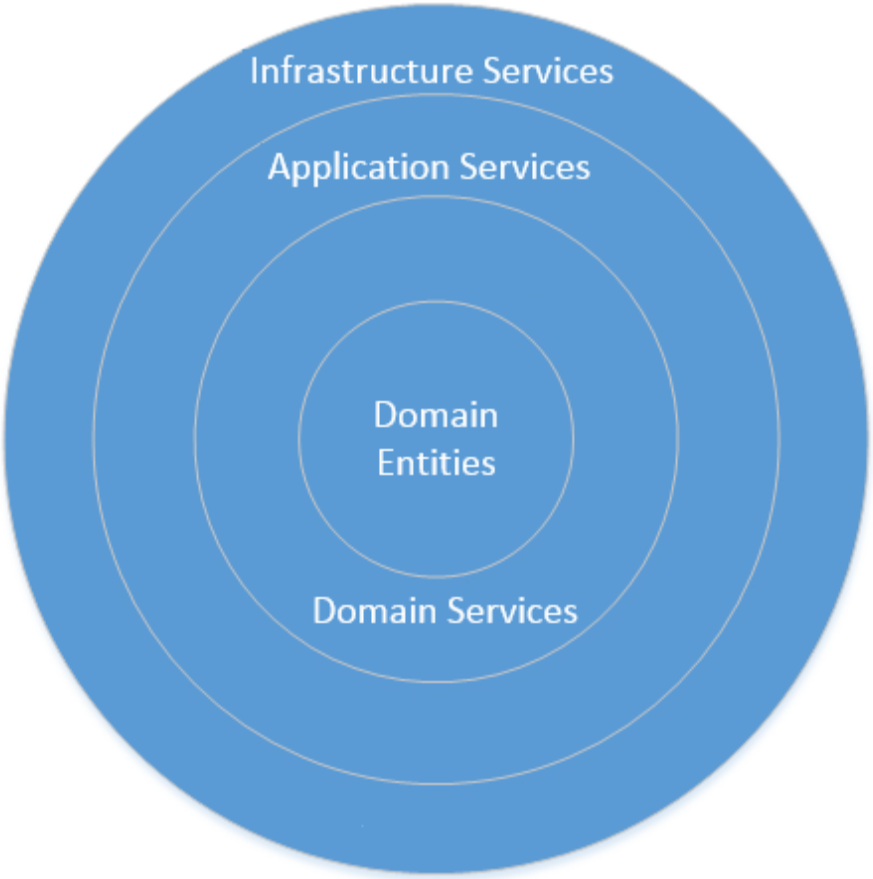


Рисунок 2.4 – Пример луковичной архитектуры

Приложение, разрабатываемое при помощи данной архитектуры может быть разделено на неограниченное количество слоев. Классическая же реализация разделена на 4 слоя:

- Domain Entities – единственный независимый слой. В нем хранятся модель домена – классы тех сущностей, которыми оперирует программа;
- Domain Service – в данном слое хранятся интерфейсы взаимодействия с моделью домена. Обычно это интерфейсы классов, которые отвечают за работу с источниками данных;
- Application Service – слой, в котором располагаются вспомогательные классы для работы с моделью домена. Как правило на данном уровне располагаются только интерфейсы этих классов;
- Application Service – на данном уровне хранятся классы, связанные с бизнес логикой. Почти всегда тут хранятся только интерфейсы, а реализация выносится на следующий уровень;
- Infrastructure Service – хранит в себе конкретные реализации интерфейсов с предыдущих слоев, в том числе вся бизнес логика и все классы для работы с источниками данных, пользовательский интерфейс, механизмы взаимодействия с пользователем и тесты.

2.2.4 Выбор архитектурной модели

В качестве архитектурной модели была выбрана трехуровневая архитектура, как компромисс между скоростью разработки и легкостью последующей поддержки и разработки. Луковичная же архитектура, которая подходит больше для больших проектов, является избыточной для разрабатываемого сайта.

2.3 Паттерн проектирования Unit of work

Unit of work – паттерн проектирования, который используется для инкапсулирования логики работы с источниками данных [7]. Состоит из отдельных частей, называемые репозиториями (от англ. Repository). Каждый репозиторий отвечает за работу с отдельной сущностью в базе данных. Схема работы представлена на рисунке 2.5.

В архитектуре разрабатываемого сайта Unit of Work будет выполнять роль уровня доступа к данным. В нем будет инкапсулирована логика работы с ORM, что обеспечит отсутствие зависимости и позволит нам в будущем при желании поменять саму ORM без переписывания кода остальных частей сайта. Репозитории будут обеспечивать доступ к отдельным сущностям сайта, таким как информация о пользователе, сообщения, оставленные пользователями и т.д.

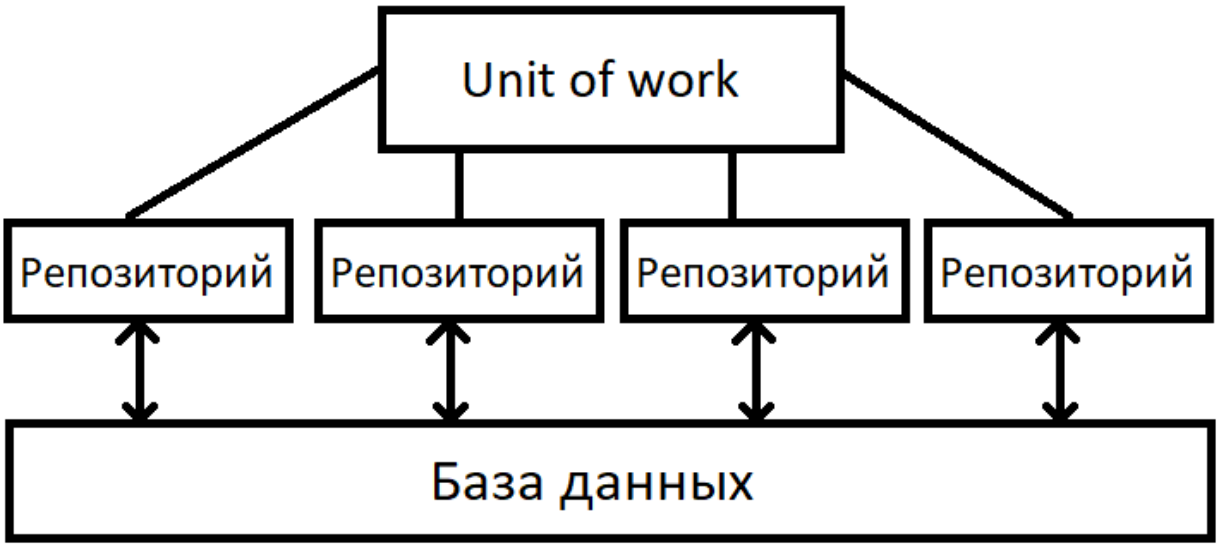


Рисунок 2.5 – Схема работы Unit of Work

2.4 Паттерн проектирования Data Transfer Object

DTO (от англ. Data Transfer Object) – один из паттернов проектирования, объект, который используется для передачи данных между подсистемами приложения. Не содержит в себе никакой логики, только данные. При использовании данного паттерна мы можем передавать в функции DTO, вместо множества параметров, что очень удобно, особенно при большом объеме данных. В разрабатываемом сайте будем использовать DTO для обмена информацией между уровнями архитектуры.

Важно подчеркнуть, что данные, которые мы будем извлекать из базы данных будут так же являться DTO, как и модели, которые мы будем передавать в представления при использовании Шаблона MVC.

2.5 Аутентификация и авторизация пользователя

Аутентификации – это проверка соответствия субъекта и того, за кого он пытается себя выдать, с помощью некой уникальной информации. Система аутентификации проверяет подлинность пользователей и является первым шагом к ограничению доступа к важной части приложения. Как правило, в простейшем случае аутентификации на сайтах происходит с помощью уникального имени пользователя, называемого логином (от англ. Login) и пароля.

Двухфакторная аутентификация – тип аутентификации пользователя, который использует данных двух разных типов, что обеспечивает более эффективную защиту от несанкционированного проникновения. На практике это обычно выглядит так: сначала пользователь использует логин и пароль, после это система проверят правильность предоставленных данных и

задействует второй уровень аутентификации. Обычно это отправка специального одноразового кода на телефон, с помощью смс, или на электронную почту. После этого, пользователь использует полученный код для окончательной аутентификации.

Большинство сайтов в настоявшее время переходят к двухфакторной аутентификации пользователей из-за соображений безопасности. Как правило, зачастую пользователи используют один и тот же логин и пароль для множества разных сайтов. И в случае взлома одного из них и получения логина и пароля пользователя, злоумышленник может получить доступ к конфиденциальной информации пользователя содержащейся на других веб-сайтах, которые не используют двухфакторную авторизацию.

Авторизация – предоставление прав или привилегий на выполнение определенных действий пользователю, связанным с информационной безопасностью и компьютерной безопасностью в целом, и для контроля доступа в частности. С точки зрения сайта, авторизация – предоставление доступа к отдельным функциям и частям сайта, например, к панели администратора.

2.5.1 Библиотека ASP.NET Core Identity

ASP.NET Core Identity – библиотека, входящая в состав ASP .NET Core, предоставляющая систему авторизации и аутентификации пользователей (в том числе и двухфакторную).

Использование Identity вместе с ASP .NET – довольно распространенная практика, при создании веб-сайтов и веб-сервисов. Однако у такого похода есть несколько проблем:

- плохая адаптируемость для трехуровневой архитектуры. Достаточно сложно грамотно разделить данную систему на несколько уровней;
- зависимость от Entity Framework. Стандартная реализация Identity очень сильно зависит от Entity Framework, чего бы хотелось избежать в разрабатываемой системе.

С учетом вышеперечисленных недостатков, было принято решение разработать свою систему аутентификации и авторизации.

2.5.2 Технология Cookie

HTTP cookie (также называемый web-cookie или просто cookie) – это небольшой фрагмент данных, отправленных с веб-сайта и хранящихся на компьютере пользователя. Cookie были разработаны как надежный механизм для веб-сайтов, чтобы запоминать информацию о состоянии, например, товары, добавленные в корзину покупок в интернет-магазине. Веб-клиент (обычно веб-браузер) всякий раз при попытке открыть страницу

соответствующего сайта пересылает этот фрагмент данных веб-серверу в составе HTTP-запроса.

С момента своего появления cookie вызывали обеспокоенность пользователей Интернета, поскольку слежение за действиями и предпочтениями пользователей может подвергнуть опасности тайну личной жизни. Как результат, в Европейском союзе, Соединённых Штатах, и в других странах были приняты соответствующие законы, регулирующие применение cookie. В любом случае, разрабатываемый сайт не будет следить за действиями пользователей. А cookie будут использоваться только для аутентификации.

Cookie будут использоваться в системе аутентификации и авторизации пользователя в разрабатываемом сайте. В случае успешной аутентификации, сайт будет отправлять зашифрованный уникальный идентификатор пользователю. Со стороны пользователя, при взаимодействии с сайтом, веб-браузер пользователя будет отправлять обратно эти данные с HTTP-запросом, которые будут расшифровываться и использоваться веб-сервером для авторизации.

Подход, в котором используются cookie для аутентификации и авторизации типичен для большинства современных существующих веб-сайтов. Без их использования невозможно определить, какой именно пользователь обращается к сайту.

2.6 Диаграмма вариантов использования

Диаграмма вариантов использования (другое название – диаграмма прецедентов) является исходным концептуальным представлением системы в процессе ее проектирования и разработки [8]. Данная диаграмма состоит из акторов, вариантов использования и отношений между ними. При построении диаграммы могут использоваться также общие элементы нотации: примечания и механизмы расширения.

Вариант использования представляет собой последовательность действий, выполняемых системой в ответ на событие, инициируемое некоторым действующим лицом. В общем случае, вариант использования описывает взаимодействие между пользователем и системой.

Действующее лицо (актор; от англ. Actor) – роль, которую пользователь играет по отношению к системе. Представляют собой роли, а не конкретных людей. Показывать на диаграмме действующих лиц следует только в том случае, когда им действительно необходимы некоторые варианты использования.

Выделяют несколько их типов связи между действующими лицами и вариантами использования, а именно: связи коммуникации (communication), включения (include), расширения (extend) и обобщения (generalization).

В ходе разработке были выделены следующие действующие лица (акторы):

- не аутентифицированный пользователь;
- пользователь (т.е. аутентифицированный пользователь);
- модератор группы;
- модератор сайта;
- администратор сайта.

Диаграмма вариантов использования разрабатываемого сайта представлена на рисунках 2.6-2.10.

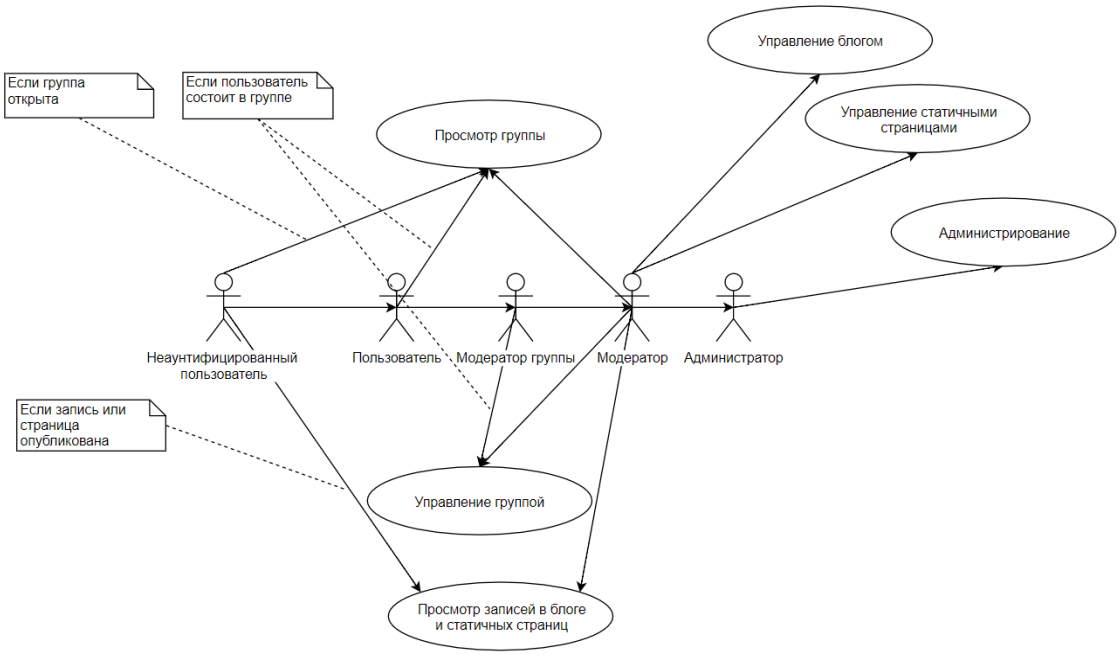


Рисунок 2.6 – Диаграмма вариантов использования разрабатываемого сайта

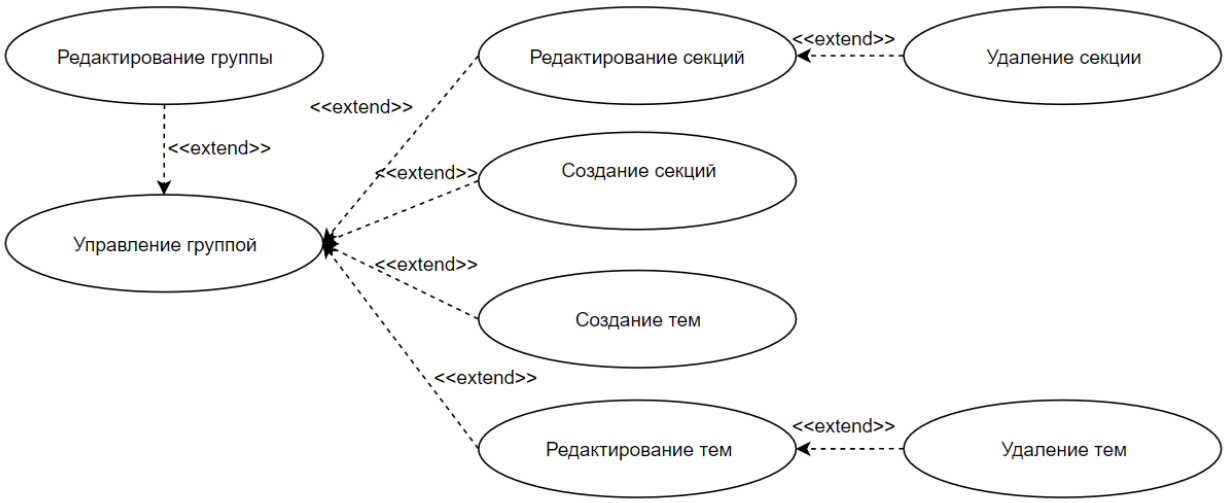


Рисунок 2.7 – Расширение диаграммы вариантов использования “Управление группой”

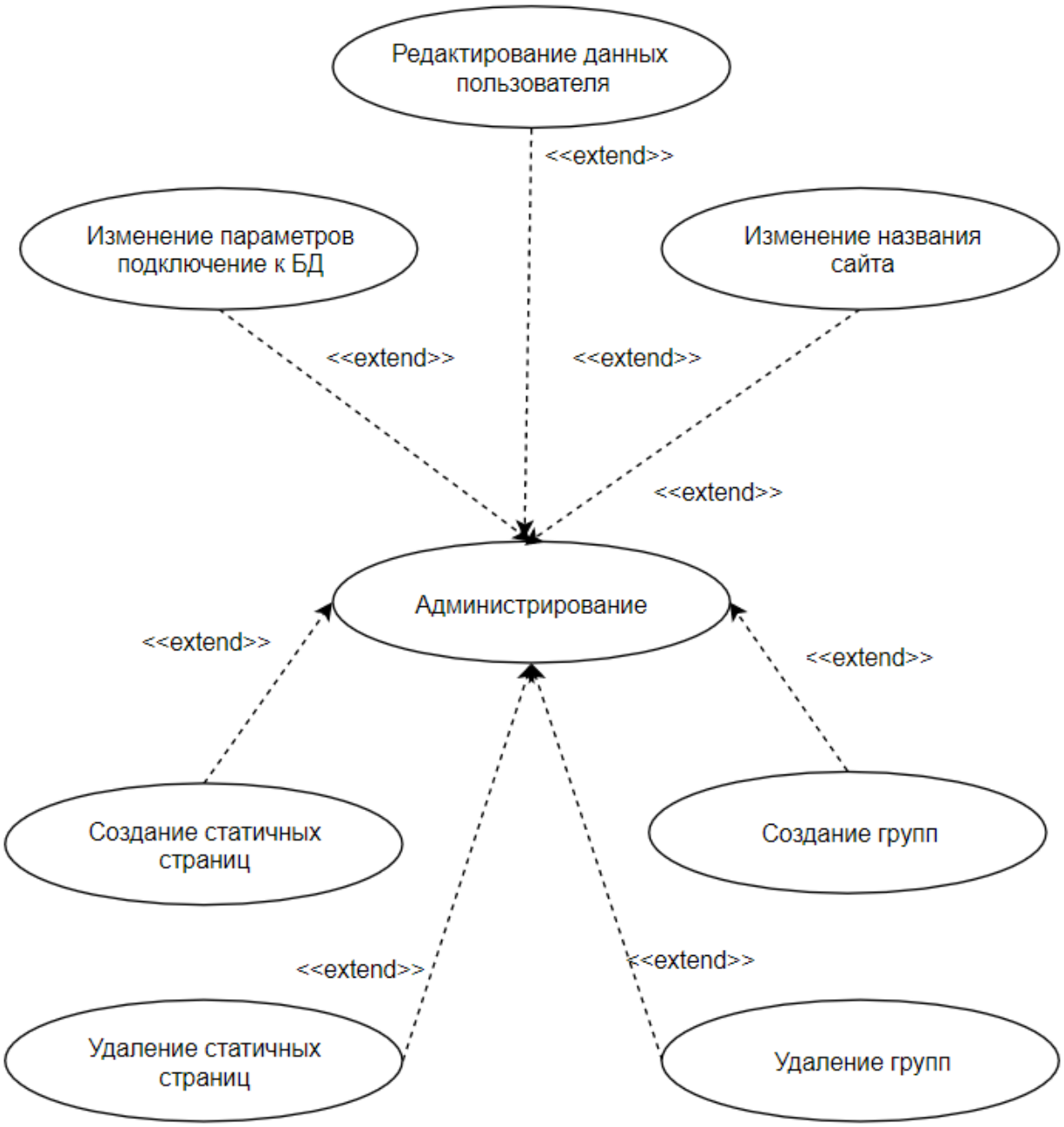


Рисунок 2.8 – Расширение диаграммы вариантов использования “Администрирование”

Рисунок 2.9 – Расширение диаграммы вариантов использования “Управление блогом”

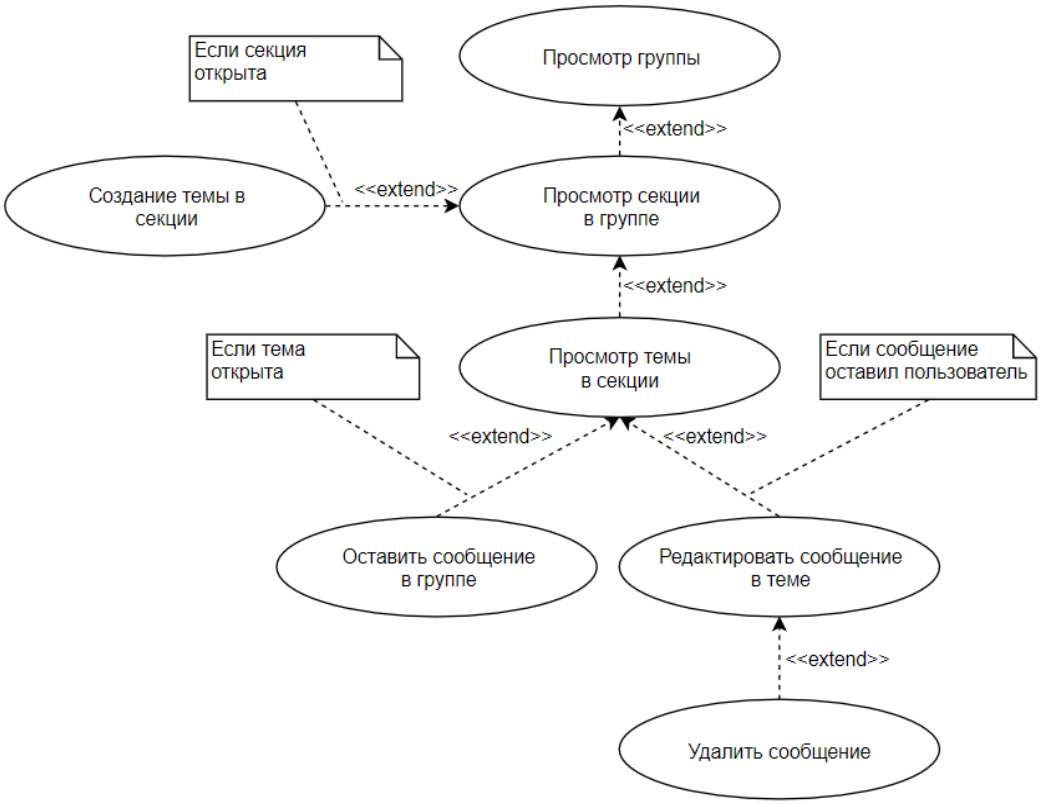


Рисунок 2.10 – Расширение диаграммы вариантов использования “Просмотр группы”

2.7 ER-модель

Модель сущность-связь (ER-модель) (англ. Entity-Relationship Model, ERM) — модель данных, позволяющая описывать концептуальные схемы предметной области. Данная модель была предложена Питером Ченом в 1976г. Использование ER-модели является основным подходом в проектировании баз данных, в особенности реляционных.

Основными понятиями ER-модели являются сущность, связь и атрибут. Сущность – это реальный или представляемый объект, информация о котором должна сохраняться и быть доступной. В диаграммах ER-модели сущность представляется в виде прямоугольника, содержащего имя сущности. При этом имя сущности – это имя типа, а не некоторого конкретного экземпляра этого типа. Связь – это графически изображаемая ассоциация между двумя сущностями, которая изображается в виде линии соединяющей сущности. Данная ассоциация является бинарной и может существовать между двумя разными сущностями или между сущностью и ей же самой (рекурсивная связь). В любой связи выделяются два конца (в соответствии с существующей парой связываемых сущностей), на каждом из которых указывается, сколько экземпляров данной сущности связывается.

Как правило, при разработке систем с использованием реляционных баз данных, в ER-модели отображают таблицы сущностей, которые будут в ней содержаться. Но так при разработке сайта будет использоваться ORM, в ER-модели будет содержаться информации о классах сущностей, с которыми будет оперировать сайт.

В ходе разработки архитектуры сайта были выделены 13 сущностей, описание которых представлены в таблицах 2.1–2.13. Диаграмма ER-модели разрабатываемого сайта представлена на рисунке 2.6.

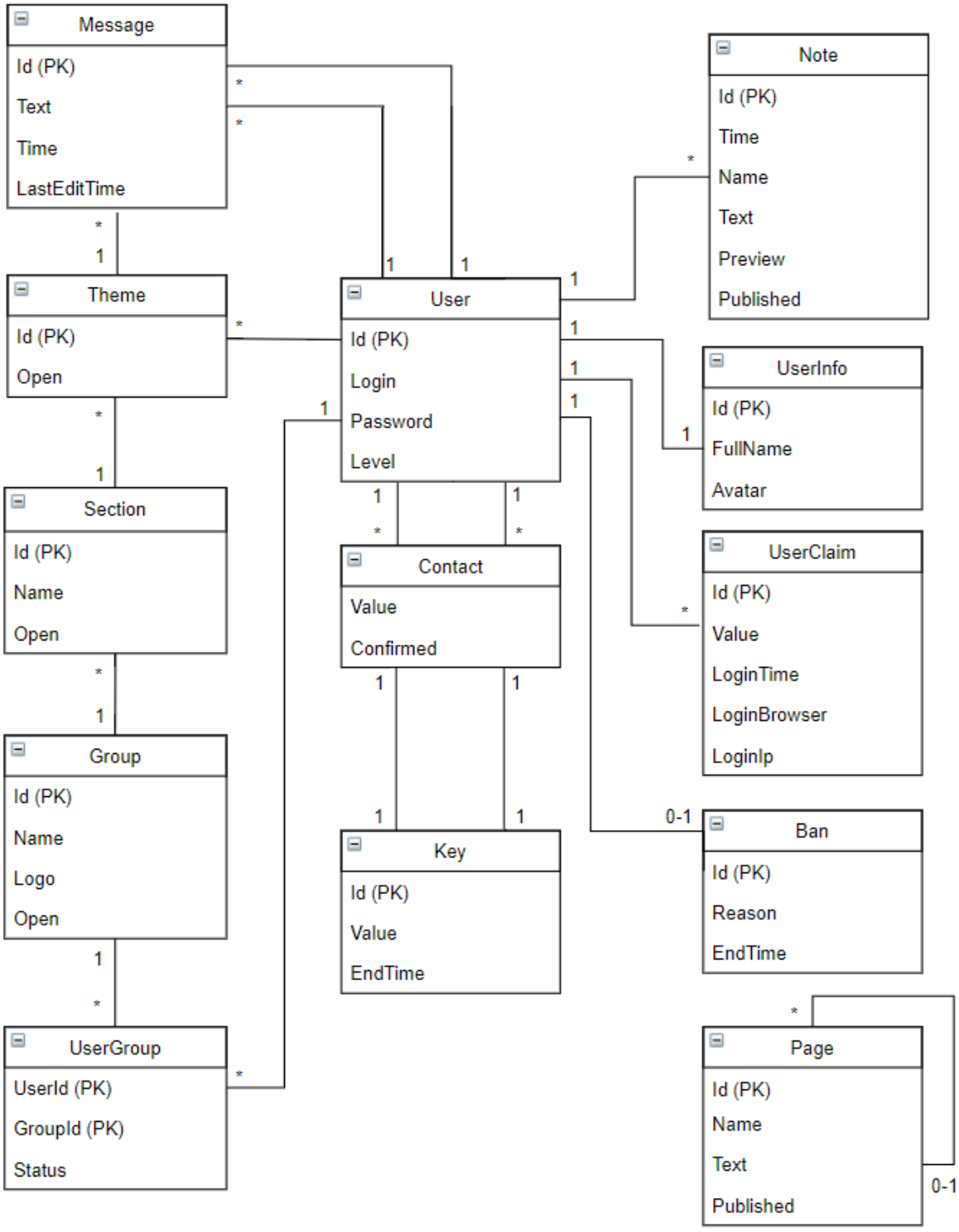


Рисунок 2.11 – Диаграмма ER-модели сайта

Таблица 2.1

Пользователь (user)		
Описание	Данная сущность содержит в себе основную информацию о пользователе.	
Название атрибута	Тип данных	Описание
Login	string	Уникальное имя пользователя
Password	string	Зашифрованный пароль пользователя
Level	byte	Уровень привилегий пользователя (0 – просто пользователь, 1- модератор, 2 – администратор)
Email	Contact	Информация об электронной почте пользователя
UserInfo	UserInfo	Дополнительная информация о пользователе
Phone	Contact	Информация о телефоне пользователя
Ban	Ban	Информация о том, заблокирован ли пользователь или нет.

Таблица 2.2

Контактные данные (Contact)		
Описание	Информация о контактных данных пользователя (телефон и почта)	
Название атрибута	Тип данных	Описание
Value	string	Номер телефона или адрес электронной почты
Confirmed	bool	Подтвержден ли контакт
Key	Key	Информация о ключе, который отправляется на телефон или почту для авторизации.
ConfirmKey	Key	Информация о ключе, который отправляется на телефон или почту для подтверждения владения контактом

Таблица 2.3

Ключ (Key)		
Описание:	Содержит в себе информацию о ключе, который отправляется на телефон или почту при двухфакторной авторизации	
Название атрибута	Тип данных	Описание
Value	string	Значение ключа

Продолжение таблицы 2.2

EndTime	DateTime	Дата и время до которых данный ключ действителен

Таблица 2.4

Блокировка (Ban)		
Описание:	Информация об блокировке пользователя на сайте	
Название атрибута	Тип данных	Описание
Reason	string	Причина блокировки, которую указывает администратор
EndTime	DateTime	Дата и время разблокировки

Таблица 2.5

Данные об авторизации пользователя (UserClaim)		
Описание:	Содержит в себе информацию об авторотационных данных пользователя для последующей его аутентификации при работе с сайтом	
Название атрибута	Тип данных	Описание
User	User	Пользователь
Value	string	Уникальное значение, с помощью которого можно точно определить пользователя, будет содержаться в куках (Cooke)
LoginTime	DateTime	Дата и время авторизации
LoginBrowser	string	Название браузера, с помощью которого была произведена авторизация
LoginIP	string	IP-адрес, с которого была выполнена авторизация

Таблица 2.6

Дополнительные данные пользователя (UserInfo)		
Описание:	Дополнительная информация о пользователе	
Название атрибута	Тип данных	Описание
FullName	string	Фамилия, имя и отчество пользователя
Avatar	string	Ссылка на расположение изображения (аватара) пользователя

Таблица 2.7

Группа (Group)		
Описание:	Сущность отдельной группы на сайте	
Название атрибута	Тип данных	Описание

Продолжение таблицы 2.7

Name	string	Название группы
Logo	string	Ссылка на изображение (логотип) группы
Open	bool	Открытая или закрытая группа

Таблица 2.8

Пользователь-группа (UserGroup)		
Описание:	Вспомогательная сущность, которая обеспечивает связь многие ко многим между пользователями и группами	
Название атрибута	Тип данных	Описание
User	User	Пользователь
Group	Group	Группа, в которой состоит пользователь
Status	byte	Уровень доступа к группе (0 – простой пользователь, 1 – модератор группы)

Таблица 2.9

Секция (Section)		
Описание:	Сущность отдельной секции в группе	
Название атрибута	Тип данных	Описание
Name	string	Название секции
Group	Group	Группа, в которой находится секция
Open	bool	Открыта секция, или нет

Таблица 2.10

Тема (Theme)		
Описание:	Сущность отдельной темы внутри секции	
Название атрибута	Тип данных	Описание
Name	string	Название
Open	bool	Открытая или закрытая тема
Section	Section	Секция, внутри которой располагается тема

Таблица 2.11

Сообщение (Message)		
Описание:	Сообщение внутри темы	
Название атрибута	Тип данных	Описание
Owner	User	Пользователь создавший сообщение
LastEditor	User	Пользователь, который последний раз редактировал сообщение

Продолжение таблицы 2.11

Theme	Theme	Тема, в которой располагается с сообщение
Text	string	Текст сообщения
Time	DateTime	Дата и время, когда было написано сообщение
LastEditTime	DateTime	Дата и время последнего редактирования сообщения

Таблица 2.12

Запись в блоге (Note)		
Описание:	Сообщение внутри темы	
Название атрибута	Тип данных	Описание
Owner	User	Пользователь, который создал запись
Preview	string	Ссылка на изображение-превью записи
Name	string	Название записи
Text	string	Текст записи
Published	bool	Опубликовано ли сообщение
Time	DateTime	Дата и время создания записи

Таблица 2.13

Статичная страница (Page)		
Описание:	Сообщение внутри темы	
Название атрибута	Тип данных	Описание
Name	string	Название страницы
Text	string	Текст страницы
Published	bool	Опубликовано ли сообщение
ParentPage	Page	Родительская страница

2.8 Диаграмма размещения

Диаграмма размещения отражает физические взаимосвязи между программными и аппаратными компонентами системы. Она является хорошим средством для того, чтобы показать маршруты перемещения объектов и компонентов в распределенной системе [9]. На рисунке 2.6 представлена диаграмма размещения разрабатываемого сайта.

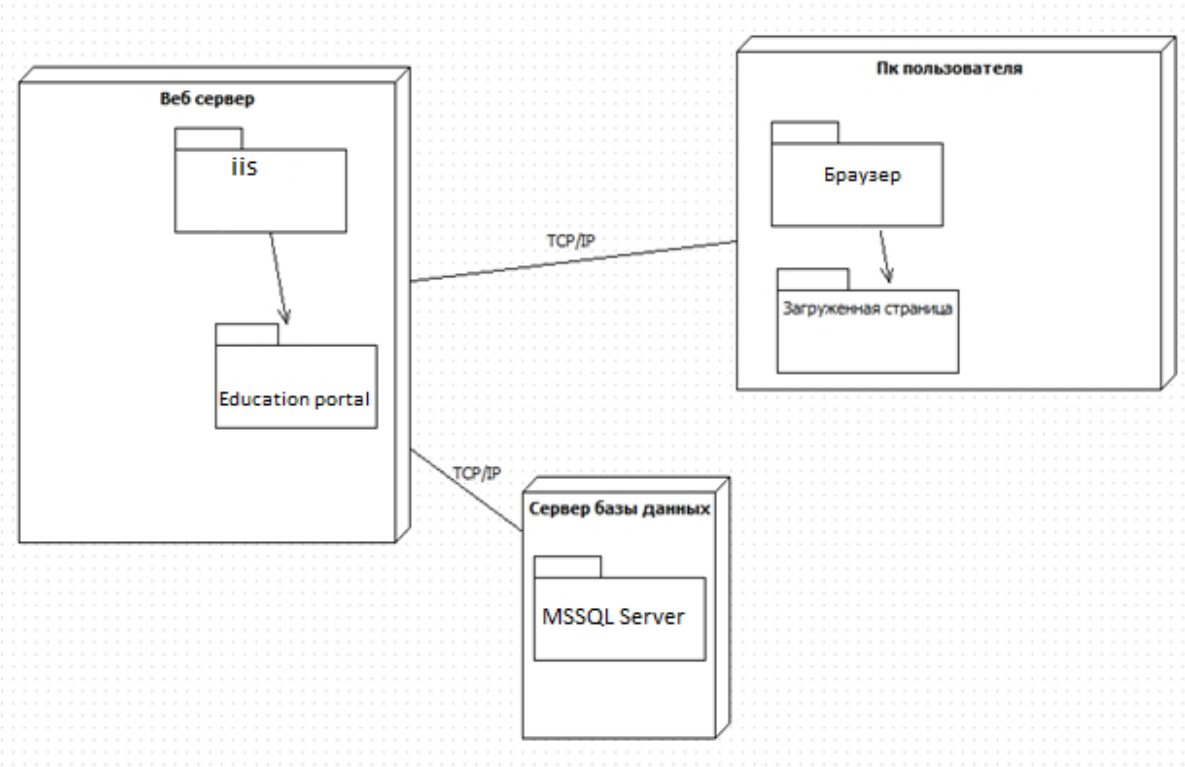


Рисунок 2.12 – Диаграмма размещения разрабатываемого сайта

2.9 Выводы по разделу

В данном разделе был произведен обзор архитектурных моделей приложений, из которых была выбрана трехуровневая архитектура, были представлены: MVC, паттерны проектирования Unit of Work и Data Transfer Object, которые будут использоваться при разработке сайта. Были продемонстрированы: ER-модель, диаграмма вариантов использования и диаграмма размещения разрабатываемого сайта.

3 РАЗРАБОТКА ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА

Пользовательский интерфейс (от англ. User interface, UI) – это элементы и компоненты программы, оказывающие влияние на взаимодействие пользователя с программным обеспечением; это совокупность правил, методов и программно-аппаратных средств, обеспечивающих взаимодействие пользователя с компьютером [10].

В подавляющем большинстве случаев под термином «пользовательский интерфейс» имеется в виду графический интерфейс пользователя (от англ. graphical user interface, GUI) — разновидность пользовательского интерфейса, в котором человек взаимодействует с системой при помощи графических компонентов (окна, пиктограммы, полосы прокрутки и т.п.), отображаемых на экране.

Пользовательский интерфейс часто воспринимают только как внешний вид программы. В действительности он включает все аспекты, оказывающие влияние на взаимодействие пользователя и системы, и определяется в первую очередь такими факторами, как характер и набор задач пользователя, которые он решает с помощью системы, а также возможности устройств ввода-вывода информации и вычислительных ресурсов системы.

Пользовательский интерфейс означает среду и метод общения человека с компьютером (совокупность приемов взаимодействия с компьютером). Он часто отождествляется с диалогом, который подобен диалогу или взаимодействию между двумя людьми.

При разработке графического интерфейса веб-сайта следует учесть следующие факторы:

- лёгкость чтения информации. Текст должен легко различаться на цвете фона. Похожие оттенки заставляют посетителя напрягать зрение;
- простота навигации. Навигация по сайту не должна вызывать трудностей;
- удобство просмотра информации. Сайт должен адаптироваться под разные разрешения окна браузера;
- отображение во всех браузерах. Дизайн сайта должен одинаково хорошо отображаться в любом браузере, а также в разных версиях одного браузера.

3.1 Технологии разработки графического интерфейса пользователя веб-сайтов

3.1.1 Язык разметки гипертекста

Язык разметки гипертекста (от англ. Hypertext Markup Language, HTML) – это основной язык разметки, используемый для отображения веб-страниц в

Интернете [11]. Другими словами, веб-страницы состоят из HTML, который используется для отображения текста, изображений или других ресурсов через веб-браузер. HTML – это простой текст, то есть он не компилируется и может быть прочитан людьми.

Когда пользователь открывает свой веб-браузер и набирает веб-адрес (URL) веб-страницы, браузер отправляет запрос на сервер. Затем веб-браузер получает документ HTML и связанные с ним файлы, в том числе: изображения, видео и другие необходимые файлы, и преобразовывает текст разметки, который содержится в HTML файл в изображение, которое выводится в окно браузера.

HTML5 является пятой версией стандарта HTML. Разработка стартовала 2007 году, а повсеместное использование в веб-сайтах HTML5 началось с 2010 года. Окончательный стандарт официально был принят 28 октября 2014 года.

Предыдущий стандарт HTML, HTML 4.01, был принят в 1999 году – за пятнадцать лет до стандарта HTML5. Однако в десятилетие, предшествовавшее HTML5, большинство веб-сайтов было написано на XHTML, более строгой версии HTML, опубликованной в 2000 году. HTML5 был создан как единый язык разметки, который мог бы сочетать синтаксические нормы HTML и XHTML. Он расширяет, улучшает и рационализирует разметку документов, а также добавляет новые возможности, например, возможность добавления в веб-страницу видеозаписей, аудиозаписей и математических формул.

За последние несколько десятилетий глобальная сеть интернет пережила множество изменений, но HTML всегда был и остается основным языком, используемым для разработки веб-страниц.

3.1.2 Каскадные таблицы стилей

CSS (от англ. Cascading Style Sheets — каскадные таблицы стилей) – язык описания внешнего вида документа, написанного с использованием языка разметки, как правило HTML.

CSS используется разработчиками веб-страниц для задания цветов, шрифтов, расположения отдельных элементов страницы и других аспектов представления внешнего вида этих веб-страниц [12].

Основной целью разработки CSS являлось разделение описания логической структуры веб-страницы (которое производится с помощью HTML или других языков разметки) от описания внешнего вида этой веб-страницы (которое теперь производится с помощью формального языка CSS). Такое разделение может увеличить доступность документа, предоставить большую гибкость и возможность управления его представлением, а также уменьшить сложность и повторяемость в

структурном содержимом. Кроме того, CSS позволяет представлять один и тот же документ в различных стилях, например, если необходимо создать версию страницы для слабовидящих, нет необходимости переписывать саму страницу, достаточно описать отдельный стиль с помощью CSS.

Основной недостаток CSS – возможность разного отображения одной и той-же страницы в браузерах разных производителей, так как разные браузеры могут по-разному интерпретировать CSS и не поддерживать некоторые методы, которые поддерживают другие языки. При разработке веб-сайтов следует учитывать данную проблему и тестировать отображение разрабатываемого веб-сайта в различных браузерах.

3.1.3 Язык программирования JavaScript

JavaScript (JS) – это язык программирования, в основном используемый в Интернете. Он применяется вместе с HTML-страницами. JavaScript – это интерпретируемый язык, то есть образом, его не нужно компилировать.

JavaScript позволяет делает веб-страницы интерактивным и динамичным. Это позволяет страницам реагировать на действия пользователя, проявлять специальные эффекты, проверять данные, создавать файлы cookie, определять браузер пользователя и т. д.

3.1.4 Технология AJAX

AJAX (от англ. Asynchronous JavaScript and XML) – это технология взаимодействия с веб-сервером без перезагрузки веб-страницы. Поскольку не требуется каждый раз обновлять страницу целиком, повышается скорость работы с сайтом и удобство его использования.

Технология AJAX предоставляет широкий спектр возможностей, например, при прохождении регистрации на некоторых сайтах пользователь вводит логин – и через мгновение ему выдается информация о том, свободен он или нет. Также при введении поискового запроса в Google после каждой буквы или слова предлагается несколько вариантов запроса. Это значительно повышает комфорт работы с сайтами.

3.1.5 Фреймворк jQuery

jQuery – распространённая библиотека JavaScript, основная цель которой упрощение взаимодействия JavaScript и HTML. Также библиотека jQuery предоставляет удобные средства для работы с AJAX.

Одной из главных особенностей jQuery является ее независимость от браузера, в том смысле, что ее функции будут выдавать одинаковый результат во всех современных браузерах, что не гарантируется при использовании стандартных функций JavaScript.

3.2 Диаграмма переходов между страницами

Диаграмма переходов между страницами представляет собой схематичное представление переходов между разными частями программы. С точки зрения сайта, диаграмма показывает, каким образом пользователь будет переходить между страницами, иными словами показывает навигацию между страницами на сайте.

Правильно разработанная навигация является одним из важнейших факторов в удобстве пользования сайтом. В противном случае пользователю будет сложно отыскать нужную ему информацию при использовании сайта.

Для удобства пользователя, форум внутри группы можно будет разделить на отдельные секции. Внутри секций будут располагаться темы форума.

Статические страницы – информационные страницы, на которых будут располагаться информация об учреждении, которая редко меняется.

Диаграмма переходом между страницами разрабатываемого сайта представлена на рисунке 3.1

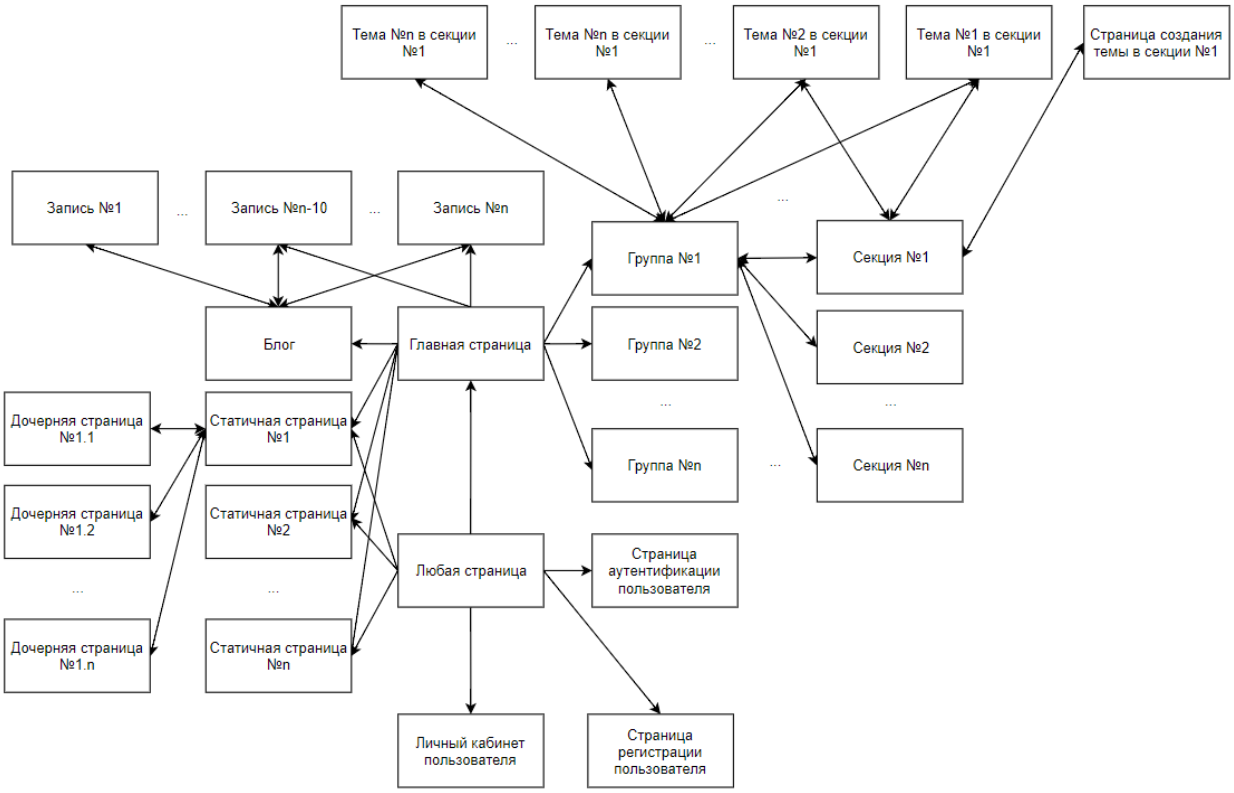


Рисунок 3.1 – Диаграмма переходом между страницами разрабатываемого сайта

3.3 Разработка графического макета сайта

Графический макет сайта – это демонстрация графического пользовательского интерфейса будущего сайта: как он будет выглядеть в течении эксплуатации, после того, как наполнится информацией [13]. Макет

сайта не интерактивный – то есть пункты меню на этой стадии разработки сайта не активны. Такой макет представляет собой только набор картинок – и называется поэтому графическим макетом. Макеты разрабатываемого сайта представлены на рисунках 3.1–3.5.

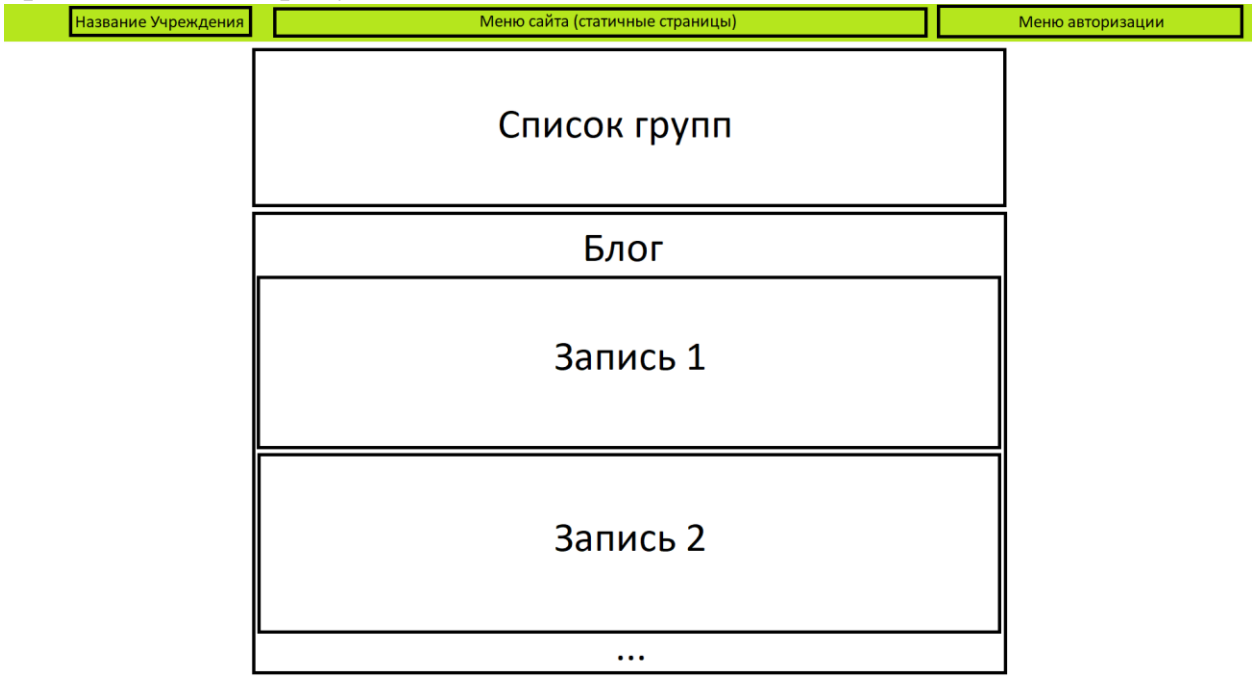


Рисунок 3.2 – Графический макет главной страницы разрабатываемого сайта

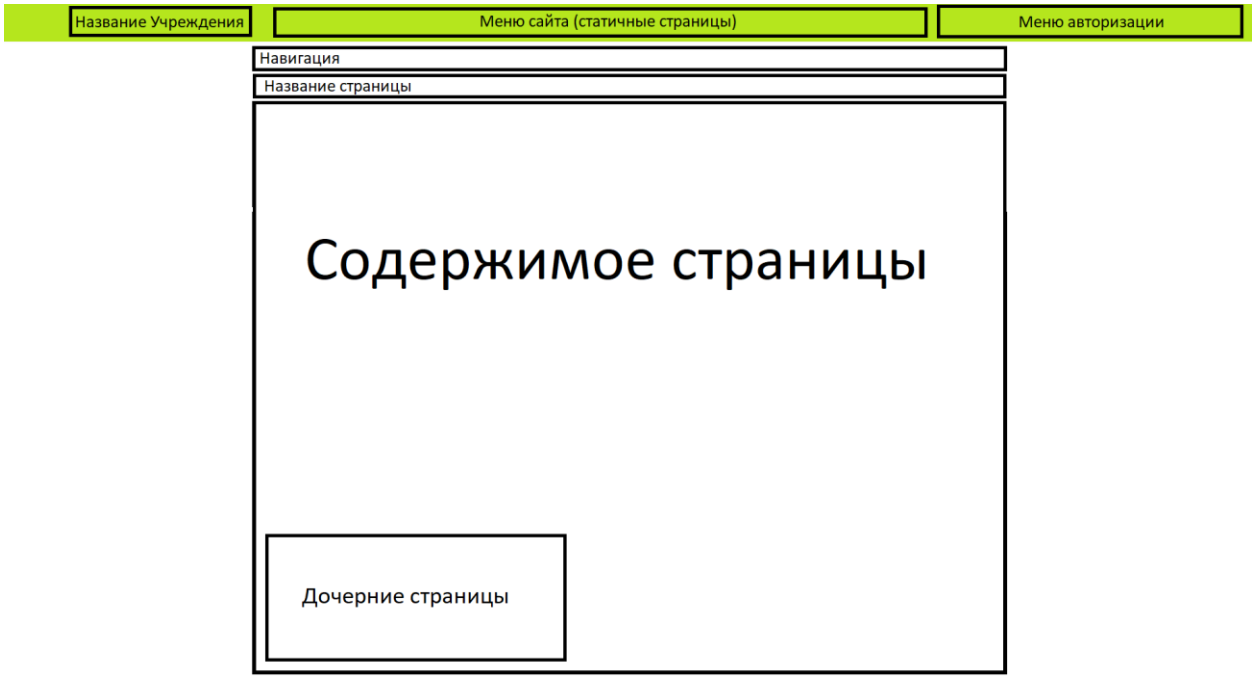


Рисунок 3.3 – Графический макет статичной страницы разрабатываемого сайта

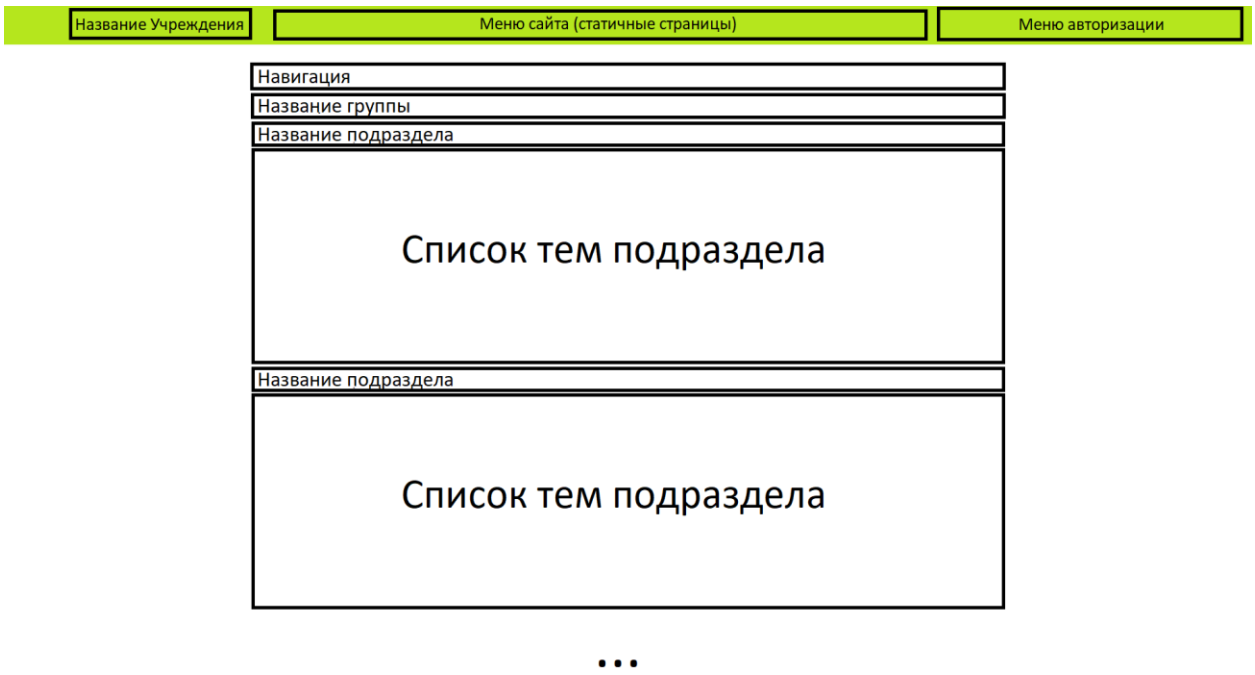


Рисунок 3.4 – Графический макет страницы группы разрабатываемого сайта

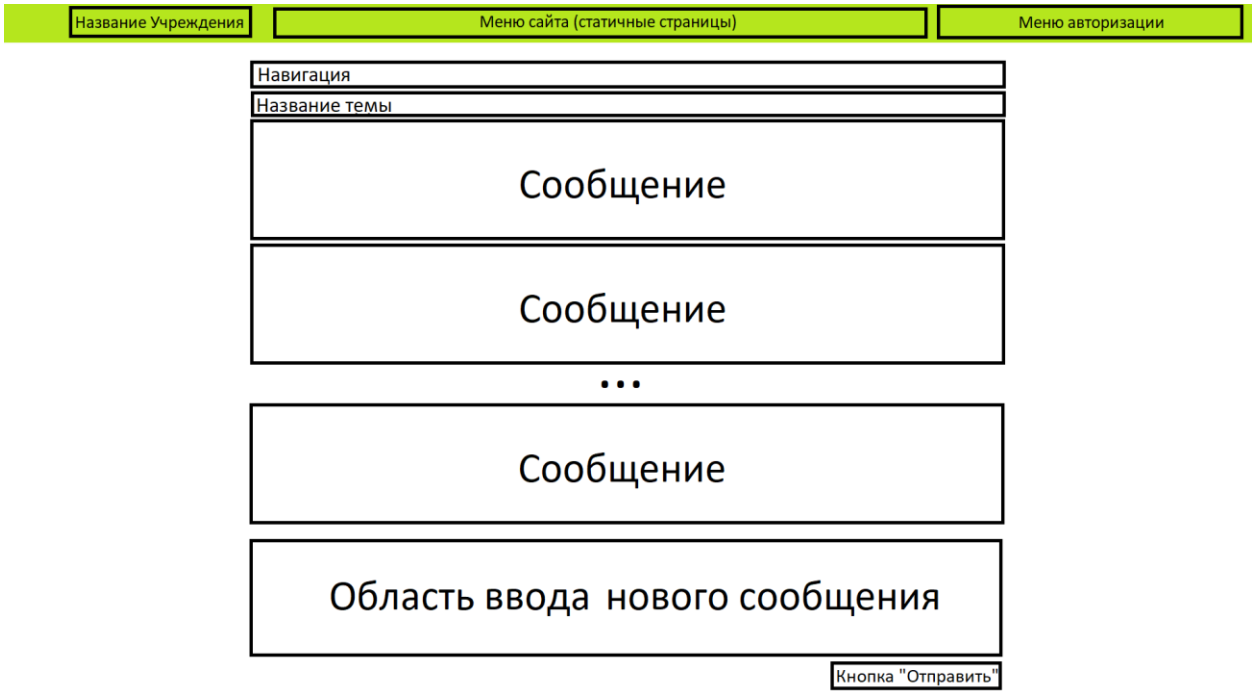


Рисунок 3.5 – Графический макет страницы темы разрабатываемого сайта

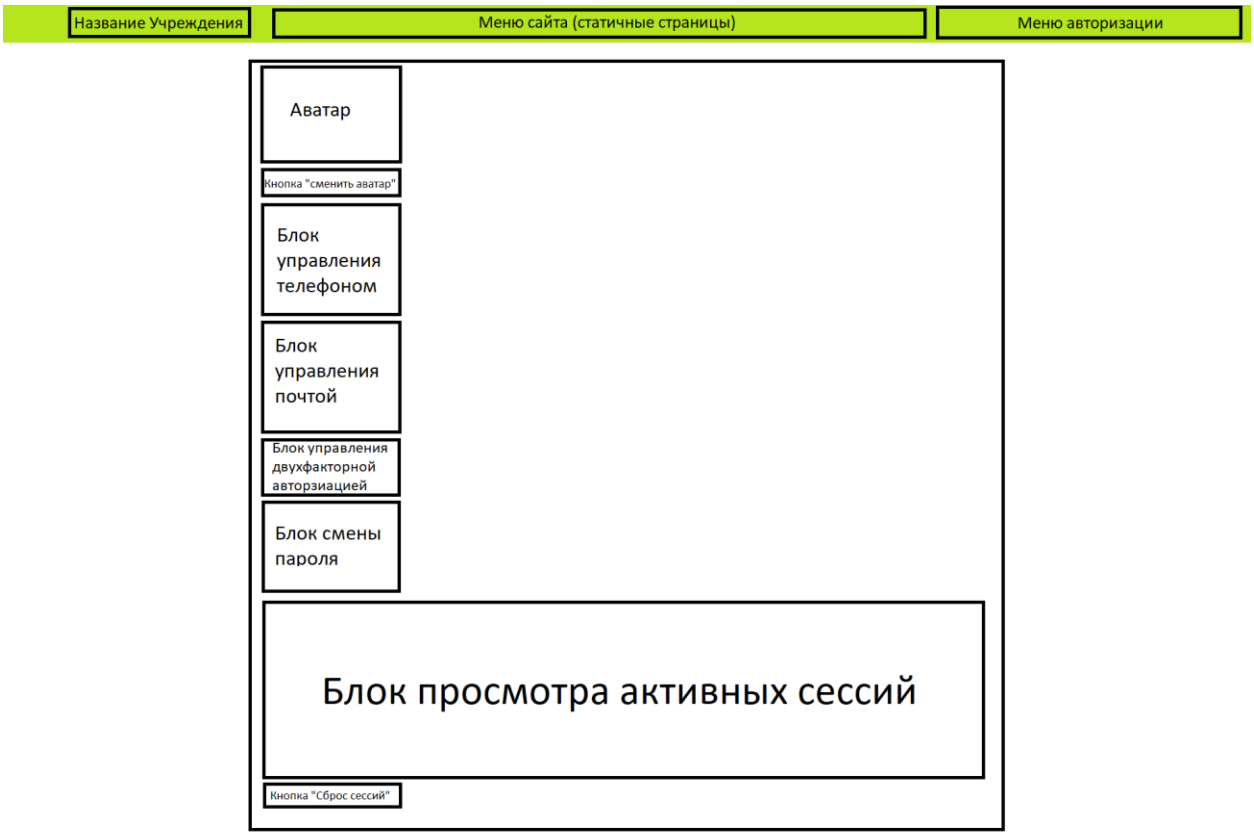


Рисунок 3.6 – Графический макет личного кабинета пользователя разрабатываемого сайта

3.4 Концепция Material Design

Material Design – современная концепция графического интерфейса пользователя приложений, разработанная компанией Google для своих проектов [14]. В данной концепции все в интерфейсе выглядит и ведет себя, как бумага. Экраны – это листы бумаги, а на них – чернила. В дизайне используются тени, отбрасываемые, когда один лист располагается над другим. Бумага, тем не менее, имеет и ряд нереалистичных свойств, например, у нее нет текстуры, а листы могут бесшовно соединяться и разделяться. Стиль Google Material Design в целом поддерживает тенденции минимализма, использования ярких цветов и разнообразных шрифтов.

Использование этой концепции поможет создать красивый и современный интерфейс в разрабатываемом сайте.

Важно при использовании данной концепции выбрать яркие и сочетаемые цвета, которые будут использоваться в приложении, как правило три или четыре цвета. Для разрабатываемого сайта был выбран белый цвет для фона и черный цвет для основного текста, что хорошо скажется на читаемости. В качестве вспомогательного цвета был выбран ярко-синий (#4083ff), а текст, который будет располагаться на нем, будет белым.

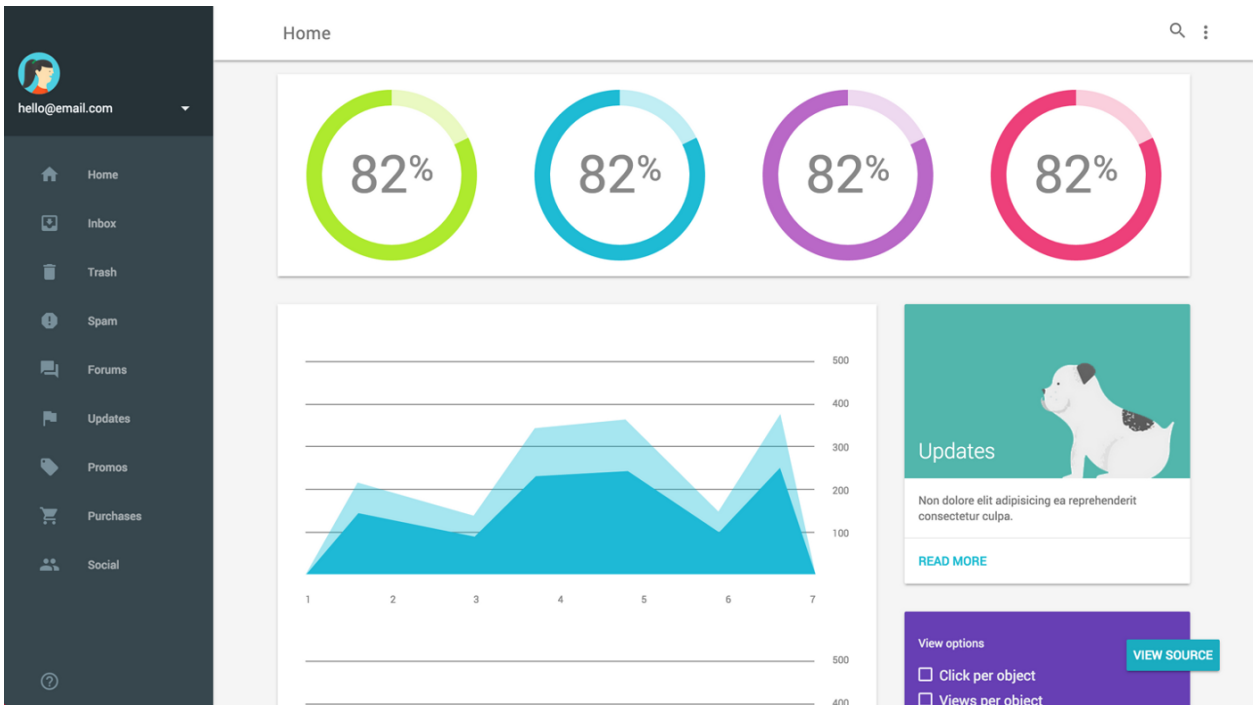


Рисунок 3.7 – Пример сайта, созданного с использованием концепции Material Design

3.5 Выводы по разделу

В разделе 3.1 были представлены технологии разработки графического интерфейса пользователей: HTML, CSS, JavaScript, AJAX и jQuery. В разделах 3.3 и 3.4 были продемонстрированы макет разрабатываемого сайта и диаграмма переходов между страницами веб-сайта. В разделе 3.5 была показана концепция Material Design, которая позволяет разрабатывать красивый и современный пользовательский интерфейс, так же были выбраны основные цвета для сайта.

4 РАЗРАБОТКА ПРОГРАММНОГО КОДА

4.1 Принципы разработки

Принципы разработки ПО — это набор определенных правил и рекомендаций, которым желательно следовать при написании исходного кода программы. Принципы разработки ПО помогают разрабатывать более качественный код, который легче читать и поддерживать в будущем. Ниже приведены основные принципы, которые использовались при разработке сайта

4.2 Принципы SOLID

SOLID – акронимом первых пяти принципов объектно-ориентированного программирования (ООП), которые были выдвинуты Робертом К. Мартином в начале 2000-х [15]. В наше время данные принципы являются де-факто стандартом при написании программ в объектно-ориентированных языках программирования.

Акроним образован по первым буквам названий SOLID-принципов:

- Single Responsibility Principle (Принцип единственной обязанности)
- Open/Closed Principle (Принцип открытости/закрытости)
- Liskov Substitution Principle (Принцип подстановки Лисков)
- Interface Segregation Principle (Принцип разделения интерфейсов)
- Dependency Inversion Principle (Принцип инверсии зависимостей)

Эти принципы, в сочетании друг с другом, помогают программисту разрабатывать программное обеспечение, которое легко поддерживать, расширять и изменять.

4.2.1 Принцип единственной ответственности

Принцип единственной ответственности (от англ. Single Responsibility Principle) гласит, что на каждый класс должна быть возложена одна-единственная обязанность. Не должно быть более одной причины для изменения класса.

Конкретное применение принципа зависит от контекста. Если класс выполняет несколько различных функций, и они изменяются по отдельности, то стоит применить принцип единственной обязанности. Иными словами, у класса несколько причин для изменения. Например, если у нас есть класс, который создает отчет и выводит его на печать, то стоит разделить этот класс на два, на тот, который создает отчет, и на тот, который этот отчет отправляет на печать. Если же функции решают одну задачу, то нет смысла применять данный принцип.

4.2.2 Принцип открытости/закрытости

Принцип открытости/закрытости (от англ. Open/Closed Principle) – программные объекты (классы, модули, функции и т. д.) должны быть открыты для расширения, но закрыты для модификации. Суть этого принципа состоит в том, что программа должна быть построена таким образом, что все ее последующие изменения должны быть реализованы с помощью добавления нового кода, а не изменения уже существующего.

Как правило программные проекты в течении своего жизненного цикла постоянно изменяются, например, из-за добавления новых функции, или изменения старых, что может занять довольно большое время из-за некачественного кода, когда для этого нужно изменять другие уже готовые и протестированные участки кода. Использование данного принципа позволяет этого избежать.

4.2.3 Принцип подстановки Лисков

Принцип подстановки Лисков (от англ. Liskov Substitution Principle) – если для каждого объекта o1 типа S существует объект o2 типа T, такой, что для любой программы P, определенной в терминах T, поведение P не изменяется при замене o2 на o1, то S является подтипом T. Иными словами, если S – класс наследник класса T, то замена объектов T в программе на объекты S не должно приводить к изменению работы программы. Таким образом, классы наследники должны расширять функционал родительского класса, без изменения базового функционала.

4.2.4 Принцип разделения интерфейсов

Принцип разделения интерфейсов (от англ. Interface Segregation Principle) гласит, что Клиенты не должны вынужденно зависеть от методов, которыми не пользуются. При нарушении этого принципа клиент, использующий некоторый интерфейс со всеми его методами, зависит от методов, которыми не пользуется, и поэтому оказывается восприимчив к изменениям в этих методах. В конечном счете это приводит к жесткой зависимости между различными частями интерфейса, которые могут быть не связаны при его реализации.

В этом случае интерфейс разделяется на отдельные части. После этого эти интерфейсы могут независимо применятся и изменяться, что делает систему менее связной, и тем самым ее легче модифицировать и обновлять.

4.2.4 Принцип разделения интерфейсов

Принцип разделения интерфейсов (от англ. Dependency Inversion Principle) – Модули верхнего уровня не должны зависеть от модулей нижнего уровня. И те, и другие должны зависеть от абстракций. Абстракции не должны зависеть от деталей. Детали должны зависеть от абстракций. Основной смысл принципа инверсии зависимостей прост: вместо создания зависимостей напрямую, класс должен требовать их у более высокого уровня через аргументы метода или конструктора. При этом зависимость должна передаваться не в виде экземпляров конкретных классов, а в виде интерфейсов или абстрактных классов.

4.3 Принцип KISS

KISS (акроним для “keep it short and simple”) – принцип проектирования, разработанный в ВМС США в 1960-х годах [16]. Данный принцип утверждает, что большинство систем работают лучше всего, если они остаются простыми. Принцип декларирует простоту системы в качестве основной цели и/или ценности.

С точки зрения разработки ПО, следует разрабатывать как можно более простую программу, которая будет отвечать заданным требованиям. Не имеет смысла закладывать в программу функции, которые не будут использоваться, или их использование будет маловероятным, так как большинству пользователей достаточно базового функционала, а увеличение сложности только вредит удобству использования программы. Нет необходимости учитывать все возможные варианты поведения как системы, так и пользователя, так как это просто невозможно, а принятие в расчет всех маловероятных событий просто иррационально с точки зрения времени и стоимости разработки. Так же не всегда есть необходимость в абсолютной математической точности, данные необходимо обрабатывать с той точность, которая достаточна для качественно решения задачи.

4.4 Выводы по разделу

В данном разделе были представлены основные принципы, которые были использованы при разработке программного кода веб-сайта. Принципы SOLID и KISS позволяют разработать качественный программный код, который является легко читаемым и не вызывает проблем при дальнейшей разработке и поддержки.

5 ОПИСАНИЕ ПРОГРАММЫ

5.1 Главная страница сайта

На главной странице сайта (см. рисунок 5.1) расположен список групп и блог.

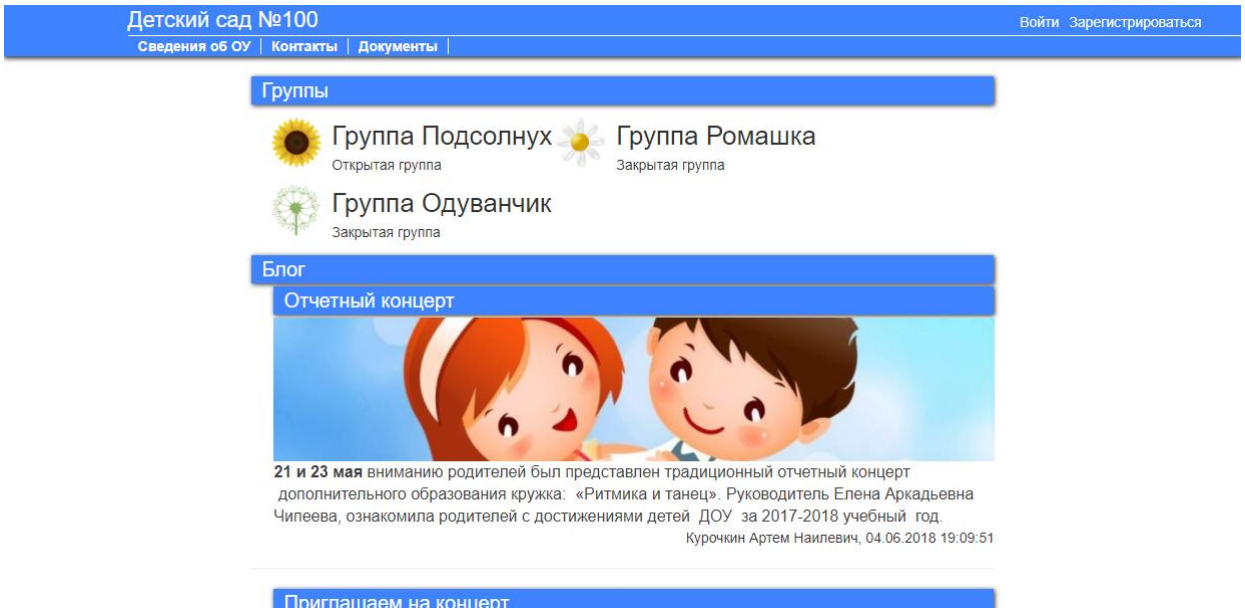


Рисунок 5.1 – Главная страница сайта

5.2 Страница регистрации

Чтобы зарегистрироваться на сайте, необходимо нажать на ссылку “Зарегистрироваться” в верхней части сайта (см. рисунок 5.1).

На странице регистрации необходимо заполнить поля и нажать на флажок “я не робот”, после чего появится кнопка зарегистрироваться (см. рисунок 5.2). Важно отметить, что при регистрации не обязательно указывать номер телефона и адрес электронной почты, достаточно указать что-то одно.

Поля, в которые введено некорректное значение помечаются красным цветом. В случае, если хотя бы в одном поле введено некорректное значение, то кнопка “зарегистрироваться” не появится (см рисунок 5.3). В целом это сделано для удобства пользователя, и не влияет на работу серверной части, так как в любом случае регистрационные данные ещё раз будут проверены, но уже на стороне сервера.

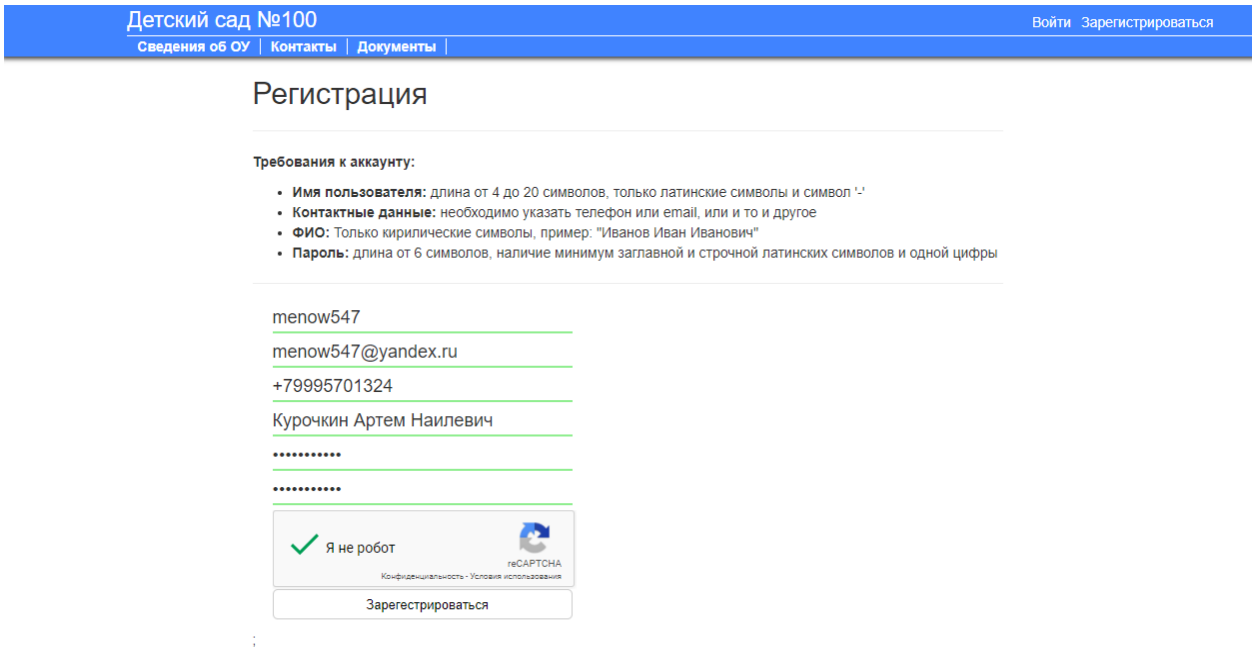


Рисунок 5.2 – Страница регистрации пользователя

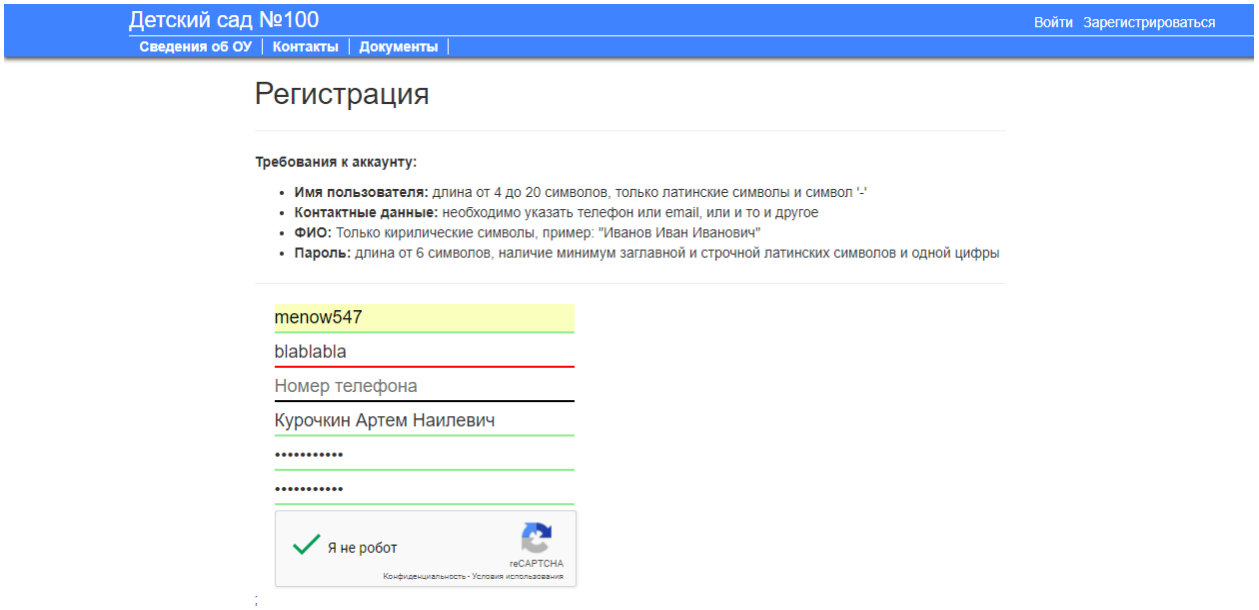


Рисунок 5.3 – Страница регистрации пользователя с некорректно введённым значением

5.3 Страница аутентификации

Вход в аккаунт происходит через страницу аутентификации, на которую возможно попасть через ссылку “Войти” в верхней части сайта. Для аутентификации можно использовать имя аккаунта, либо адрес электронной почты, либо номер телефона.

Если будет введен неправильный логин или пароль, сайт выдаст сообщение о том, что такой пользователь не найден (см. рисунок 5.5).

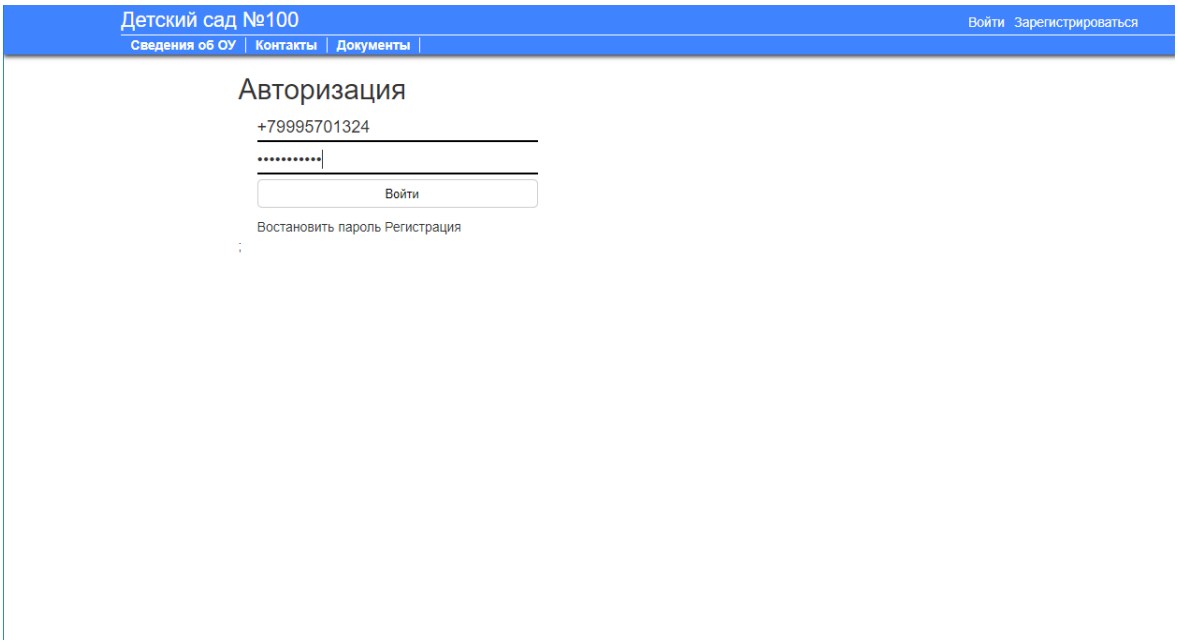


Рисунок 5.4 – Страница аутентификации

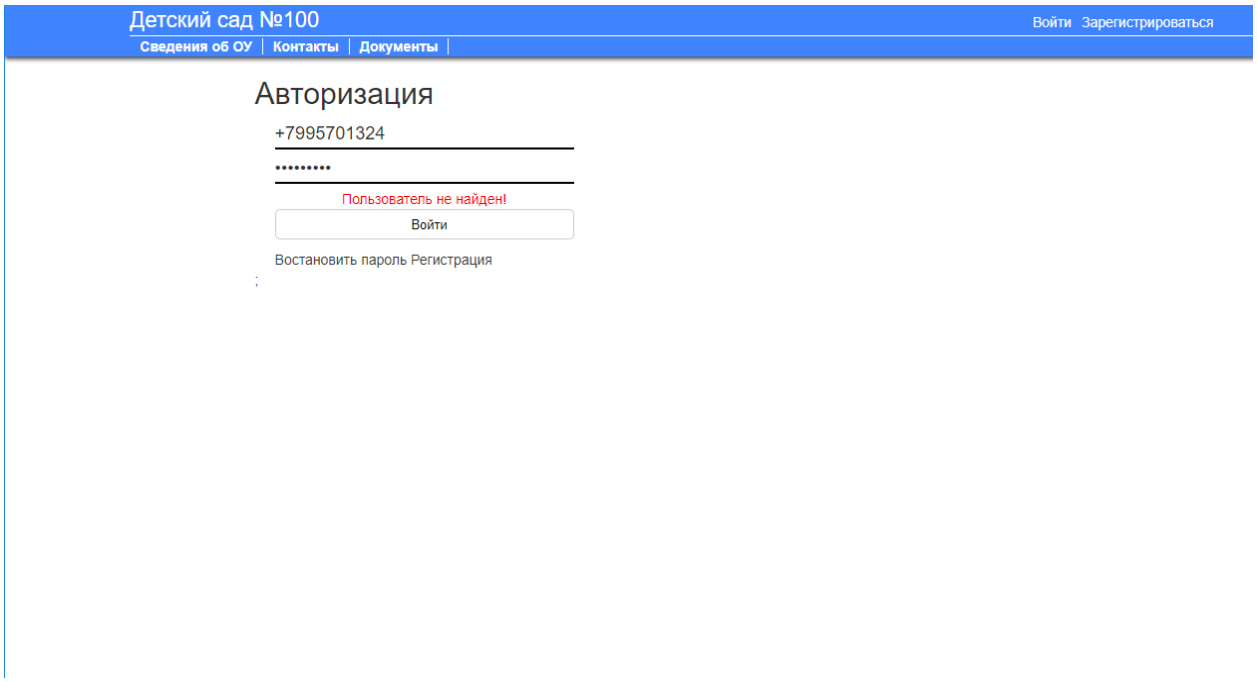


Рисунок 5.5 – Страница аутентификации при неправильно введённом логине или пароле

Если у пользователя включена двухфакторная аутентификация, то после ввода логина и пароля, на его телефон или почту будет отправлен код, который необходимо ввести в сплывающее окно и нажать кнопку войти (см рисунок 5.6).

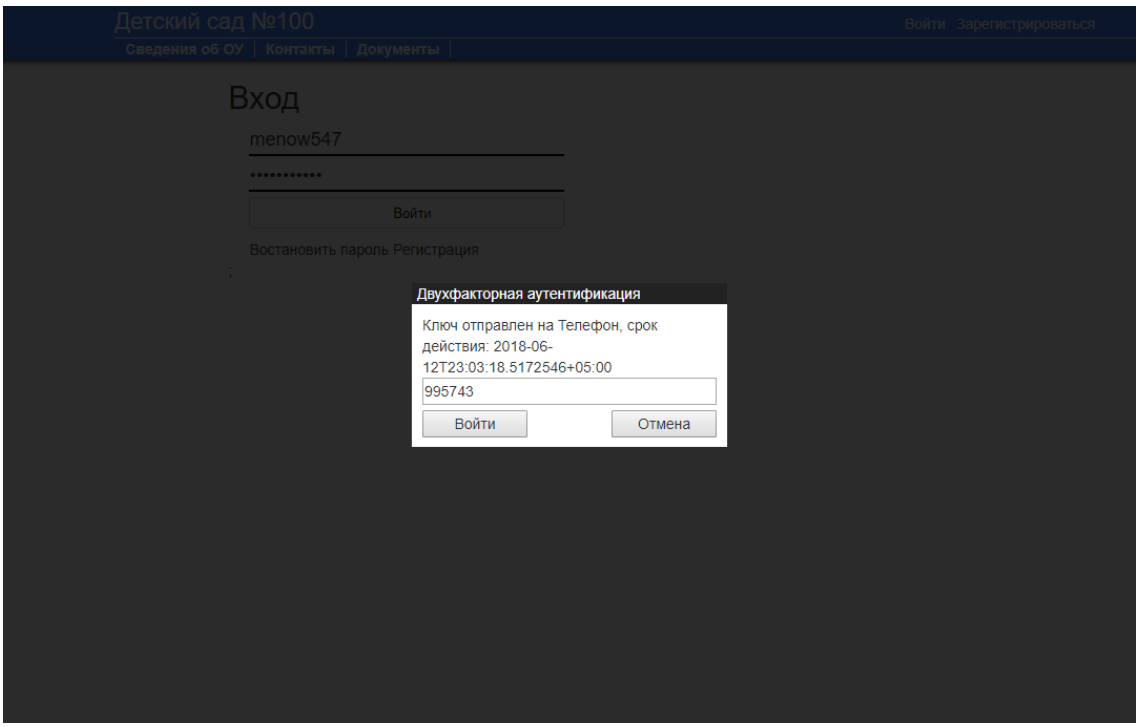


Рисунок 5.6 – Двухфакторная аутентификация

5.4 Личный кабинет пользователя

После аутентификации, у пользователя поменяется главная страница и верхняя часть сайта, появляется доступ в личный кабинет. Вход в него происходит при помощи нажатия на ссылку “Профиль” в верхней части сайта (см. рисунок 5.7).

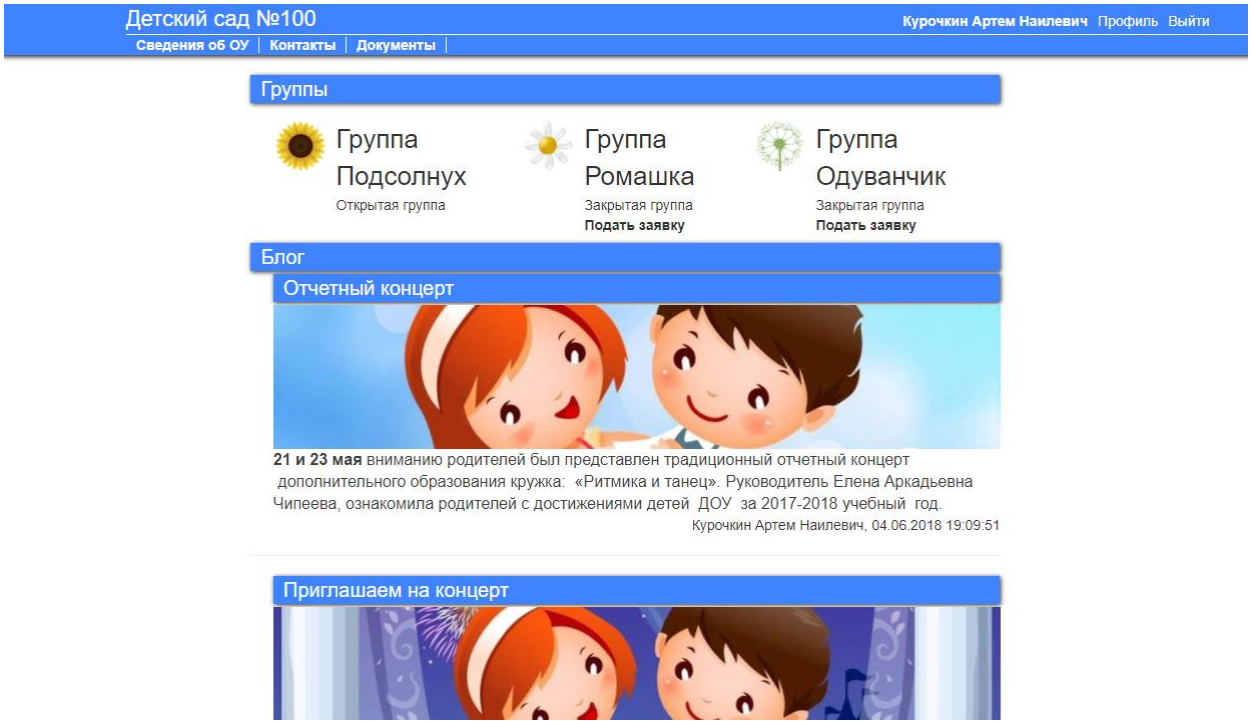


Рисунок 5.7 – Главная страница сайта после аутентификации

В личном кабинете пользователя можно подтвердить, бросить или задать номер телефона и адрес почты телефона в разделах “Email” и “Телефон” соответственно. Также есть возможность изменить аватар, воспользовавшись кнопкой “Выбрать аватар” и “Изменить аватар” в разделе “Аватар”, изменить пароль в разделе “Пароль”, поменять способ аутентификации в разделе “Метод аутентификации”, где включается и выключается двухфакторная аутентификация (см рисунок 5.8).

При сбросе подтвержденных телефона или почты и смене типа аутентификации, необходимо подтвердить эти действие так же, как и при двухфакторной аутентификации (см. рисунок 5.6).

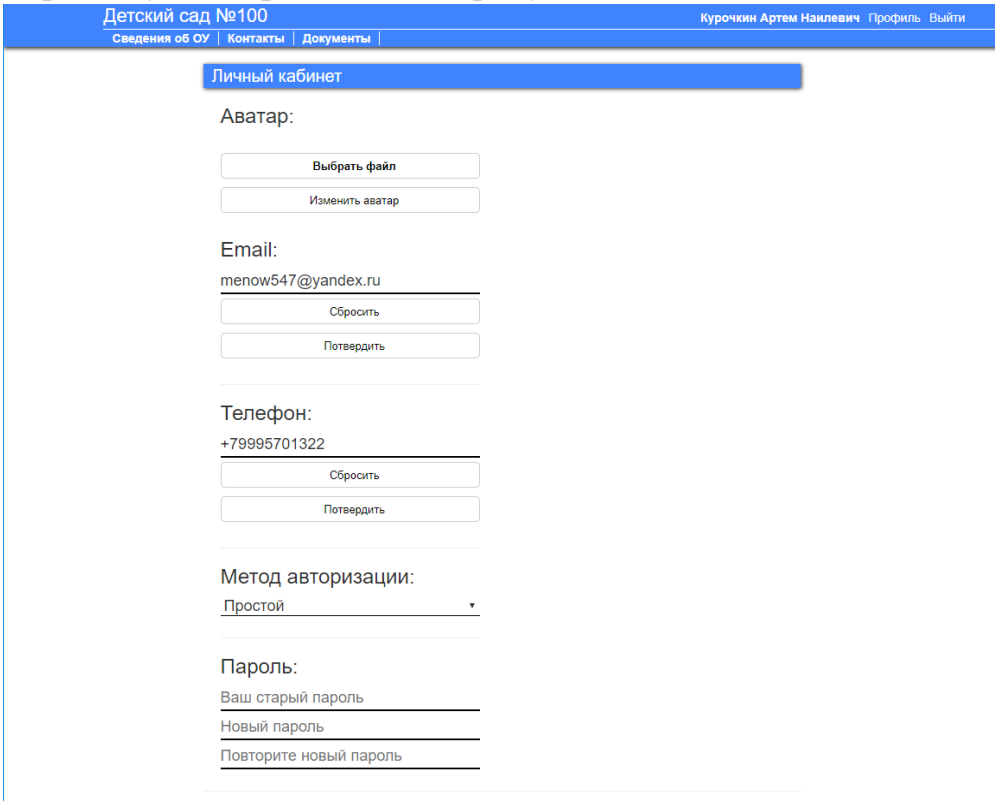


Рисунок 5.8 – Страница личного кабинета пользователя

5.5 Группы

Пользователи могу подавать запросы на вступления в группы при помощи нажатия на ссылку “Подать заявку” под названием нужной группы на главной странице сайта (см. рисунок 5.7).

Внутри групп простые пользователи могу создавать темы в открытых секциях (т.е. в секциях, которые открыты для создания тем). Для этого необходимо нажать на ссылку “Создать тему” напротив названия нужной секции (см. рисунок 5.9). После нажатия на ссылку пользователя переносит на страницу создания новой темы (см. рисунок 5.10). После нажатия на кнопку “Создать”, создается новая тема и пользователь перенаправляется в эту тему (см. рисунок 5.11)

Пользователи так же могут редактировать созданные ими темы. Для этого необходимо нажать на ссылку “Редактировать” в самой теме (см. рисунок 5.11). Страница редактирования темы аналогична странице ее создания.

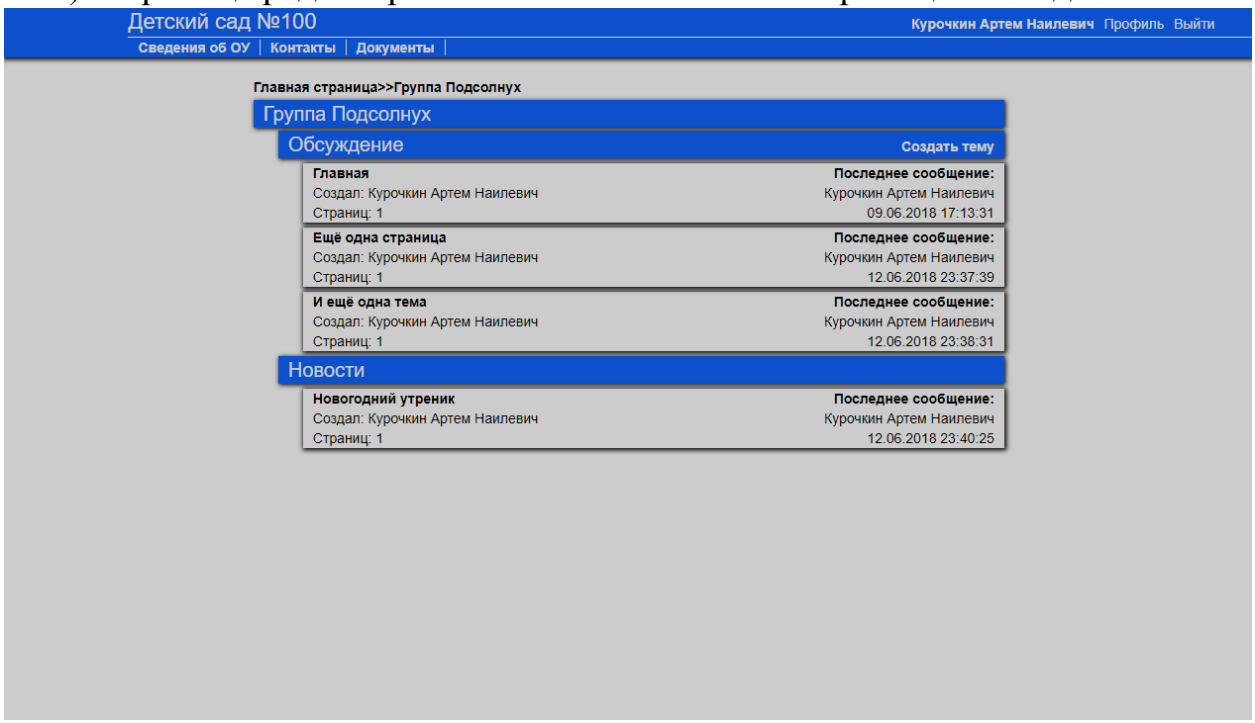


Рисунок 5.9 – Страница группы

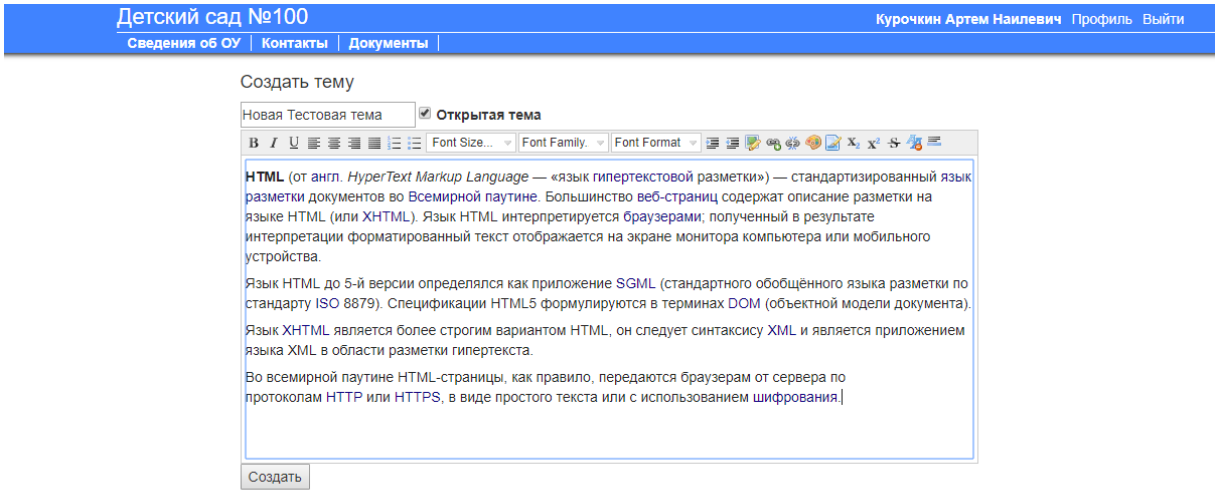


Рисунок 5.10 – Страница создания темы

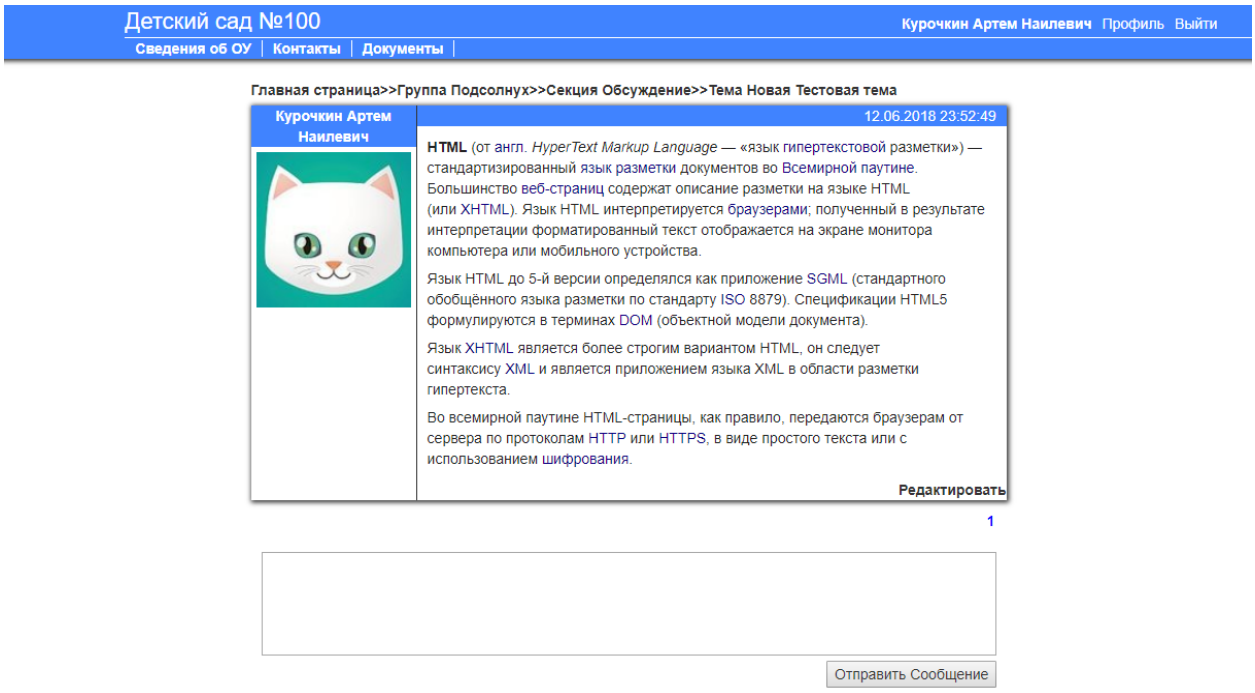


Рисунок 5.11 – Страница темы

Пользователи могут оставлять сообщения в открытых группах (т.е. в группах, в которых разрешено оставлять сообщения). Для этого необходимо ввести нужный текст в поле ввода внизу страницы и нажать кнопку “Отправить сообщение” (см. рисунок 5.12). После этого появится новое сообщение в теме (см. рисунок 5.13).

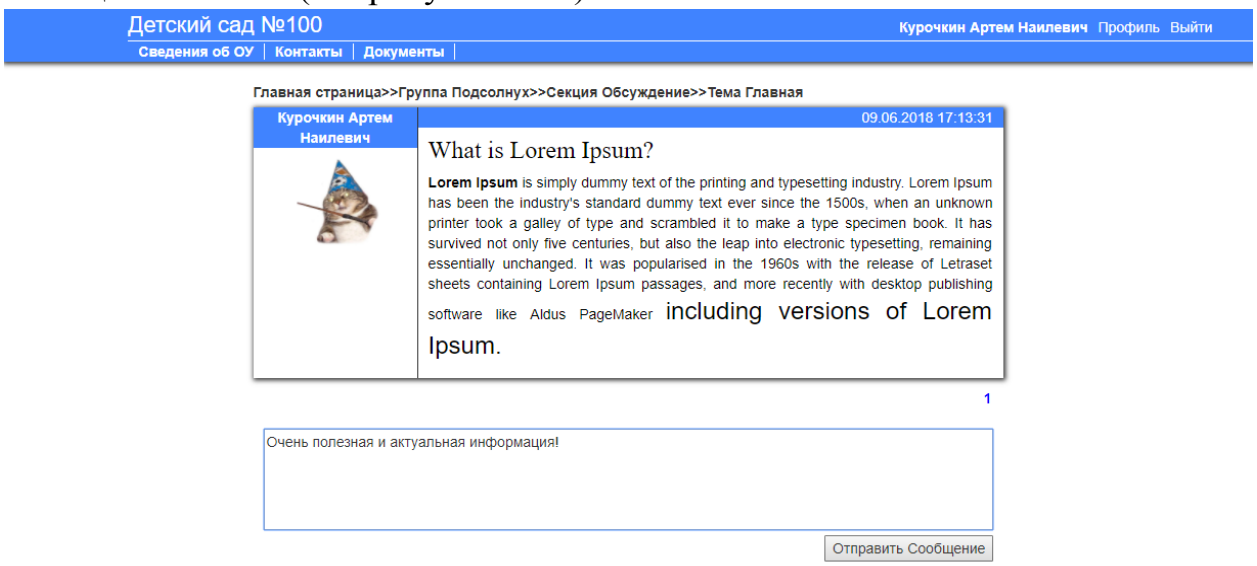


Рисунок 5.12 – Добавление нового сообщения в тему

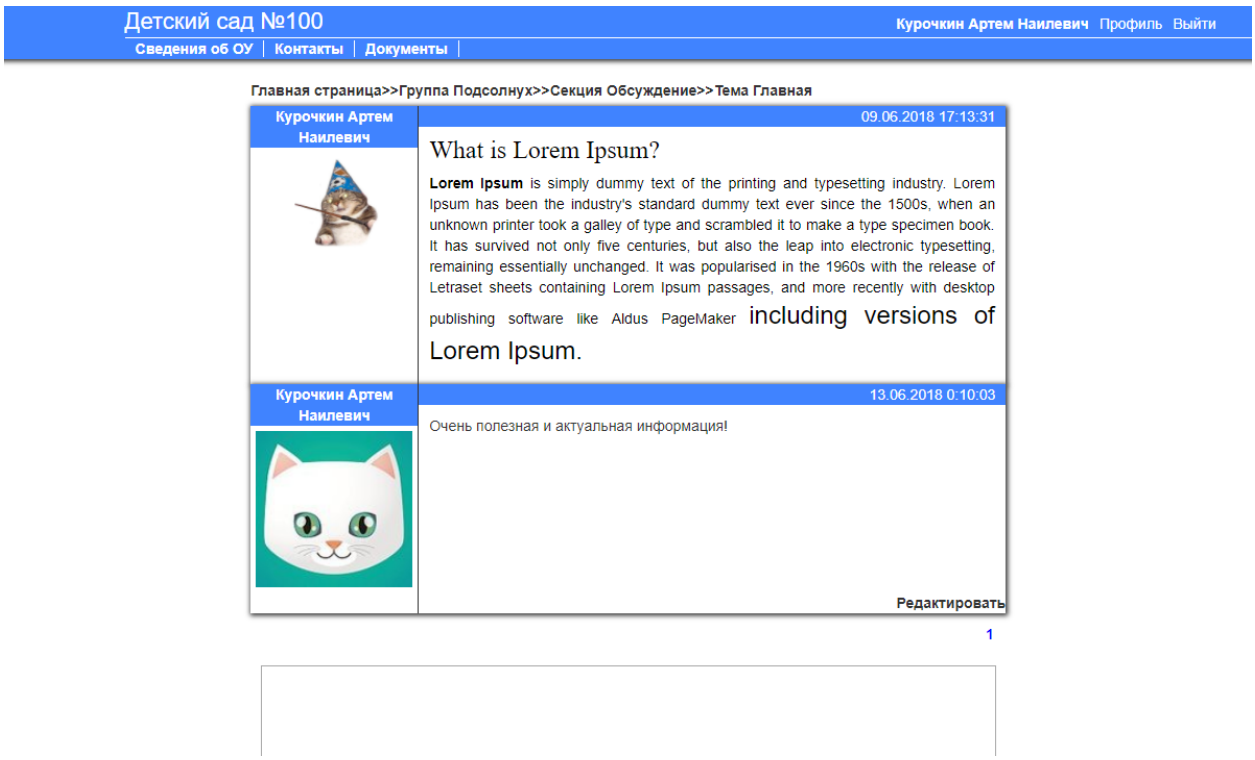


Рисунок 5.13 – Страница темы после добавления сообщения

Администраторы, модераторы и модераторы группы могут создавать новые секции, для этого необходимо нажать на ссылку “Добавить секцию” внутри группы (см. рисунок 5.14).

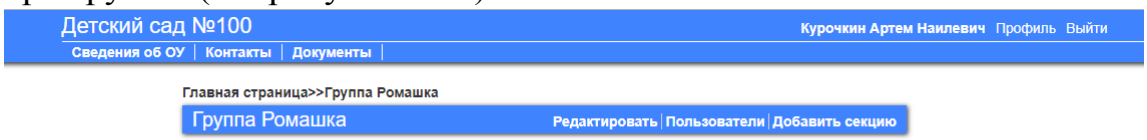


Рисунок 5.14 – Страница группы

Так же они могут управлять ролями пользователей, которые состоят в группе, добавлять и удалять пользователей из группы. Для этого необходимо нажать на ссылку “Пользователи” (см. рисунок 5.14). Страница управления пользователями группы представлены на рисунках 5.15-5.16.

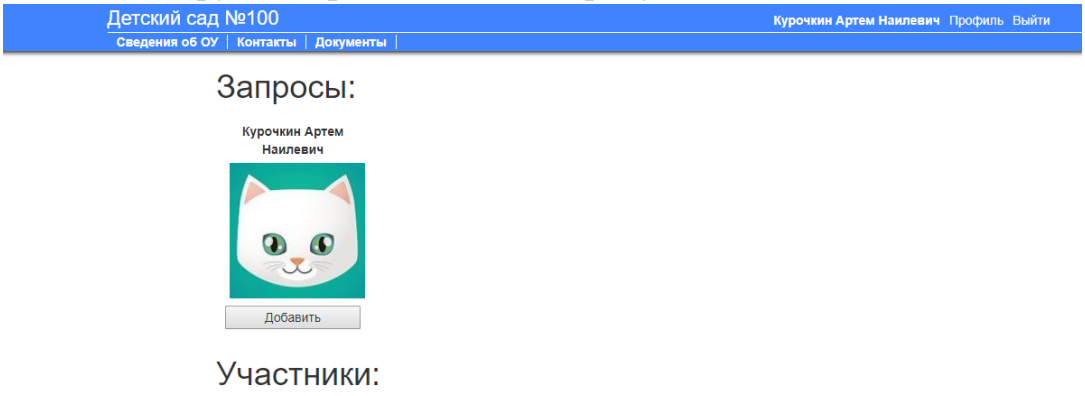


Рисунок 5.15 – Раздел “Пользователи” группы

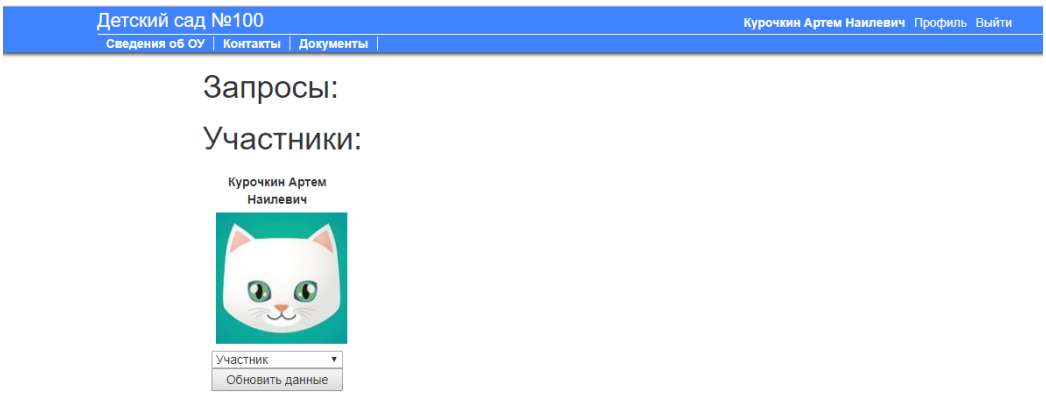


Рисунок 5.16 – Раздел “Пользователи” группы после добавления нового пользователя

Администраторы, модераторы и модераторы группы могут редактировать группы, для этого необходимо на главной странице нажать на ссылку “Редактировать” внизу необходимой группы на главной странице, или внутри самой группы нажать на аналогичную ссылку (см. рисунок 5.14). Администраторы так же могут создавать и удалять группы. Страница создания и редактирования темы идентичны (см. рисунок 5.18).

При нажатии на кнопку “сменить логотип” открывается меню выбора изображений (см. рисунок 5.18).

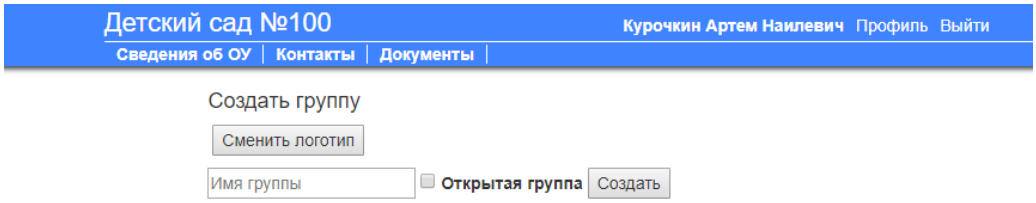


Рисунок 5.17 – Страница создания новой группы

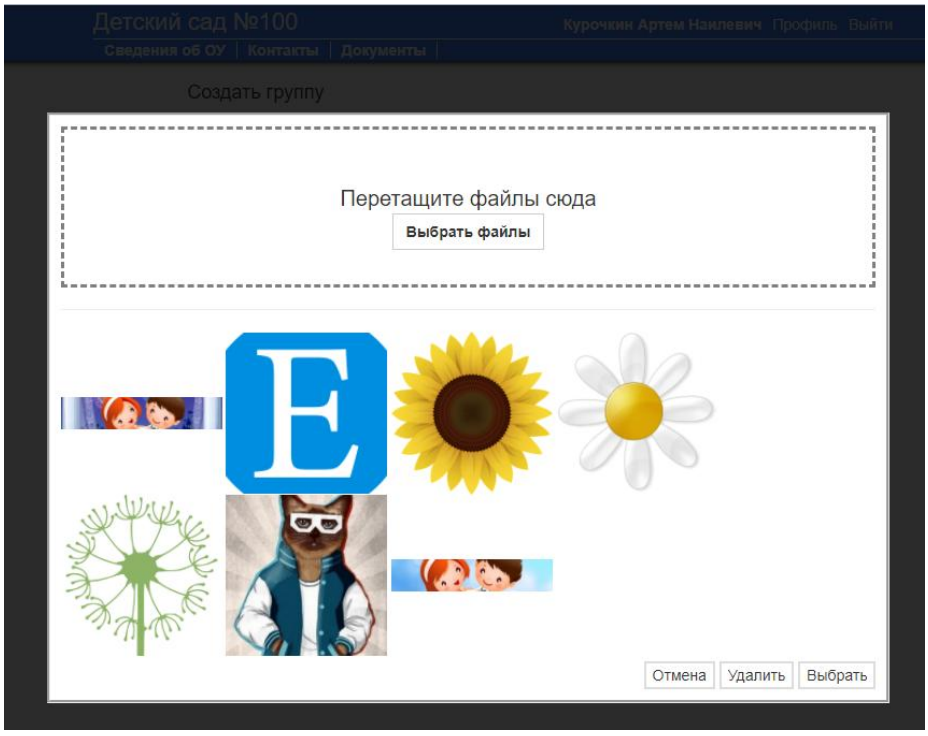


Рисунок 5.18 – Меню выбора изображения

5.6 Статичные страницы

Статичные страницы на сайте выполняют информационную роль, на них располагается информация об учреждении. У каждой статичной страницы могут быть дочерние страницы, переход на которые осуществляется посредством ссылок внизу родительской страницы. Те страницы, которые не имеют родительской, доступны в главном меню в верхней части сайта. Пример информационной страницы представлен на рисунке 5.19.

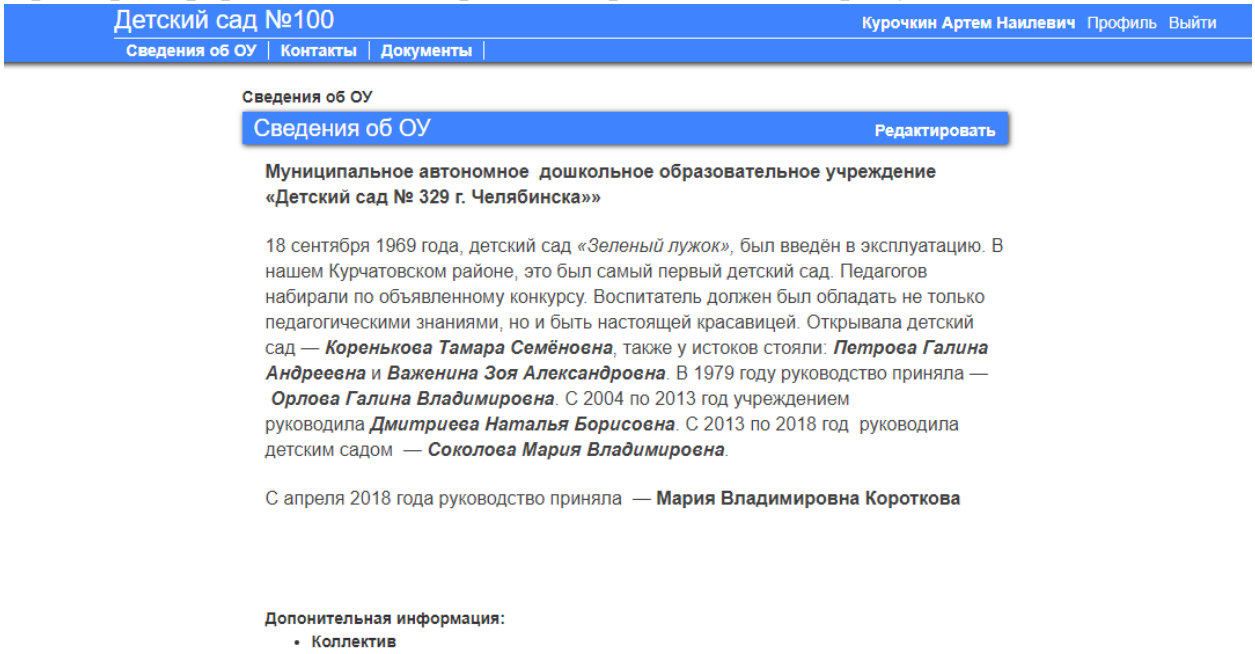


Рисунок 5.19 – Пример статичной страницы

5.7 Блог

Блог служит для того, чтобы в нем располагались новости о жизни учреждения. Блог доступен на главной странице сайте (последние 10 записей) (см. рисунок 5.1), либо на отдельной странице, переход на которую осуществляется посредством нажатия на ссылку “Блог” на главной странице сайта (см. рисунок 5.1).

Создавать записи в блоге могут администраторы и модераторы. Для этого необходимо либо на главной странице, либо на странице блога нажать на ссылку “создать запись”. Страница создания записи представлена на рисунке 5.20.

Страница редактирования записей полностью идентична странице создания.

При создании или редактировании записи можно выбрать картинку-превью, которая будет отображаться выше текста записи (см. рисунок 5.1). Для этого необходимо нажать на кнопку “сменить превью” (см. рисунок

5.20). После этого откроется меню выбора изображения, аналогичное представленному на рисунке 5.18.

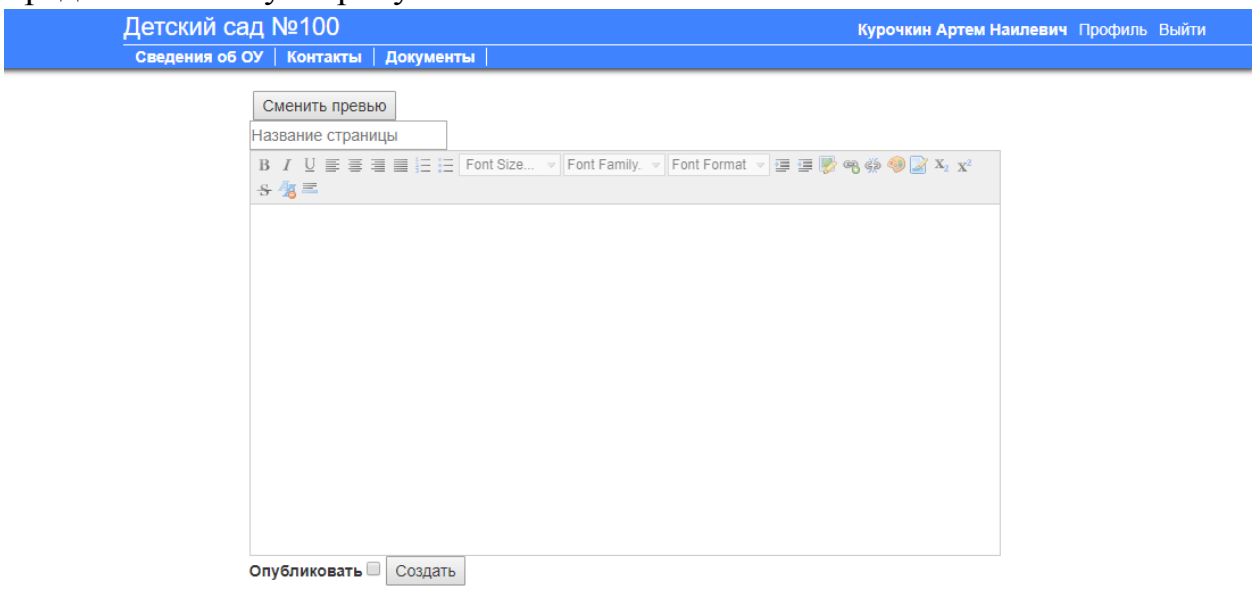


Рисунок 5.20 – Страница создания записей в блог

5.8 Панель администратора

Панель администратора сайта доступна только администраторам сайта. Данная панель разделена на три раздела:

- конфигурация;
- пользователи;
- страницы.

Страница панели администратора представлена на рисунке 5.21.

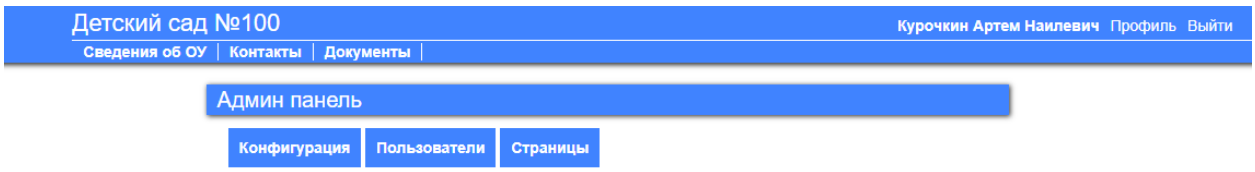


Рисунок 5.21 – Панель администратора

5.8.1 Раздел “Конфигурация сайта”

В разделе “Конфигурация сайта” панели администратора (см. рисунок 5.22) есть возможность настроить: название учреждения, которое выводится в верхней части сайта, конфигурацию подключения к базе данных и логотип сайта.

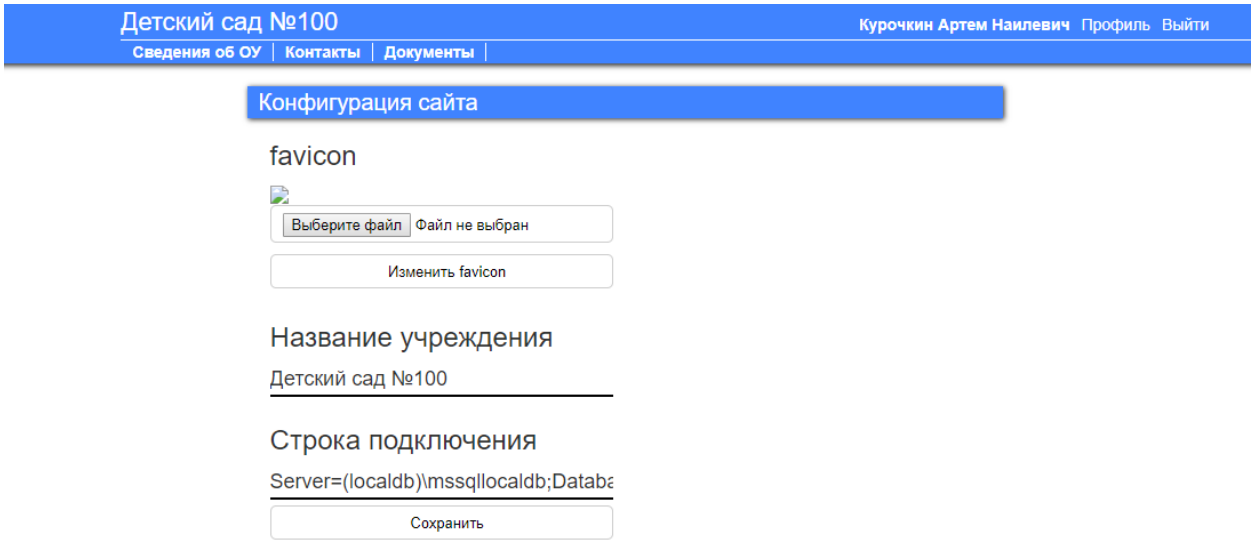


Рисунок 5.22 – Раздел “Конфигурация сайта” панели администратора

5.8.2 Раздел “Пользователи”

В данном разделе редактируются данные пользователей. Для этого в строке поиск пользователя необходимо ввести имя и фамилию пользователя и нажать кнопку “Поиск” (см. рисунок 5.23). В результате на страницу будет выведена информация о пользователях с таким именем и фамилией (см. рисунок 5.23).

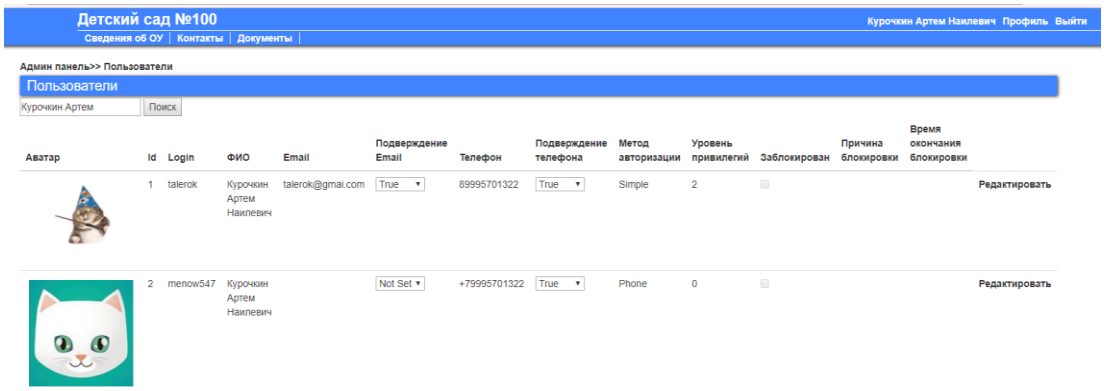


Рисунок 5.23 – Раздел “Пользователи” в панели администратора

Для редактирования данных пользователя необходимо нажать на ссылку “Редактировать” напротив нужного аккаунта (см. рисунок 5.23). После этого администратора перенаправит на страницу редактирования данных пользователя (см. рисунок 5.24). На этой странице так же возможно блокировать и разблокировать пользователей.

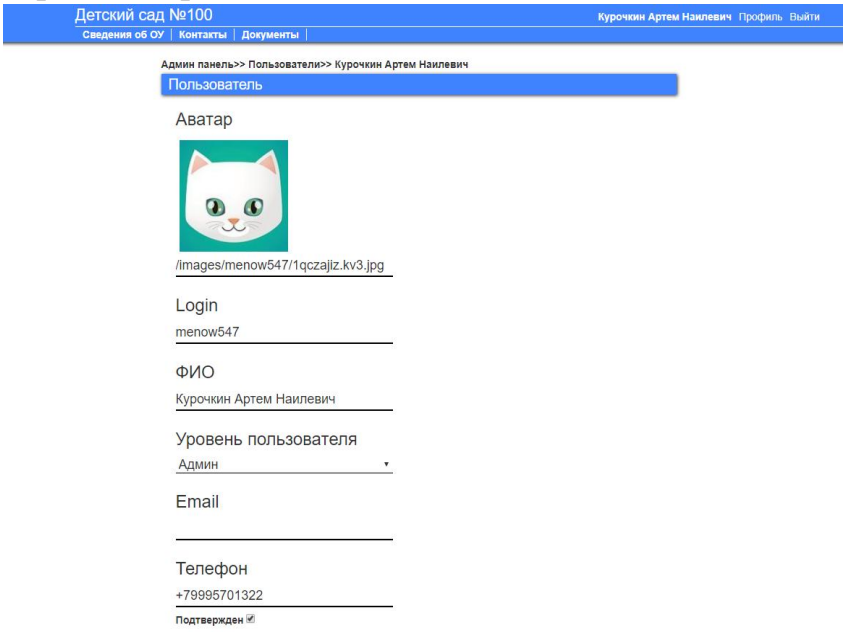


Рисунок 5.24 – Страница редактирования данных пользователя

5.8.3 Раздел “Страницы”

В этом разделе производится добавление, удаление и изменение статичных страниц на сайте. Для создания новой страницы необходимо нажать на ссылку “Создать страницу” (см. рисунок 5.25), которая перенаправляет на страницу создания новой статичной страницы (см. рисунок 5.26).

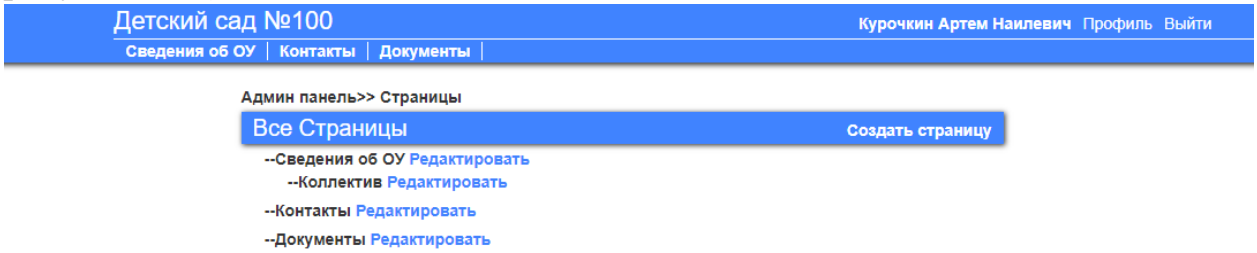


Рисунок 5.25 – Раздел “Страницы” панели администратора

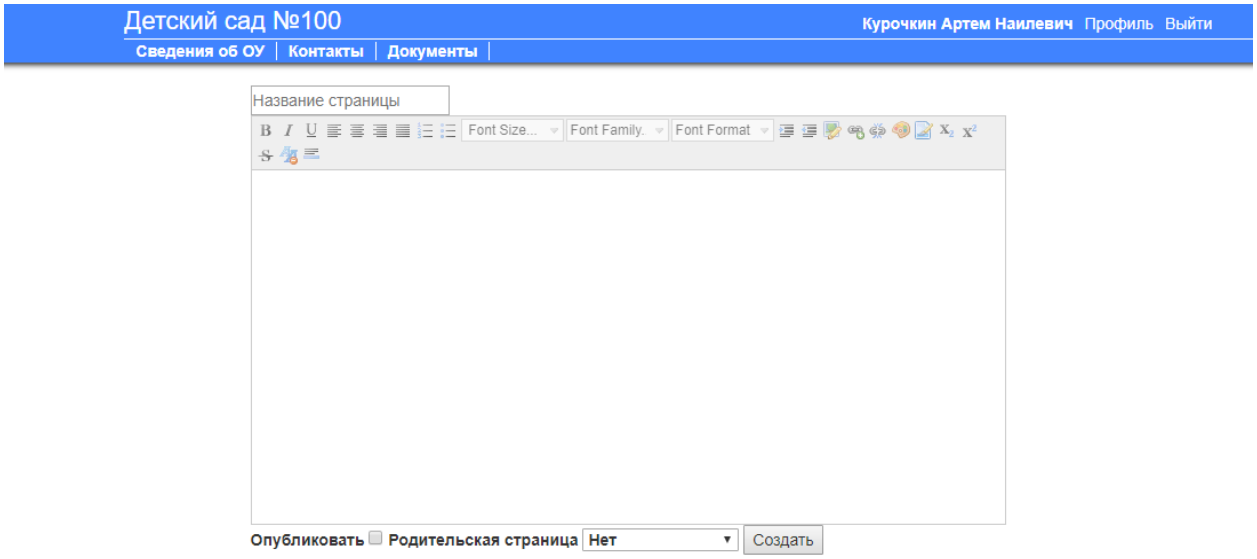


Рисунок 5.26 – Страница создания статичных страниц

При создании или редактировании статичной страницы ей можно указать родительскую страницу. После этого ссылка на данную страницу будет отображаться в нижней части страницы-родителя (см. рисунок 5.19).

У администратора есть возможность редактировать страницы. Для этого необходимо нажать на ссылку “Редактировать” в разделе “Страницы” панели администратора (см. рисунок 5.25), либо на аналогичную ссылку на самой странице (см. рисунок 5.19).

Страница редактирования и создания статичных страниц полностью идентичны.

5.9 Выводы по разделу

В разделе 5 была продемонстрирована работа всех систем разработанного сайта, из чего можно сделать вывод, что весь функционал, который планировался для разработанного сайта – реализован и полностью работоспособен.

ЗАКЛЮЧЕНИЕ

В данной работе были проанализированы требования к веб-сайту, проведен обзор и выбор существующих средств разработки, описаны их достоинства и недостатки.

Разработана архитектура веб-сайта и дизайн его графического интерфейса пользователя. Была выполнена программная реализация.

На последнем этапе была проведена полная проверка работоспособности всех систем сайта.

В результате было разработан сайт для дошкольного образовательного учреждения, отвечающий всем указанным требованиям. Таким образом, все поставленные задачи были успешно выполнены.

В дальнейшем планируется реализовать дополнительный функционал веб-сайта, который включает в себя:

- возможность использования других реляционных СУБД;
- система отправки пользователями личных сообщений;
- аутентификация через сторонние сервисы.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Уильям, Р. Станек Internet Information Services (IIS) 7.0. Справочник администратора / Уильям Р. Станек. - М.: Русская Редакция, БХВ-Петербург, 2012. - 528 с.
2. Сабин-Вильсон, WordPress для чайников / Сабин-Вильсон. - М.: Вильямс, 2010. - 368 с.
3. Туманов, В.Е. Основы проектирования реляционных баз данных / В.Е. Туманов. - М.: Бином, 2011. - 420 с.
4. Кириллов, В.В. Введение в реляционные базы данных (+ CD-ROM) / В.В. Кириллов. - М.: БХВ-Петербург, 2016. - 318 с.
5. ORM или как забыть о проектировании БД [Электронный ресурс] / Режим доступа: https://habr.com/post/237889/, свободный. – Загл. с экрана.
6. Магдануров Г., Юнев В. ASP.NET MVC Framework; БХВ-Петербург - Москва, 2010. - 320 с.
7. Паттерн Unit of Work [Электронный ресурс] / Режим доступа: https://metanit.com/sharp/mvc5/23.3.php, свободный. – Загл. с экрана.
8. Диаграммы вариантов использования [Электронный ресурс] / Режим доступа: https://www.intuit.ru/studies/courses/32/32/lecture/1004, свободный. – Загл. с экрана.
9. Диаграммы размещения [Электронный ресурс] / Режим доступа: https://studfiles.net/preview/2806643/page:6/, свободный. – Загл. с экрана.
10. Нейгард, Release it! Проектирование и дизайн ПО для тех, кому не всё равно / Нейгард. - М.: Питер, 2014. - 625 с
11. Фримен, Изучаем HTML, XHTML и CSS / Фримен. - М.: Питер, 2013. - 720 с.
12. Квинт, Создаем сайты с помощью HTML, XHTML и CSS / Квинт. - М.: Питер, 2014. - 448 с.
13. Евсеев, Д. А. Web-дизайн в примерах и задачах / Д.А. Евсеев, В.В. Трофимов. - М.: КноРус, 2015. - 272 с.
14. Material Design: на Луну и обратно [Электронный ресурс] / Режим доступа: https://habr.com/company/redmadrobot/blog/252773/, свободный. – Загл. с экрана.
15. Гамма, Приемы объектно-ориентированного проектирования. Паттерны проектирования / Гамма - Москва: СИНТЕГ, 2016. - 366 с.
16. Зиглер, Методы проектирования программных систем / Зиглер. - М.: Мир, 2015. - 328 с.

ПРИЛОЖЕНИЕ 1 ОПИСАНИЕ ПРОГРАММЫ

1.Общие сведения

Веб-сайт для дошкольного образовательного учреждения. Для работы программы необходимы: сервер IIS Server 7.0, СУБД MSSQL Server. Визуальная структура сайта была реализована с помощью HTML, CSS, JavaScript и фреймворка jQuery. Серверная часть была написана на ASP .NET Core 2.1.

2. Функциональное назначение

Данная система предназначена для использования в качестве сайта для любого дошкольного образовательного учреждения.

3. Используемые технические средства

Для работы системы необходим компьютер с объемом оперативной памяти не менее 1Гб оперативной памяти, тактовой частотой процессора не менее 2 ГГц, подключение к интернету с пропускной способностью не менее 10 Мбит/сек.

4. Вызов и загрузка

Для доступа к системе используется веб браузер, пользователю необходимо ввести адрес сайта.

5. Входные и выходные данные

На входе системе подаются адреса страниц и данные, которые пользователь вводит в полях форм, а на выходе веб страницы.

ПРИЛОЖЕНИЕ 2 ТЕКСТ ПРОГРАММЫ

Файл Chat.cs

```
using System;
using System.Collections.Generic;
using System.Text;

namespace Education.DAL.Entities.Chat
{
    class Chat
    {
        public User Owner { get; set; }
        public ICollection UserChat { get; set; }
        public ICollection ChatMessage { get; set; }
    }
}
```

Файл ChatMessage.cs

```
using System;
using System.Collections.Generic;
using System.Text;

namespace Education.DAL.Entities.Chat
{
    class ChatMessage
    {
        public User Owner { get; set; }
        public string Text { get; set; }
    }
}
```

Файл UserChat.cs

```
using System;
using System.Collections.Generic;
using System.Text;

namespace Education.DAL.Entities.Chat
{
    class UserChat
    {
    }
}
```

Файл Group.cs

```
using System;
using System.Collections.Generic;
using System.Text;

namespace Education.DAL.Entities
{
    public class Group : Entity
    {
        public string Name { get; set; }
        public string Logo { get; set; }
        public bool Open { get; set; }
        public virtual ICollection Users { get; set; }
        public virtual ICollection Sections { get; set; }
        public Group()
        {
            Users = new List();
            Sections = new List();
        }
```

```
    }
}
```

Файл Message.cs

```
using System;
using System.Collections.Generic;
using System.Text;

namespace Education.DAL.Entities
{
    public class Message : Entity
    {
        public virtual User Owner { get; set; }
        public string Text { get; set; }
        public DateTime Time { get; set; }
        public virtual User LastEditor { get; set; }
        public virtual DateTime LastEditTime { get; set; }
        public virtual Theme Theme { get; set; }
    }
}
```

Файл Section.cs

```
using System;
using System.Collections.Generic;
using System.Text;

namespace Education.DAL.Entities
{
    public class Section : Entity
    {
        public string Name { get; set; }
        public virtual ICollection Themes { get; set; }
        public virtual Group Group { get; set; }
        public bool Open { get; set; }
        public Section()
        {
            Themes = new List();
        }
    }
}
```

Файл Theme.cs

```
using System;
using System.Collections.Generic;
using System.Text;

namespace Education.DAL.Entities
{
    public class Theme : Entity
    {
        public string Name { get; set; }
        public virtual ICollection Messages { get; set; }
        public bool Open { get; set; }
        public virtual Section Section { get; set; }
        public Theme(){
            Messages = new List();
        }
    }
}
```

Файл UserGroup.cs

```
using System;
using System.Collections.Generic;
```

```
using System.Text;

namespace Education.DAL.Entities
{
    public enum UserGroupStatus
    {
        request,
        member,
        owner
    }
    public class UserGroup
    {
        public int GroupId { get; set; }
        public virtual Group Group { get; set; }
        public int UserId { get; set; }
        public virtual User User { get; set; }
        public UserGroupStatus Status { get; set; }
    }
}
```

Файл Note.cs

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text;

namespace Education.DAL.Entities.Pages
{
    public class Note : Entity
    {
        public virtual User Owner { get; set; }
        public string Preview { get; set; }
        public DateTime Time { get; set; }
        public string Name { get; set; }
        public string Text { get; set; }
        public bool Published { get; set; }
    }
}
```

Файл Page.cs

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text;

namespace Education.DAL.Entities.Pages
{
    public class Page : Entity
    {
        public string Name { get; set; }
        public string Text { get; set; }
        public bool Published { get; set; }
        public virtual ICollection ChildPages { get; set; }
        public virtual Page ParentPage { get; set; }
    }
}
```

Файл Ban.cs

```
using System;
using System.Collections.Generic;
using System.IO;
```

```
using System.Linq;
using System.Text;

namespace Education.DAL.Entities
{
    public class Ban : Entity
    {
        public string Reason { get; set; }
        public DateTime EndTime { get; set; }
    }
}
```

Файл Contact.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
using System.Text;

namespace Education.DAL.Entities
{
    public class Contact : Entity
    {
        public string Value { get; set; }
        public bool Confirmed { get; set; }
        public virtual Key Key { get; set; }
        public virtual Key ConfirmKey { get; set; }
    }
}
```

Файл Key.cs

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text;

namespace Education.DAL.Entities
{
    public class Key : Entity
    {
        public string Value { get; set; }
        public DateTime EndTime { get; set; }
    }
}
```

Файл User.cs

```
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;

namespace Education.DAL.Entities
{
    public enum AuthType
    {
        Simple,
        Email,
        Phone
    }

    public class User : Entity
    {
        public string Login { get; set; }
        public virtual Contact Email { get; set; }
```

```
        public virtual Contact Phone { get; set; }
        public string Password { get; set; }
        public virtual UserInfo Info { get; set; }
        public short Level { get; set; }
        public AuthType authType { get; set; }
        public virtual Ban Ban { get; set;}
        public virtual ICollection Groups { get; set; }
        public User() { }
    }
}
```

Файл UserClaim.cs

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text;

namespace Education.DAL.Entities
{
    public class UserClaim : Entity
    {
        public virtual User User { get; set; }
        public string Value { get; set; }
        public DateTime LoginTime { get; set; }
        public string LoginBrowser { get; set; }
        public string LoginIp { get; set; }
    }
}
```

Файл UserInfo.cs

```
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;

namespace Education.DAL.Entities
{
    public class UserInfo : Entity
    {
        public string FullName { get; set; }
        public string Avatar { get; set; }
    }
}
```

Файл Entity.cs

```
using System;
using System.Collections.Generic;
using System.Text;

namespace Education.DAL.Entities
{
    public class Entity
    {
        public int Id { get; set; }
    }
}
```

Файл IConfVal.cs

```
using Education.DAL.Entities;
using System;
using System.Collections.Generic;
using System.Text;

namespace Education.DAL.Interfaces
{
```

```
    public interface IConfVal
    {
        T Value { get; }
        bool Confirmed { get; }
    }
}
```

Файл IRepos.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace Education.DAL.Interfaces
{
    public interface IRepos where TEntity : class
    {
        IQueryable Get();
        void Add(TEntity item);
        void Add(IEnumerable items);
        void Delete(TEntity item);
        void Delete(IEnumerable items);
        void Edited(TEntity item);
        void Edited(IEnumerable items);
    }

}
```

Файл IUOW.cs

```
using System;
using System.Collections.Generic;
using System.Text;
using Education.DAL.Entities;
using Education.DAL.Entities.Pages;

namespace Education.DAL.Interfaces
{
    public interface IUOW : IDisposable
    {
        IRepos UserRepository { get; }
        IRepos UserInfoRepository { get; }
        IRepos ContactRepository { get; }
        IRepos AuthKeyRepository { get; }
        IRepos BanRepository { get; }
        IRepos UserClaimRepository { get; }
        //------------Forums
        IRepos MessageRepository { get; }
        IRepos ThemeRepository { get; }
        IRepos SectionRepository { get; }
        IRepos GroupRepository { get; }
        IRepos UserGroupRepository { get; }
        //------------Pages
        IRepos NoteRepository { get; }
        IRepos PageRepository { get; }
        //------------------
        void SaveChanges();
    }
}
```

Файл IUOWFactory.cs

```
using System;
using System.Collections.Generic;
using System.Text;
```

```
namespace Education.DAL.Interfaces
{
    public interface IUOWFactory
    {
        IUOW Get();
    }
}
```

Файл EFContext.cs

```
using Education.DAL.Entities;
using Education.DAL.Entities.Pages;
using Microsoft.EntityFrameworkCore;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace Education.DAL.Repositories
{
    class EFContext : DbContext
    {
        public EFContext(DbContextOptions options) : base(options)
        {
            Database.EnsureCreated();
        }

        public override DbSet Set()
        {
            var a = typeof(TEntity);
            return base.Set();
        }

        protected override void OnModelCreating(ModelBuilder modelBuilder)
        {
            modelBuilder.Entity().ToTable("Users");
            modelBuilder.Entity().ToTable("Claims");
            modelBuilder.Entity().ToTable("Groups");
            modelBuilder.Entity().ToTable("Sections");
            modelBuilder.Entity().ToTable("Themes");
            modelBuilder.Entity().ToTable("Notes");
            //------------------------------------------------
            modelBuilder.Entity().HasKey(x => new { x.UserId, x.GroupId });
            modelBuilder.Entity()
                .HasOne(x => x.Group).WithMany(x => x.Users).HasForeignKey(x
=> x.GroupId).OnDelete(DeleteBehavior.Cascade);
            modelBuilder.Entity()
                .HasOne(x => x.User).WithMany(x => x.Groups).HasForeignKey(x
=> x.UserId).OnDelete(DeleteBehavior.Cascade);
            //------------------------------------------------
            modelBuilder.Entity().HasOne(x => x.Theme).WithMany(x =>
x.Messages).OnDelete(DeleteBehavior.Cascade);
            modelBuilder.Entity().HasOne(x => x.Section).WithMany(x =>
x.Themes).OnDelete(DeleteBehavior.Cascade);
            modelBuilder.Entity().HasOne(x => x.Group).WithMany(x =>
x.Sections).OnDelete(DeleteBehavior.Cascade);
            //------------------------------------------------
            modelBuilder.Entity().HasOne(x => x.ParentPage).WithMany(x =>
x.ChildPages).OnDelete(DeleteBehavior.ClientSetNull);

        }
```

```
    }
}
```

Файл EFRepos.cs

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Text;
using System.Linq;
using Microsoft.EntityFrameworkCore;
using Education.DAL.Entities;

namespace Education.DAL.Repositories
{
    class EFRepos : Interfaces.IRepos where TEntity : class
    {
        protected DbSet dbSet;
        protected DbContext dbContext;

        public EFRepos(DbContext context)
        {
            dbContext = context;

            dbSet = dbContext.Set();

        }

        public void Add(TEntity item)
        {
            dbSet.Add(item);
        }

        public void Add(IEnumerable items)
        {
            dbSet.AddRange(items);
        }

        public void Delete(TEntity item)
        {
            dbContext.Entry(item).State = EntityState.Deleted;
        }

        public void Delete(IEnumerable items)
        {
            dbSet.RemoveRange(items);
        }

        public void Edited(TEntity item)
        {
            dbContext.Entry(item).State = EntityState.Modified;

        }

        public void Edited(IEnumerable items)
        {
            foreach (var item in items)
            {
                dbContext.Entry(item).State = EntityState.Modified;
            }
        }

        public IQueryable Get()
        {
```

```
            return dbSet;
        }
    }
}
```

Файл EFUOW.cs

```
using Microsoft.EntityFrameworkCore;
using Education.DAL.Entities;
using Education.DAL.Interfaces;
using Microsoft.EntityFrameworkCore.Proxies;
using System;
using Education.DAL.Entities.Pages;

namespace Education.DAL.Repositories
{
    public class EFUOW : IUOW
    {
        private EFContext dbContext;
        //-------------------------------
        public EFUOW(string connSrting)
        {
            var optionsBuilder = new DbContextOptionsBuilder();
            var options = optionsBuilder
                .UseLazyLoadingProxies()
                .UseSqlServer(connSrting)
            .Options;
            dbContext = new EFContext(options);
        }

        private IRepos InitRepos() where T : class
        {
            return new EFRepos(dbContext);
        }

        public void SaveChanges()
        {
            dbContext.SaveChanges();
        }

        public void Dispose()
        {
            dbContext.Dispose();
            GC.SuppressFinalize(this);
        }

        #region Repos
        //-------------------
        private IRepos UserInfoRepository_p;
        private IRepos UserRepository_p;
        private IRepos ContactRepository_p;
        private IRepos AuthKeyRepository_p;
        private IRepos BanRepository_p;
        private IRepos UserClaimRepository_p;
        private IRepos MessageRepository_p;
        private IRepos ThemeRepository_p;
        private IRepos SectionRepository_p;
        private IRepos GroupRepository_p;
        private IRepos UserGroupRepository_p;
        private IRepos NoteRepository_p;
        private IRepos PageRepository_p;

        public IRepos UserInfoRepository
        {
```

```
    get
    {
        if (UserInfoRepository_p == null)
            UserInfoRepository_p = InitRepos();
        return UserInfoRepository_p;
    }
}

public IRepos UserRepository
{
    get
    {
        if (UserRepository_p == null)
            UserRepository_p = InitRepos();
        return UserRepository_p;
    }
}

public IRepos ContactRepository
{
    get
    {
        if (ContactRepository_p == null)
            ContactRepository_p = InitRepos();
        return ContactRepository_p;
    }
}

public IRepos AuthKeyRepository
{
    get
    {
        if (AuthKeyRepository_p == null)
            AuthKeyRepository_p = InitRepos();
        return AuthKeyRepository_p;
    }
}

public IRepos BanRepository
{
    get
    {
        if (BanRepository_p == null)
            BanRepository_p = InitRepos();
        return BanRepository_p;
    }
}

public IRepos UserClaimRepository
{
    get
    {
        if (UserClaimRepository_p == null)
            UserClaimRepository_p = InitRepos();
        return UserClaimRepository_p;
    }
}

public IRepos MessageRepository
{
    get
    {
        if (MessageRepository_p == null)
```

```
            MessageRepository_p = InitRepos();
        return MessageRepository_p;
    }
}

public IRepos ThemeRepository
{
    get
    {
        if (ThemeRepository_p == null)
            ThemeRepository_p = InitRepos();
        return ThemeRepository_p;
    }
}

public IRepos SectionRepository
{
    get
    {
        if (SectionRepository_p == null)
            SectionRepository_p = InitRepos();
        return SectionRepository_p;
    }
}

public IRepos GroupRepository
{
    get
    {
        if (GroupRepository_p == null)
            GroupRepository_p = InitRepos();
        return GroupRepository_p;
    }
}

public IRepos UserGroupRepository
{
    get
    {
        if (UserGroupRepository_p == null)
            UserGroupRepository_p = InitRepos();
        return UserGroupRepository_p;
    }
}
public IRepos NoteRepository
{
    get
    {
        if (NoteRepository_p == null)
            NoteRepository_p = InitRepos();
        return NoteRepository_p;
    }
}
public IRepos PageRepository
{
    get
    {
        if (PageRepository_p == null)
            PageRepository_p = InitRepos();
        return PageRepository_p;
    }
}
#endregion
```

```
    }
}
```

Файл EFUOWFactory.cs

```
using Education.DAL.Interfaces;

namespace Education.DAL.Repositories
{
    public class EFUOWFactory : IUOWFactory
    {
        private string cString;
        public EFUOWFactory(string connString)
        {
            cString = connString;
        }

        public IUOW Get()
        {
            return new EFUOW(cString);
        }
    }
}
```

Файл GroupEditDTO.cs

```
using System;
using System.Collections.Generic;
using System.Text;

namespace Education.BLL.DTO.Forum.Edit
{
    public class GroupEditDTO
    {
        public int Id { get; set; }
        public string Name { get; set; }
        public string Logo { get; set; }
        public bool Open { get; set; }
    }
}
```

Файл MessageEdItDTO.cs

```
using System;
using System.Collections.Generic;
using System.Text;

namespace Education.BLL.DTO.Forum.Edit
{
    public class MessageEditDTO
    {
        public int Id { get; set; }
        public int ThemeId { get; set; }
        public string Text { get; set; }
    }
}
```

Файл SectionEditDTO.cs

```
using System;
using System.Collections.Generic;
using System.Text;

namespace Education.BLL.DTO.Forum.Edit
{
    public class SectionEditDTO
    {
```

```
        public int Id { get; set; }
        public int GroupId { get; set; }
        public string Name { get; set; }
        public bool Open { get; set; }
    }
}
```

Файл ThemeEditDTO.cs

```
using System;
using System.Collections.Generic;
using System.Text;

namespace Education.BLL.DTO.Forum.Edit
{
    public class ThemeEditDTO
    {
        public int Id { get; set; }
        public int SectionId { get; set; }
        public string Name { get; set; }
        public bool Open { get; set; }
        public string Text { get; set; }
    }
}
```

Файл CreateResultDTO.cs

```
using System;
using System.Collections.Generic;
using System.Text;

namespace Education.BLL.DTO.Forum
{
    public enum AccessCode
    {
        Succsess,
        NoPremision,
        NotFound,
        Error
    }

    public class CreateResultDTO
    {
        public static CreateResultDTO NoPremision
        {
            get
            {
                return new CreateResultDTO(-1, AccessCode.NoPremision);
            }
        }

        public static CreateResultDTO NotFound
        {
            get
            {
                return new CreateResultDTO(-2, AccessCode.NotFound);
            }
        }

        public static CreateResultDTO Error
        {
            get
            {
                return new CreateResultDTO(-3, AccessCode.Error);
            }
```

```
        }

        public CreateResultDTO(int id, AccessCode code)
        {
            Id = id;
            Code = code;
        }
        public int Id { get; set; }
        public AccessCode Code { get; set; }
    }
}
```

Файл ForumDTO.cs

```
using System;
using System.Collections.Generic;
using System.Text;

namespace Education.BLL.DTO.Forum
{
    public class ForumDTO
    {
        public bool CanCreateGroup { get; set; }
        public IEnumerable Groups { get; set; }
    }
}
```

Файл GroupAccessDTO.cs

```
using System;
using System.Collections.Generic;
using System.Text;

namespace Education.BLL.DTO.Forum
{
    public class GroupAccessDTO : FullAccessDTO
    {
        public bool CanControlUsers { get; set; }
    }
}
```

Файл GroupDTO.cs

```
using System;
using System.Collections.Generic;
using System.Text;

namespace Education.BLL.DTO.Forum
{
    public class GroupDTO
    {
        public int Id { get; set; }
        public string Name { get; set; }
        public string Logo { get; set; }
        public bool Open { get; set; }
        public GroupAccessDTO Access { get; set; }
        public IEnumerable Sections { get; set; }
    }
}
```

Файл GroupPreviewDTO.cs

```
using Education.DAL.Entities;
using System;
using System.Collections.Generic;
using System.Text;
```

```
namespace Education.BLL.DTO.Forum
{
    public class UserGroupInfo
    {
        public bool Member { get; set; }
        public UserGroupStatus Status { get; set; }
    }

    public class GroupPreviewDTO
    {
        public int Id { get; set; }
        public string Name { get; set; }
        public string Logo { get; set; }
        public bool Open { get; set; }
        public GroupAccessDTO Access { get; set; }
        public UserGroupInfo Status { get; set; }
    }
}
```

Файл MessageDTO.cs

```
using System;
using System.Collections.Generic;
using System.Text;

namespace Education.BLL.DTO.Forum
{
    public class MessageRoute
    {
        public int ThemeId { get; set; }
        public string ThemeName { get; set; }
        public ThemeRoute ThemeRoute { get; set; }
    }


    public class MessageDTO
    {
        public int Id { get; set; }
        public UserPublicInfoDTO Owner { get; set; }
        public string Text { get; set; }
        public DateTime Time { get; set; }
        public UserPublicInfoDTO LastEditor { get; set; }
        public DateTime LastEditTime { get; set; }
        public AccessDTO Access { get; set; }
        public MessageRoute Route { get; set; }
    }
}
```

Файл MessagePreviewDTO.cs

```
using System;
using System.Collections.Generic;
using System.Text;

namespace Education.BLL.DTO.Forum
{
    public class MessagePreviewDTO
    {
        public int ThemeId { get; set; }
        public int Page { get; set; }
    }
}
```

Файл SectionDTO.cs

```
using System;
```

```
using System.Collections.Generic;
using System.Text;

namespace Education.BLL.DTO.Forum
{
    public class SectionRoute
    {
        public int GroupId { get; set; }
        public string GroupName { get; set; }
    }

    public class SectionDTO
    {
        public int Id { get; set; }
        public bool Open { get; set; }
        public string Name { get; set; }
        public FullAccessDTO Access { get; set; }
        public IEnumerable Themes { get; set; }
        public SectionRoute Route { get; set; }
    }
}
```

Файл ThemeDTO.cs

```
using System;
using System.Collections.Generic;
using System.Text;

namespace Education.BLL.DTO.Forum
{
    public class ThemeRoute
    {
        public int SectionId { get; set; }
        public string SectionName { get; set; }
        public SectionRoute SectionRoute { get; set; }
    }

    public class ThemeDTO
    {
        public int Id { get; set; }
        public bool Open { get; set; }
        public string Name { get; set; }
        public FullAccessDTO Access { get; set; }
        public int Pages { get; set; }
        public int CurPage { get; set; }
        public IEnumerable Messages { get; set; }
        public ThemeRoute Route { get; set; }
    }
}
```

Файл ThemePreviewDTO.cs

```
using System;
using System.Collections.Generic;
using System.Text;

namespace Education.BLL.DTO.Forum
{
    public class ThemePreviewDTO
    {
        public int Id { get; set; }
        public string Name { get; set; }
        public int Pages { get; set; }
        public UserPublicInfoDTO Owner { get; set; }
        public MessageDTO LastMessages { get; set; }
```

```
    }
}
```

Файл UserGroupDTO.cs

```
using Education.DAL.Entities;
using System;
using System.Collections.Generic;
using System.Text;

namespace Education.BLL.DTO.Forum
{
    public class UserGroupDTO
    {
        public UserPublicInfoDTO UserInfo { get; set; }
        public UserGroupStatus Status { get; set; }
    }
}
```

Файл UserGroupInfoDTO.cs

```
using System;
using System.Collections.Generic;
using System.Text;

namespace Education.BLL.DTO.Forum
{
    public class UserGroupInfoDTO
    {
        public static UserGroupInfoDTO NoPremision
        {
            get
            {
                return new UserGroupInfoDTO { Code = AccessCode.NoPremision};
            }
        }

        public static UserGroupInfoDTO NotFound
        {
            get
            {
                return new UserGroupInfoDTO { Code = AccessCode.NotFound };
            }
        }

        public static UserGroupInfoDTO Error
        {
            get
            {
                return new UserGroupInfoDTO { Code = AccessCode.Error };
            }
        }

        public AccessCode Code { get; set; }
        public IEnumerable Data { get; set; }
    }
}
```

Файл BlogDTO.cs

```
using System;
using System.Collections.Generic;
using System.Text;

namespace Education.BLL.DTO.Pages
{
```

```
    public class BlogDTO
    {
        public int Page { get; set; }
        public int Pages { get; set; }
        public IEnumerable Notes { get; set; }
        public bool CanCreate { get; set; }
    }
}
```

Файл NoteDTO.cs

```
using System;
using System.Collections.Generic;
using System.Text;

namespace Education.BLL.DTO.Pages
{
    public class NoteDTO : NoteEditDTO
    {
        public DateTime Time { get; set; }
        public UserPublicInfoDTO Owner { get; set; }
        public AccessDTO Access { get; set; }

    }
}
```

Файл NoteEditDTO.cs

```
using System;
using System.Collections.Generic;
using System.Text;

namespace Education.BLL.DTO.Pages
{
    public class NoteEditDTO
    {
        public int Id { get; set; }
        public string Preview { get; set; }
        public string Name { get; set; }
        public string Text { get; set; }
        public bool Published { get; set; }
    }
}
```

Файл PageDTO.cs

```
using System;
using System.Collections.Generic;
using System.Text;

namespace Education.BLL.DTO.Pages
{
    public class PageDTO : PageEditDTO
    {
        public AccessDTO Access { get; set; }
        public IEnumerable ChildPages { get; set; }
        public string ParentName { get; set; }
    }
}
```

Файл PageEditDTO.cs

```
using System;
using System.Collections.Generic;
using System.Text;

namespace Education.BLL.DTO.Pages
```

```
{
    public class PageEditDTO : PagePreviewDTO
    {
        public int? ParentId { get; set; }
        public string Text { get; set; }
        public bool Published { get; set; }
    }
}
```

Файл PageInfo.cs

```
using System;
using System.Collections.Generic;
using System.Text;

namespace Education.BLL.DTO.Pages
{
    public class PageInfo
    {
        public int Id { get; set; }
        public string Name { get; set; }
        public IEnumerable Childs { get; set; }
        public PageInfo Parent { get; set; }
        public bool Published { get; set; }
    }
}
```

Файл PagePreviewDTO.cs

```
using System;
using System.Collections.Generic;
using System.Text;

namespace Education.BLL.DTO.Pages
{
    public class PagePreviewDTO
    {
        public int Id { get; set; }
        public string Name { get; set; }
    }
}
```

Файл PagesDTO.cs

```
using System;
using System.Collections.Generic;
using System.Text;

namespace Education.BLL.DTO.Pages
{
    public class PagesDTO
    {
        public bool CanCreate { get; set; }
        public IEnumerable MainPages { get; set; }
    }
}
```

Файл RoomDTO.cs

```
using System;
using System.Collections.Generic;
using System.Text;

namespace Education.BLL.DTO.Room
{
    public class RoomDTO
    {
```

```
        public UserPublicInfoDTO Companion { get; set; }
        public IEnumerable Messages { get; set; }
    }
}
```

Файл RoomMessageDTO.cs

```
using Education.DAL.Entities;
using System;
using System.Collections.Generic;
using System.Text;

namespace Education.BLL.DTO.Room
{
    public class RoomMessageDTO
    {
        public int Id { get; set; }
        public UserPublicInfoDTO From { get; set; }
        public UserPublicInfoDTO To { get; set; }
        public string Text { get; set; }
        public DateTime Time { get; set; }
    }
}
```

Файл RoomPreviewDTO.cs

```
using System;
using System.Collections.Generic;
using System.Text;

namespace Education.BLL.DTO.Room
{
    public class RoomPreviewDTO
    {
        public UserPublicInfoDTO Companion { get; set; }
        public RoomMessageDTO LastMessage { get; set; }
    }
}
```

Файл AdminUserInfoDTO.cs

```
using System;
using System.Collections.Generic;
using System.Text;

namespace Education.BLL.DTO.User
{
    public class AdminUserInfoDTO : UserInfoDTO
    {
        public int Id { get; set; }
        public string Login { get; set; }
        public short Level { get; set; }
        public bool Banned { get; set; }
        public string BanReason { get; set; }
        public DateTime? BanTime { get; set; }
    }
}
```

Файл ClaimInfoDTO.cs

```
using System;
using System.Collections.Generic;
using System.Text;

namespace Education.BLL.DTO.User
{
    public class ClaimInfoDTO
```

```
    {
        public DateTime LoginTime { get; set; }
        public string Ip { get; set; }
        public string Browser { get; set; }
    }
}
```

Файл LoginInfoDTO.cs

```
using System;
using System.Collections.Generic;
using System.Text;

namespace Education.BLL.DTO.User
{
    public class LoginInfoDTO
    {
        public string Login { get; set; }
        public string Password { get; set; }
        public string Key { get; set; }
        public string Browser { get; set; }
        public string IP { get; set; }
    }
}
```

Файл RegUserInfo.cs

```
using System;
using System.Collections.Generic;
using System.Text;

namespace Education.BLL.DTO.User
{
    public class RegUserInfo
    {
        public string Login { get; set; }
        public string Email { get; set; }
        public string Phone { get; set; }
        public string Name { get; set; }
        public string Password { get; set; }
    }
}
```

Файл UserDTO.cs

```
using System;
using System.Collections.Generic;
using System.Text;

namespace Education.BLL.DTO.User
{
    public class UserDTO
    {
        public int Id { get; set; }
        public int ClaimId { get; set; }
        public string ClaimValue { get; set; }
        /*public string Login { get; set; }
        public string FullName { get; set; }
        public string Email { get; set; }
        public string PhoneNumber { get; set; }
        public string Password { get; set; }*/
    }
}
```

Файл UserInfoDTO.cs

```
using Education.DAL.Entities;
```

```
using System;
using System.Collections.Generic;
using System.Text;

namespace Education.BLL.DTO.User
{
    public class UserInfoDTO
    {
        public string Avatar { get; set; }
        public string email { get; set; }
        public bool? emailConfirm { get; set; }
        public string phone { get; set; }
        public string name { get; set; }
        public bool? phoneConfirm { get; set; }
        public AuthType authType { get; set; }
    }
}
```

Файл UserProfileDTO.cs

```
using Education.DAL.Entities;
using System;
using System.Collections.Generic;
using System.Text;

namespace Education.BLL.DTO.User
{
    public class UserProfileDTO : UserInfoDTO
    {
        public IEnumerable Claims { get; set; }
    }
}
```

Файл AccessDTO.cs

```
using System;
using System.Collections.Generic;
using System.Text;

namespace Education.BLL.DTO
{
    public class AccessDTO
    {
        public bool CanRead { get; set; }
        public bool CanUpdate { get; set; }
        public bool CanDelete { get; set; }
    }
}
```

Файл AuthResult.cs

```
using Education.BLL.Services.UserServices.Interfaces;
using Education.DAL.Entities;
using System;
using System.Collections.Generic;
using System.Security.Claims;
using System.Text;

namespace Education.BLL.DTO
{
    public class AuthResult
    {
        public AuthStatus Status { get; set; }
        public AuthType authType { get; set; }
        public DateTime KeyTime { get; set; }
        public ClaimsIdentity Identity { get; set; }
```

```
        public string Comment { get; set; }

    }
}
```

Файл ConfirmResult.cs

```
using Education.BLL.Services.UserServices.Interfaces;
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text;

namespace Education.BLL.DTO
{
    public class ConfirmResult
    {
        public ConfirmCode Status { get; set; }
        public DateTime KeyTime { get; set; }
    }
}
```

Файл FileDTO.cs

```
using System;
using System.Collections.Generic;
using System.Text;

namespace Education.BLL.DTO
{
    public class FileDTO
    {
        public long Size { get; set; }
        public string Type { get; set; }
    }
}
```

Файл FullAccessDTO.cs

```
using System;
using System.Collections.Generic;
using System.Text;

namespace Education.BLL.DTO
{
    public class FullAccessDTO : AccessDTO
    {
        public bool CanCreateElements{ get; set; }
    }
}
```

Файл ImageInfoDTO.cs

```
using System;
using System.Collections.Generic;
using System.Text;

namespace Education.BLL.DTO
{
    public class ImageInfoDTO
    {
        public string Path { get; set; }
        public long Size { get; set; }
    }
}
```

Файл RestoreResult.cs

```
using Education.DAL.Entities;
using System;
using System.Collections.Generic;
using System.Text;

namespace Education.BLL.DTO
{
    public enum RestoreCode
    {
        Succsess,
        KeySent,
        WorngKey,
        UserNotFound,
        NeedContact,
        WrongPassword,
        NeedNewKey
    };
    public class RestoreResult
    {
        public RestoreCode Status { get; set; }
        public AuthType? SendTo { get; set; }
        public DateTime KeyTime { get; set; }
    }
}
```

Файл UserPublicInfoDTO.cs

```
using System;
using System.Collections.Generic;
using System.Text;

namespace Education.BLL.DTO
{
    public class UserPublicInfoDTO
    {
        public int Id { get; set; }
        public string Name { get; set; }
        public string AvatarPath { get; set; }
    }
}
```

Файл IDTOHelper.cs

```
using Education.BLL.DTO;
using Education.DAL.Entities;
using System;
using System.Collections.Generic;
using System.Text;

namespace Education.BLL.Logic.Interfaces
{
    public interface IDTOHelper
    {
        UserPublicInfoDTO GetUser(User user);
    }
}
```

Файл IForumDTOHelper.cs

```
using Education.BLL.DTO.Forum;
using Education.DAL.Entities;
using System;
using System.Collections.Generic;
using System.Text;
```

```
namespace Education.BLL.Logic.Interfaces
{
    public interface IForumDTOHelper
    {
        GroupDTO GetDTO(Group group, User user);
        SectionDTO GetDTO(Section section, User user);
        ThemeDTO GetDTO(Theme theme, User user, int page);
        ThemeDTO GetDTO(Theme theme, User user);
        MessageDTO GetDTO(Message message, User user);
        GroupPreviewDTO GetDTO(User user, Group group);
        UserGroupDTO GetDTO(UserGroup userGroup);
        MessagePreviewDTO GetDTO(Message message);
    }
}
```

Файл IGetUserDTO.cs

```
using Education.BLL.DTO.User;
using Education.DAL.Entities;
using Education.DAL.Interfaces;
using System;
using System.Collections.Generic;
using System.Text;

namespace Education.BLL.Logic.Interfaces
{
    public interface IGetUserDTO
    {
        User Get(UserDTO userDTO, IUOW Data);
    }
}
```

Файл IGroupRules.cs

```
using Education.DAL.Entities;
using System;
using System.Collections.Generic;
using System.Text;

namespace Education.BLL.Logic.Interfaces
{
    public interface IGroupRules
    {
        bool CanCreate(User user);

        bool CanEdit(User user, Group group);

        bool CanDelete(User user, Group group);

        bool CanRead(User user, Group group);

        bool CanControlUsers(User user, Group group);
    }
}
```

Файл IMessageRules.cs

```
using Education.DAL.Entities;
using System;
using System.Collections.Generic;
using System.Text;

namespace Education.BLL.Logic.Interfaces
{
    public interface IMessageRules
    {
```

```
        bool CanCreate(User user, Theme theme);

        bool CanEdit(User user, Message message);

        bool CanDelete(User user, Message message);

        bool CanRead(User user, Message message);
    }
}
```

Файл INoteRules.cs

```
using Education.DAL.Entities;
using Education.DAL.Entities.Pages;
using System;
using System.Collections.Generic;
using System.Text;

namespace Education.BLL.Logic.Interfaces
{
    public interface INoteRules
    {
        bool CanCreate(User user);
        bool CanEdit(User user, Note note);
        bool CanDelete(User user, Note note);
        bool CanRead(User user, Note note);
    }
}
```

Файл IPageMap.cs

```
using Education.BLL.DTO.Pages;
using Education.DAL.Entities.Pages;
using System;
using System.Collections.Generic;
using System.Text;

namespace Education.BLL.Logic.Interfaces
{
    public interface IPageMap
    {
        IEnumerable Get { get; }
        void Add(Page page);
        void Update(Page page);
        void Delete(int id);
    }
}
```

Файл IPageRules.cs

```
using Education.DAL.Entities;
using Education.DAL.Entities.Pages;
using System;
using System.Collections.Generic;
using System.Text;

namespace Education.BLL.Logic.Interfaces
{
    public interface IPageRules
    {
        bool CanCreate(User user);
        bool CanEdit(User user, Page page);
        bool CanDelete(User user, Page page);
        bool CanRead(User user, Page page);
    }
}
```

Файл ISectionRules.cs

```
using Education.DAL.Entities;
using System;
using System.Collections.Generic;
using System.Text;

namespace Education.BLL.Logic.Interfaces
{
    public interface ISectionRules
    {
        bool CanCreate(User user, Group group);

        bool CanEdit(User user, Section section);

        bool CanDelete(User user, Section section);

        bool CanRead(User user, Section section);
    }
}
```

Файл IThemeRules.cs

```
using Education.DAL.Entities;
using System;
using System.Collections.Generic;
using System.Text;

namespace Education.BLL.Logic.Interfaces
{
    public interface IThemeRules
    {
        bool CanCreate(User user, Section section);

        bool CanEdit(User user, Theme theme);

        bool CanDelete(User user, Theme theme);

        bool CanRead(User user, Theme theme);
    }
}
```

Файл GroupRules.cs

```
using Education.BLL.Logic.Interfaces;
using Education.DAL.Entities;
using System;
using System.Collections.Generic;
using System.Linq;

namespace Education.BLL.Logic.Rules
{
    public class GroupRules : IGroupRules
    {
        public bool CanCreate(User user)
        {
            if (user == null) return false;
            if (user.Level > 1) return true;
            else return false;
        }

        public bool CanEdit(User user, Group group)
        {
            if (group == null) throw new ArgumentNullException("group");
            if (user == null) return false;
```

```
            if (user.Level > 1) return true;

            var usergroup = group.Users.FirstOrDefault(x => x.User == user);
            if (usergroup != null && usergroup.Status ==
UserGroupStatus.owner)
                return true;

            return false;
        }

        public bool CanDelete(User user, Group group)
        {
            return CanCreate(user);
        }

        public bool CanRead(User user, Group group)
        {
            if (group == null) throw new ArgumentNullException("group");
            if (group.Open) return true;
            else
            {
                if (user == null) return false;
                if (user.Level > 0) return true;
                var usergroup = group.Users.FirstOrDefault(x => x.User ==
user);
                if (usergroup == null || usergroup.Status ==
UserGroupStatus.request)
                    return false;
            }
            return true;
        }

        public bool CanControlUsers(User user, Group group)
        {
            return CanEdit(user, group);
        }
    }
}
```

Файл MessageRules.cs

```
using Education.BLL.Logic.Interfaces;
using Education.DAL.Entities;
using System;
using System.Collections.Generic;
using System.Text;

namespace Education.BLL.Logic.Rules
{
    public class MessageRules : IMessageRules
    {
        private IThemeRules ThemeRules;
        private ISectionRules SectionRules;
        public MessageRules(IThemeRules themeRules, ISectionRules
sectionRules)
        {
            ThemeRules = themeRules;
            SectionRules = sectionRules;
        }

        public bool CanCreate(User user, Theme theme)
        {
            if (user == null) return false;
            if (theme == null) throw new ArgumentNullException("theme");
```

```
            if (theme.Open && ThemeRules.CanRead(user, theme)) return true;
            else if (!theme.Open && SectionRules.CanEdit(user,
theme.Section)) return true;
            else return false;
        }

        public bool CanDelete(User user, Message message)
        {
            return CanEdit(user, message);
        }

        public bool CanEdit(User user, Message message)
        {
            if (user == null) return false;
            if (message == null) throw new ArgumentNullException("theme");
            if (message.Theme == null)
                throw new NullReferenceException("message.Theme must be not
null");
            if (message.Theme.Open && message.Owner.Id == user.Id &&
ThemeRules.CanRead(user, message.Theme))
                return true;
            else if (SectionRules.CanEdit(user, message.Theme.Section))
                return true;
            else return false;
        }

        public bool CanRead(User user, Message message)
        {
            return ThemeRules.CanRead(user, message.Theme);
        }
    }
}
```

Файл NoteRules.cs

```
using Education.BLL.Logic.Interfaces;
using Education.DAL.Entities;
using Education.DAL.Entities.Pages;
using System;
using System.Collections.Generic;
using System.Text;

namespace Education.BLL.Logic.Rules
{
    public class NoteRules : INoteRules
    {
        public bool CanCreate(User user)
        {
            return user != null && user.Level > 0;
        }

        public bool CanDelete(User user, Note note)
        {
            return CanCreate(user);
        }

        public bool CanEdit(User user, Note note)
        {
            return CanCreate(user);
        }

        public bool CanRead(User user, Note note)
        {
            return note.Published || CanEdit(user, note);
```

```
        }
    }
}
```

Файл PageRules.cs

```
using Education.BLL.Logic.Interfaces;
using Education.DAL.Entities;
using Education.DAL.Entities.Pages;
using System;
using System.Collections.Generic;
using System.Text;

namespace Education.BLL.Logic.Rules
{
    public class PageRules : IPageRules
    {
        public bool CanCreate(User user)
        {
            return user != null && user.Level == 2;
        }

        public bool CanDelete(User user, Page page)
        {
            return CanCreate(user);
        }

        public bool CanEdit(User user, Page page)
        {
            return user != null && user.Level > 0;
        }

        public bool CanRead(User user, Page page)
        {
            return page.Published || CanEdit(user, page);
        }
    }
}
```

Файл SectionRules.cs

```
using Education.BLL.Logic.Interfaces;
using Education.DAL.Entities;
using System;
using System.Collections.Generic;
using System.Text;

namespace Education.BLL.Logic.Rules
{
    public class SectionRules : ISectionRules
    {
        private IGroupRules GroupRules;

        public SectionRules(IGroupRules groupRules)
        {
            GroupRules = groupRules;
        }

        public bool CanCreate(User user, Group group)
        {
            return GroupRules.CanEdit(user, group);
        }

        public bool CanDelete(User user, Section section)
        {
```

```
            return CanEdit(user, section);
        }

        public bool CanEdit(User user, Section section)
        {
            if (section == null) throw new ArgumentNullException("section");
            if (section.Group == null) throw new
NullReferenceException("section.Group must be not null");
            return GroupRules.CanEdit(user, section.Group);
        }

        public bool CanRead(User user, Section section)
        {
            if (section == null) throw new ArgumentNullException("section");
            if (section.Group == null) throw new
NullReferenceException("section.Group must be not null");
            return GroupRules.CanRead(user, section.Group);
        }
    }
}
```

Файл ThemeRules.cs

```
using Education.BLL.Logic.Interfaces;
using Education.DAL.Entities;
using System;
using System.Collections.Generic;
using System.Linq;

namespace Education.BLL.Logic.Rules
{
    public class ThemeRules : IThemeRules
    {
        private ISectionRules SectionRules;

        private bool IsOwner(Theme theme, User user)
        {
            var firstMessage = theme?.Messages?.FirstOrDefault();
            if (firstMessage == null || firstMessage.Owner.Id != user.Id)
return false;
            else return true;
        }

        public ThemeRules(ISectionRules sectionRules)
        {
            SectionRules = sectionRules;
        }

        public bool CanCreate(User user, Section section)
        {
            if (section == null) throw new ArgumentNullException("section");
            if (user == null) return false;
            if (section == null) throw new ArgumentNullException("section");
            if (section.Open && SectionRules.CanRead(user, section)) return
true;
            else if (!section.Open && SectionRules.CanEdit(user, section))
return true;
            else return false;
        }

        public bool CanDelete(User user, Theme theme)
        {
            return SectionRules.CanEdit(user, theme.Section);
        }
```

```
        public bool CanEdit(User user, Theme theme)
        {
            if (theme == null) throw new ArgumentNullException("theme");
            if (user == null) return false;
            if (IsOwner(theme, user)) return true;
            if (SectionRules.CanEdit(user, theme.Section)) return true;
            else return false;
        }

        public bool CanRead(User user, Theme theme)
        {
            return SectionRules.CanRead(user, theme.Section);
        }
    }
}
```

Файл ConfirmException.cs

```
using System;
using System.Collections.Generic;
using System.Text;

namespace Education.BLL.Logic
{
    public enum ConfirmError
    {
        NotFound,
        AlreadyConfirmed
    }

    public class ConfirmException : Exception
    {
        public ConfirmError Status { get; private set; }

        public ConfirmException(ConfirmError error) : base()
        {
            Status = error;
        }
    }
}
```

Файл DBInitException.cs

```
using System;
using System.Collections.Generic;
using System.Text;

namespace Education.BLL.Logic
{
    class DBInitException : Exception
    {
        public DBInitException(string reason) : base(reason)
        {

        }
    }
}
```

Файл DTOHelper.cs

```
using Education.BLL.DTO;
using Education.BLL.Logic.Interfaces;
using Education.DAL.Entities;
using System;
using System.Collections.Generic;
```

```
using System.Text;

namespace Education.BLL.Logic
{
    public class DTOHelper : IDTOHelper
    {
        public UserPublicInfoDTO GetUser(User user)
        {
            if (user == null) return null;
            return new UserPublicInfoDTO
            {
                AvatarPath = user.Info.Avatar,
                Id = user.Id,
                Name = user.Info.FullName
            };
        }
    }
}
```

Файл ForumDTOHelper.cs

```
using Education.BLL.DTO;
using Education.BLL.DTO.Forum;
using Education.BLL.Logic.Interfaces;
using Education.DAL.Entities;
using System;
using System.Collections.Generic;
using System.Linq;

namespace Education.BLL.Logic
{
    public class ForumDTOHelper : IForumDTOHelper
    {
        private IMessageRules MessageRules;
        private IThemeRules ThemeRules;
        private ISectionRules SectionRules;
        private IGroupRules GroupRules;
        private IDTOHelper DTOHelper;
        private static int MessagesPerPage = 10;
        private SectionRoute GetRoute(Section section)
        {
            if (section.Group == null) return null;
            return new SectionRoute
            {
                GroupId = section.Group.Id,
                GroupName = section.Group.Name
            };
        }

        private ThemeRoute GetRoute(Theme theme)
        {
            if (theme.Section == null) return null;
            return new ThemeRoute
            {
                SectionRoute = GetRoute(theme.Section),
                SectionId = theme.Section.Id,
                SectionName = theme.Section.Name
            };
        }

        private MessageRoute GetRoute(Message message)
        {
            if (message.Theme == null) return null;
            return new MessageRoute
```

```
            {
                ThemeRoute = GetRoute(message.Theme),
                ThemeId = message.Theme.Id,
                ThemeName = message.Theme.Name
            };
        }

        public ForumDTOHelper(IMessageRules messageRules, IThemeRules
themeRules, ISectionRules sectionRules, IGroupRules groupRules, IDTOHelper
dtoHelper)
        {
            MessageRules = messageRules;
            ThemeRules = themeRules;
            SectionRules = sectionRules;
            GroupRules = groupRules;
            DTOHelper = dtoHelper;
        }

        public GroupDTO GetDTO(Group group, User user)
        {
            if (group == null) return null;
            var sections = new List();
            foreach(var Section in group.Sections)
                sections.Add(GetDTO(Section, user));
            return new GroupDTO
            {
                Logo = group.Logo,
                Name = group.Name,
                Open = group.Open,
                Id = group.Id,
                Access = new GroupAccessDTO
                {
                    CanCreateElements = SectionRules.CanCreate(user, group),
                    CanDelete = GroupRules.CanDelete(user, group),
                    CanRead = GroupRules.CanRead(user, group),
                    CanUpdate = GroupRules.CanEdit(user, group),
                    CanControlUsers = GroupRules.CanControlUsers(user, group)
                },
                Sections = sections
            };
        }

        public SectionDTO GetDTO(Section section, User user)
        {
            if (section == null) return null;
            var themes = new List();
            foreach (var theme in section.Themes)
                themes.Add(new ThemePreviewDTO
                {
                    Id = theme.Id,
                    Pages = GetThemePages(theme),
                    Name = theme.Name,
                    Owner =
DTOHelper.GetUser(theme?.Messages?.FirstOrDefault()?.Owner),
                    LastMessages = GetDTO(theme?.Messages?.LastOrDefault(),
user)
                });
            return new SectionDTO
            {
                Id = section.Id,
                Name = section.Name,
                Open = section.Open,
                Themes = themes,
```

```
                Route = GetRoute(section),
                Access = new FullAccessDTO
                {
                    CanCreateElements = ThemeRules.CanCreate(user, section),
                    CanDelete = SectionRules.CanDelete(user, section),
                    CanRead = SectionRules.CanRead(user, section),
                    CanUpdate = SectionRules.CanEdit(user, section)
                }
            };
        }

        private ThemeDTO CreateThemeDTO(Theme theme, User user, IEnumerable
messages)
        {
            var mres = new List();
            foreach (var message in messages) mres.Add(GetDTO(message,
user));
            return new ThemeDTO
            {
                Name = theme.Name,
                Open = theme.Open,
                Messages = mres,
                Id = theme.Id,
                Route = GetRoute(theme),
                Access = new FullAccessDTO
                {
                    CanCreateElements = MessageRules.CanCreate(user, theme),
                    CanDelete = ThemeRules.CanDelete(user, theme),
                    CanRead = ThemeRules.CanRead(user, theme),
                    CanUpdate = ThemeRules.CanEdit(user, theme)
                }
            };
        }

        private int GetThemePages(Theme theme)
        {
            int res = theme.Messages.Count / MessagesPerPage;
            if (theme.Messages.Count % MessagesPerPage != 0) res++;
            return res;
        }

        private IEnumerable GetMessages(Theme theme, int page = 0)
        {
            var messages = theme.Messages.OrderBy(x => x.Time);
            if (page == 0) return messages;
            else return messages.Skip((page - 1) * MessagesPerPage)
                .Take(MessagesPerPage);
        }

        public ThemeDTO GetDTO(Theme theme, User user, int page)
        {
            if (theme == null) return null;
            if (page < 1) page = 1;
            var res = CreateThemeDTO(theme, user, GetMessages(theme, page));
            res.Pages = GetThemePages(theme);
            res.CurPage = page;
            return res;
        }

        public ThemeDTO GetDTO(Theme theme, User user)
        {
            if (theme == null) return null;
            var res = CreateThemeDTO(theme, user, GetMessages(theme));
```

```
            res.Pages = 1;
            res.CurPage = 1;
            return res;
        }

        public MessageDTO GetDTO(Message message, User user)
        {
            if (message == null) return null;
            return new MessageDTO
            {
                LastEditor = DTOHelper.GetUser(message.LastEditor),
                LastEditTime = message.LastEditTime,
                Owner = DTOHelper.GetUser(message.Owner),
                Time = message.Time,
                Text = message.Text,
                Id = message.Id,
                Route = GetRoute(message),
                Access = new AccessDTO
                {
                    CanDelete = MessageRules.CanDelete(user, message),
                    CanRead = MessageRules.CanRead(user, message),
                    CanUpdate = MessageRules.CanEdit(user, message)
                }

            };
        }

        public UserGroupDTO GetDTO(UserGroup userGroup)
        {
            return new UserGroupDTO
            {
                UserInfo = DTOHelper.GetUser(userGroup.User),
                Status = userGroup.Status
            };
        }

        public GroupPreviewDTO GetDTO(User user, Group group)
        {
            UserGroupInfo info = null;

            if(user != null)
            {
                info = new UserGroupInfo();
                var usergroup = group.Users.FirstOrDefault(x => x.UserId ==
user.Id);
                if (usergroup != null)
                {
                    info.Member = true;
                    info.Status = usergroup.Status;
                }
                else info.Member = false;
            }
            return new GroupPreviewDTO
            {
                Id = group.Id,
                Logo = group.Logo,
                Name = group.Name,
                Open = group.Open,
                Status = info,
                Access = new GroupAccessDTO
                {
                    CanCreateElements = SectionRules.CanCreate(user, group),
                    CanDelete = GroupRules.CanDelete(user, group),
```

```
                    CanRead = GroupRules.CanRead(user, group),
                    CanUpdate = GroupRules.CanEdit(user, group),
                    CanControlUsers = GroupRules.CanControlUsers(user,group)
                }
            };
        }

        public MessagePreviewDTO GetDTO(Message message)
        {
            var res = GetMessages(message.Theme).ToList();
            int page = res.IndexOf(message) / MessagesPerPage + 1;
            return new MessagePreviewDTO { Page = page, ThemeId =
message.Theme.Id };
        }
    }
}
```

Файл GetUserDTO.cs

```
using Education.BLL.DTO.User;
using Education.BLL.Logic.Interfaces;
using Education.DAL.Entities;
using Education.DAL.Interfaces;
using System.Linq;
using System.Collections.Generic;
using System.Text;
using System;

namespace Education.BLL.Logic
{
    public class GetUserDTO : IGetUserDTO
    {
        private bool CheckBan(User user)
        {
            if (user.Ban == null) return false;
            if (user.Ban.EndTime < DateTime.Now) return true;
            else return false;
        }

        public User Get(UserDTO userDTO, IUOW Data)
        {
            if (userDTO == null) return null;
            var user = Data.UserRepository.Get().FirstOrDefault(x => x.Id ==
userDTO.Id);
            if (CheckBan(user)) return null;
            return user;
        }
    }
}
```

Файл KeyException.cs

```
using System;
using System.Collections.Generic;
using System.Text;

namespace Education.BLL.Logic
{
    public enum KeyError
    {
        ContactNotFound,
        KeyNotFound
    }
    public class KeyException : Exception
    {
```

```
        public KeyError Status { get; private set; }
        public KeyException(KeyError status) : base()
        {
            Status = status;
        }
    }
}
```

Файл PageMap.cs

```
using Education.BLL.DTO.Pages;
using Education.BLL.Logic.Interfaces;
using Education.DAL.Entities.Pages;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading;

namespace Education.BLL.Logic
{
    public class PageMap : IPageMap
    {
        private Mutex UpdateMutex = new Mutex();

        public IEnumerable Get { get; private set; } = new List();

        private Dictionary Pages = new Dictionary();

        private void RemoveChild(PageInfo child)
        {
            if (child.Parent == null) return;
            (child.Parent.Childs as List).Remove(child);
            child.Parent = null;
        }

        private void AddChild(int ParrentId, PageInfo child)
        {
            PageInfo parent = null;
            if (!Pages.TryGetValue(ParrentId, out parent)) return;
            (parent.Childs as List).Add(child);
            child.Parent = parent;
        }

        public PageMap(IEnumerable pages)
        {
            Get = GenerateMap(pages);
        }

        private PageInfo GetPageInfo(Page page)
        {
            return new PageInfo { Id = page.Id, Name = page.Name, Published =
page.Published, Childs = new List() };
        }

        public void Add(Page page)
        {
            UpdateMutex.WaitOne();
            var PageInfo = GetPageInfo(page);
            if (page.ParentPage != null) {
                AddChild(page.ParentPage.Id, PageInfo);
            } else
                (Get as List).Add(PageInfo);
            Pages.Add(page.Id, PageInfo);
            UpdateMutex.ReleaseMutex();
```

```
}

public void Update(Page page)
{
    UpdateMutex.WaitOne();
    PageInfo PageInfo;
    if (!Pages.TryGetValue(page.Id, out PageInfo))
    {
        UpdateMutex.ReleaseMutex();
        return;
    }
    PageInfo.Name = page.Name;
    PageInfo.Published = page.Published;
    if(PageInfo.Parent == null)
    {
        if (page.ParentPage != null)
        {
            (Get as List).Remove(PageInfo);
            AddChild(page.ParentPage.Id, PageInfo);
        }
    }
    else
    {
        if(page.ParentPage == null)
        {
            RemoveChild(PageInfo);
            (Get as List).Add(PageInfo);
        }
        else if(page.ParentPage.Id != PageInfo.Parent.Id)
        {
            RemoveChild(PageInfo);
            AddChild(page.ParentPage.Id, PageInfo);
        }
    }
    UpdateMutex.ReleaseMutex();
}

public void Delete(int id)
{
    UpdateMutex.WaitOne();
    PageInfo PageInfo;
    if (!Pages.TryGetValue(id, out PageInfo)) return;

    if (PageInfo.Parent != null)
        RemoveChild(PageInfo);
    else
        (Get as List).Remove(PageInfo);
    Pages.Remove(id);

    UpdateMutex.ReleaseMutex();
}

private PageInfo GenerateElement(Page page)
{
    var elem = GetPageInfo(page);
    var childs = elem.Childs as List;
    Pages.Add(page.Id, elem);
    foreach (var ch in page.ChildPages)
    {
        var child = GenerateElement(ch);
        child.Parent = elem;
        childs.Add(child);
    }
```

```
            return elem;
        }

        private IEnumerable GenerateMap(IEnumerable pages)
        {
            Pages.Clear();
            var res = new List();
            var mainPages = pages.Where(x => x.ParentPage == null);
            foreach (var page in mainPages)
                res.Add(GenerateElement(page));
            return res;
        }


    }
}
```

Файл regRegularExpressions.cs

```
using System;
using System.Collections.Generic;
using System.Text;

namespace Education.BLL.Logic
{
    public static class regRegularExpressions
    {
        public static string EmailRegex = @".+@.+\..+";
        public static string LoginRegex = @"^[\w\d\-]{4,20}$";
        public static string PhoneRegex = @"^\+?\d{11}$";
        public static string PasswordRegex = @"((?=.*\d)(?=.*[a-z])(?=.*[A-
Z]).{6,20})";
        public static string FullNameRegex = @"^[А-Я][а-я]* [А-Я][а-я]* [А-
Я][а-я]*$";
    }
}
```

Файл UserAuthException.cs

```
using System;
using System.Collections.Generic;
using System.Text;

namespace Education.BLL.Logic
{
    public enum AuthError
    {
        UserNotFound,
        NeedNewKey,
        WrongKey,
        KeyAlreadySet,
        AuthTypeNotFound
    }
    public class UserAuthException : Exception
    {
        public AuthError Status { get; private set; }
        public UserAuthException(AuthError status) : base()
        {
            Status = status;
        }
    }
}
```

Файл IAdminService.cs

```
using Education.BLL.DTO.Forum;
```

```
using Education.BLL.DTO.User;
using System;
using System.Collections.Generic;
using System.Text;

namespace Education.BLL.Services.AdminService.Interfaces
{
    public interface IAdminService
    {
        (AccessCode Code, IEnumerable Result) Search(UserDTO userDTO, string
name);
        (AccessCode Code, IEnumerable Result) GetAll(UserDTO userDTO);
        (AccessCode Code, AdminUserInfoDTO Result) GetUser(UserDTO userDTO,
int id);
        AccessCode EditUser(UserDTO userDTO, AdminUserInfoDTO userInfo);
        AccessCode ResetClaims(UserDTO userDTO, int UserId);
        bool IsAdmin(UserDTO userDTO);
    }
}
```

Файл AdminService.cs

```
using Education.BLL.DTO.Forum;
using Education.BLL.DTO.User;
using Education.BLL.Logic.Interfaces;
using Education.BLL.Services.AdminService.Interfaces;
using Education.DAL.Entities;
using Education.DAL.Interfaces;
using System;
using System.Collections.Generic;
using System.Linq;

namespace Education.BLL.Services.AdminService
{
    public class AdminService : IAdminService
    {
        private IGetUserDTO GetUserService { get; set; }
        private IUOWFactory DataFactory { get; set; }

        public AdminService(IGetUserDTO getUserDTO, IUOWFactory dataFactory)
        {
            GetUserService = getUserDTO;
            DataFactory = dataFactory;
        }

        private AdminUserInfoDTO GetDTO(User user)
        {
            return new AdminUserInfoDTO
            {
                authType = user.authType,
                Login = user.Login,
                Avatar = user.Info?.Avatar,
                email = user.Email?.Value,
                phone = user.Phone?.Value,
                emailConfirm = user.Email?.Confirmed,
                phoneConfirm = user.Phone?.Confirmed,
                Id = user.Id,
                Level = user.Level,
                name = user.Info?.FullName,
                Banned = user.Ban != null,
                BanReason = user.Ban?.Reason,
                BanTime = user.Ban?.EndTime
            };
        }
```

```
        private void EditUser(User user, AdminUserInfoDTO userInfo)
        {
            if(userInfo.Level >= 0 || userInfo.Level <= 2)
                user.Level = userInfo.Level;
            user.authType = userInfo.authType;
            if (user.Info != null)
            {
                if(userInfo.Avatar != null)
                    user.Info.Avatar = userInfo.Avatar;
                if(userInfo.name != null)
                user.Info.FullName = userInfo.name;
            }
            if (user.Phone != null)
            {
                if(userInfo.phone != null)
                    user.Phone.Value = userInfo.phone;
                user.Phone.Confirmed = userInfo.phoneConfirm != null &&
userInfo.phoneConfirm == true;
            }
            else if(userInfo.phone != null)
            {
                user.Phone = new Contact
                {
                    Confirmed = userInfo.phoneConfirm != null &&
userInfo.phoneConfirm == true,
                    Value = userInfo.phone
                };
            }
            if (user.Email != null)
            {
                if(userInfo.email != null)
                    user.Email.Value = userInfo.email;
                user.Email.Confirmed = userInfo.emailConfirm != null &&
userInfo.emailConfirm == true;
            }
            else if (userInfo.email != null)
            {
                user.Email = new Contact
                {
                    Confirmed = userInfo.emailConfirm != null &&
userInfo.emailConfirm == true,
                    Value = userInfo.email
                };
            }
        }

        private void EditUserBan(User user, AdminUserInfoDTO userInfo, IUOW
Data)
        {
            if (user.Ban == null)
            {
                if (userInfo.Banned)
                    user.Ban = new Ban { Reason = "-", EndTime =
DateTime.Now.AddDays(1) };
            }

            if (user.Ban != null)
            {
                if (!userInfo.Banned)
                    Data.BanRepository.Delete(user.Ban);
                if (userInfo.BanReason != null)
                    user.Ban.Reason = userInfo.BanReason;
```

```
                if (userInfo.BanTime != null)
                    user.Ban.EndTime = (DateTime)userInfo.BanTime;
            }
        }

        public bool IsAdmin(UserDTO userDTO)
        {
            using(var Data = DataFactory.Get())
            {
                var user = GetUserService.Get(userDTO, Data);
                if (user == null || user.Level < 2) return false;
                else return true;
            }
        }

        public (AccessCode Code,IEnumerable Result) Search(UserDTO userDTO,
string name)
        {
            using(var Data = DataFactory.Get())
            {
                var user = GetUserService.Get(userDTO, Data);
                if (user == null || user.Level < 2) return
(AccessCode.NoPremision, null);
                var users = Data.UserRepository.Get().Where(x =>
x.Login.Contains(name) || x.Info.FullName.Contains(name));
                var res = new List();
                foreach (var us in users)
                    res.Add(GetDTO(us));
                return (AccessCode.Succsess, res);
            }
        }

        public (AccessCode Code, IEnumerable Result) GetAll(UserDTO userDTO)
        {
            using (var Data = DataFactory.Get())
            {
                var user = GetUserService.Get(userDTO, Data);
                if (user == null || user.Level < 2) return
(AccessCode.NoPremision, null);
                var users = Data.UserRepository.Get();
                var res = new List();
                foreach (var us in users)
                    res.Add(GetDTO(us));
                return (AccessCode.Succsess, res);
            }
        }

        public (AccessCode Code, AdminUserInfoDTO Result) GetUser(UserDTO
userDTO, int id)
        {
            using (var Data = DataFactory.Get())
            {
                var user = GetUserService.Get(userDTO, Data);
                if (user == null || user.Level < 2) return
(AccessCode.NoPremision, null);
                var res = Data.UserRepository.Get().FirstOrDefault(x => x.Id
== id);
                if (res == null) return (AccessCode.NotFound, null);
                return (AccessCode.Succsess, GetDTO(res));

            }
        }
```

```
        public AccessCode EditUser(UserDTO userDTO, AdminUserInfoDTO
userInfo)
        {
            using (var Data = DataFactory.Get())
            {
                var admin = GetUserService.Get(userDTO, Data);
                if (admin == null || admin.Level < 2) return
AccessCode.NoPremision;
                var user = Data.UserRepository.Get().FirstOrDefault(x => x.Id
== userInfo.Id);
                if (user == null) return AccessCode.NotFound;
                EditUser(user, userInfo);
                EditUserBan(user, userInfo, Data);
                Data.UserRepository.Edited(user);
                Data.SaveChanges();
                return AccessCode.Succsess;
            }
        }

        public AccessCode ResetClaims(UserDTO userDTO, int UserId)
        {
            using (var Data = DataFactory.Get())
            {
                var admin = GetUserService.Get(userDTO, Data);
                if (admin == null || admin.Level < 2) return
AccessCode.NoPremision;
                var user = Data.UserRepository.Get().FirstOrDefault(x => x.Id
== UserId);
                if (user == null) return AccessCode.NotFound;
                var claims = Data.UserClaimRepository.Get().Where(x => x.User
== user);
                Data.UserClaimRepository.Delete(claims);
                Data.SaveChanges();
                return AccessCode.Succsess;
            }
        }



    }
}
```

Файл IConfigService.cs

```
using System;
using System.Collections.Generic;
using System.Text;

namespace Education.BLL.Services.ConfigService.Interfaces
{
    public interface IConfigService
    {
        string ConnectionString { get; set; }
        string Name { get; set; }
        string IconPath { get; set; }
    }
}
```

Файл XMLConfigService.cs

```
using Education.BLL.Services.ConfigService.Interfaces;
using System;
using System.Collections.Generic;
using System.IO;
using System.Text;
```

```
using System.Threading;
using System.Xml.Serialization;

namespace Education.BLL.Services
{
    public class Data
    {
        public string ConnectionString { get; set; }
        public string Name { get; set; }
        public string IconPath { get; set; }
    }

    public class XMLConfigService : IConfigService
    {
        private string Path;
        private Data Config;

        private Mutex Mutex = new Mutex();

        public string ConnectionString
        {
            get
            {
                return Config.ConnectionString;
            }
            set
            {
                Mutex.WaitOne();
                try
                {
                    Config.ConnectionString = value;
                    Save();
                }
                finally
                {
                    Mutex.ReleaseMutex();
                }

            }
        }

        public string Name
        {
            get
            {
                return Config.Name;
            }
            set
            {
                Mutex.WaitOne();
                try
                {
                    Config.Name = value;
                    Save();
                }
                finally
                {
                    Mutex.ReleaseMutex();
                }
            }
        }

        public string IconPath
```

```
        {
            get
            {
                return Config.IconPath;
            }
            set
            {
                Mutex.WaitOne();
                try
                {
                    Config.IconPath = value;
                    Save();
                }
                finally
                {
                    Mutex.ReleaseMutex();
                }
            }
        }

        private void Save()
        {
            XmlSerializer formatter = new XmlSerializer(typeof(Data));
            using (FileStream fs = new FileStream(Path, FileMode.Create))
                formatter.Serialize(fs, Config);
        }

        private void CreateNew()
        {
            Config = new Data
            {
                ConnectionString =
@"Server=(localdb)\mssqllocaldb;Database=EDCTest;Trusted_Connection=True;",
                IconPath = "favicon.ico",
                Name = "EducationPortal"
            };
            Save();
        }

        private void Load()
        {
            if (!File.Exists(Path))
            {
                CreateNew();
                return;
            }

            XmlSerializer formatter = new XmlSerializer(typeof(Data));
            using (FileStream fs = new FileStream(Path,
FileMode.OpenOrCreate))
                Config = (Data)formatter.Deserialize(fs);
        }

        public XMLConfigService(string path)
        {
            Path = path;
            Load();
        }

    }
}
```

Файл IControlService.cs

```
using Education.BLL.DTO.Forum;
using Education.BLL.DTO.User;
using Education.DAL.Entities;
using System;
using System.Collections.Generic;
using System.Text;

namespace Education.BLL.Services.ForumServices.Interfaces
{
    public interface IControlService
    {
        CreateResultDTO Create(T DTO, UserDTO userDTO);

        AccessCode Update(T DTO, UserDTO userDTO);

        AccessCode Delete(int id, UserDTO userDTO);

        (AccessCode, D) Read(int id, UserDTO userDTO);

    }
}
```

Файл IForumService.cs

```
using Education.BLL.DTO.Forum;
using Education.BLL.DTO.User;
using System;
using System.Collections.Generic;
using System.Text;

namespace Education.BLL.Services.ForumServices.Interfaces
{
    public interface IForumService
    {
        ForumDTO Get(UserDTO userDTO);
    }
}
```

Файл IGroupService.cs

```
using Education.BLL.DTO.Forum;
using Education.BLL.DTO.Forum.Edit;
using Education.BLL.DTO.User;
using Education.DAL.Entities;
using System;
using System.Collections.Generic;
using System.Text;

namespace Education.BLL.Services.ForumServices.Interfaces
{
    public interface IGroupService : IControlService
    {
        UserGroupInfoDTO GetUsers(int groupId, UserDTO userDTO);
        AccessCode ChangeUserRole(int groupId, int userId, UserGroupStatus
status, UserDTO userDTO);
        AccessCode RemoveUser(int groupId, int userId, UserDTO userDTO);
        AccessCode Request(UserDTO userDTO, int GroupId);
        AccessCode Leave(UserDTO userDTO, int GroupId);
        bool CanCreate(UserDTO userDTO);
    }
}
```

Файл IMessageService.cs

```
using Education.BLL.DTO.Forum;
using Education.BLL.DTO.Forum.Edit;
```

```
using Education.BLL.DTO.User;
using System;
using System.Collections.Generic;
using System.Text;

namespace Education.BLL.Services.ForumServices.Interfaces
{
    public interface IMessageService : IControlService
    {
        bool CanCreate(int ThemeId, UserDTO userDTO);

        MessagePreviewDTO Get(int id, UserDTO userDTO);
    }
}
```

Файл ISectionService.cs

```
using Education.BLL.DTO.Forum;
using Education.BLL.DTO.Forum.Edit;
using Education.BLL.DTO.User;
using System;
using System.Collections.Generic;
using System.Text;

namespace Education.BLL.Services.ForumServices.Interfaces
{
    public interface ISectionService : IControlService
    {
        bool CanCreate(int GroupId, UserDTO userDTO);
    }
}
```

Файл IThemeService.cs

```
using Education.BLL.DTO.Forum;
using Education.BLL.DTO.Forum.Edit;
using Education.BLL.DTO.User;
using System;
using System.Collections.Generic;
using System.Text;

namespace Education.BLL.Services.ForumServices.Interfaces
{
    public interface IThemeService : IControlService
    {
        bool CanCreate(int SectionId, UserDTO userDTO);
        (AccessCode, ThemeDTO) Read(int id, int page, UserDTO userDTO);
    }
}
```

Файл ForumService.cs

```
using Education.BLL.DTO.Forum;
using Education.BLL.DTO.User;
using Education.BLL.Logic.Interfaces;
using Education.BLL.Services.ForumServices.Interfaces;
using Education.DAL.Interfaces;
using System;
using System.Collections.Generic;
using System.Text;

namespace Education.BLL.Services.ForumServices
{
    public class ForumService : IForumService
    {
        private IUOWFactory DataFactory;
```

```
        private IForumDTOHelper ForumDTOHelper;
        private IGroupRules GroupRules;
        private IGetUserDTO GetUserService;

        public ForumService(IGetUserDTO getUserDTO, IUOWFactory dataFactory,
IForumDTOHelper forumDTOHelper, IGroupRules groupRules)
        {
            GetUserService = getUserDTO;
            DataFactory = dataFactory;
            ForumDTOHelper = forumDTOHelper;
            GroupRules = groupRules;
        }

        public ForumDTO Get(UserDTO userDTO)
        {
            using (var Data = DataFactory.Get()) {
                var user = GetUserService.Get(userDTO, Data);
                var groups = new List();
                foreach(var Group in Data.GroupRepository.Get())
                    groups.Add(ForumDTOHelper.GetDTO(user, Group));
                return new ForumDTO { Groups = groups, CanCreateGroup =
GroupRules.CanCreate(user) };
            }
        }
    }
}
```

Файл GroupService.cs

```
using Education.BLL.DTO.Forum;
using Education.BLL.DTO.Forum.Edit;
using Education.BLL.DTO.User;
using Education.BLL.Logic.Interfaces;
using Education.BLL.Services.ForumServices.Interfaces;
using Education.BLL.Services.UserServices.Interfaces;
using Education.DAL.Entities;
using Education.DAL.Interfaces;
using System;
using System.Collections.Generic;
using System.Linq;

namespace Education.BLL.Services.ForumServices
{
    public class GroupService : IGroupService
    {
        private IGetUserDTO getUserService;
        private IUOWFactory DataFactory;
        private IForumDTOHelper ForumDTOHelper;
        private IGroupRules GroupRule;

        private (AccessCode Code, Group Group, User User) CheckRule(UserDTO
userDTO, int GroupId, Func checkFunc, IUOW Data)
        {
            var user = getUserService.Get(userDTO, Data);
            var group = Data.GroupRepository.Get().FirstOrDefault(x => x.Id
== GroupId);
            if (group == null) return (AccessCode.NotFound, null, user);
            if (checkFunc(user, group)) return (AccessCode.Succsess, group,
user);
            else return (AccessCode.NoPremision, null, user);
        }

        private void EditGroup(GroupEditDTO groupDTO, Group group)
        {
```

```
            group.Logo = groupDTO.Logo;
            group.Name = groupDTO.Name;
            group.Open = groupDTO.Open;
        }

        public GroupService(IGroupRules rules, IGetUserDTO getUserDTO,
IUOWFactory dataFactory, IForumDTOHelper forumDTOHelper)
        {
            getUserService = getUserDTO;
            DataFactory = dataFactory;
            ForumDTOHelper = forumDTOHelper;
            GroupRule = rules;
        }

        public CreateResultDTO Create(GroupEditDTO groupDTO, UserDTO userDTO)
        {
            using (var Data = DataFactory.Get())
            {
                var user = getUserService.Get(userDTO, Data);
                if (GroupRule.CanCreate(user))
                {
                    var group = new Group();
                    EditGroup(groupDTO,group);
                    Data.GroupRepository.Add(group);
                    Data.SaveChanges();
                    return new CreateResultDTO(group.Id,
AccessCode.Succsess);
                }
            }
            return CreateResultDTO.NoPremision;
        }

        public AccessCode Update(GroupEditDTO groupDTO, UserDTO userDTO)
        {
            using (var Data = DataFactory.Get())
            {
                var check = CheckRule(userDTO, groupDTO.Id,
GroupRule.CanEdit, Data);
                if (check.Code == AccessCode.Succsess)
                {
                    EditGroup(groupDTO, check.Group);
                    Data.GroupRepository.Edited(check.Group);
                    Data.SaveChanges();
                }
                return check.Code;
            }
        }

        public AccessCode Delete(int id, UserDTO userDTO)
        {
            using (var Data = DataFactory.Get())
            {
                var check = CheckRule(userDTO, id, GroupRule.CanDelete,
Data);
                if (check.Code == AccessCode.Succsess)
                {
                    Data.GroupRepository.Delete(check.Group);
                    Data.SaveChanges();
                }
                return check.Code;
            }
        }
```

```
        public  (AccessCode,GroupDTO) Read(int id, UserDTO userDTO)
        {
            using (var Data = DataFactory.Get())
            {
                var check = CheckRule(userDTO, id, GroupRule.CanRead, Data);
                if (check.Code == AccessCode.Succsess)
                    return (check.Code, ForumDTOHelper.GetDTO(check.Group,
check.User));
                else return (check.Code, null);
            }
        }

        public AccessCode Request(UserDTO userDTO, int GroupId)
        {
            using(var Data = DataFactory.Get())
            {
                var user = getUserService.Get(userDTO, Data);
                if (user == null) return AccessCode.NoPremision;
                var group = Data.GroupRepository.Get().FirstOrDefault(x =>
x.Id == GroupId);
                if (group == null) return AccessCode.NotFound;
                if (group.Open) return AccessCode.NoPremision;
                var usergroup = group.Users.FirstOrDefault(x => x.UserId ==
user.Id);
                if (usergroup != null) return AccessCode.Error;
                Data.UserGroupRepository.Add(new UserGroup
                {
                    Group = group,
                    User = user,
                    Status = UserGroupStatus.request
                });
                Data.SaveChanges();
                return AccessCode.Succsess;
            }
        }

        public AccessCode Leave(UserDTO userDTO, int GroupId)
        {
            using (var Data = DataFactory.Get())
            {
                var user = getUserService.Get(userDTO, Data);
                if (user == null) return AccessCode.NoPremision;
                var group = Data.GroupRepository.Get().FirstOrDefault(x =>
x.Id == GroupId);
                if (group == null) return AccessCode.NotFound;
                if (group.Open) return AccessCode.NoPremision;
                var usergroup = group.Users.FirstOrDefault(x => x.UserId ==
user.Id);
                if (usergroup == null) return AccessCode.Error;
                Data.UserGroupRepository.Delete(usergroup);
                Data.SaveChanges();
                return AccessCode.Succsess;
            }
        }

        public UserGroupInfoDTO GetUsers(int groupId, UserDTO userDTO)
        {
            using (var Data = DataFactory.Get())
            {
                var check = CheckRule(userDTO, groupId,
GroupRule.CanControlUsers, Data);
                if (check.Code == AccessCode.Succsess)
                {
```

```
                    return new UserGroupInfoDTO
                    {
                        Code = AccessCode.Succsess,
                        Data = check.Group.Users.Select(x =>
ForumDTOHelper.GetDTO(x)).ToList()
                    };
                }
                else if (check.Code == AccessCode.NoPremision) return
UserGroupInfoDTO.NoPremision;
                else if (check.Code == AccessCode.NotFound) return
UserGroupInfoDTO.NotFound;
                else return UserGroupInfoDTO.Error;
            }
        }

        public AccessCode ChangeUserRole(int groupId, int userId,
UserGroupStatus status, UserDTO userDTO)
        {
            using (var Data = DataFactory.Get())
            {
                var check = CheckRule(userDTO, groupId,
GroupRule.CanControlUsers, Data);
                if (check.Code == AccessCode.Succsess)
                {
                    var userGroup = check.Group.Users.FirstOrDefault(x =>
x.UserId == userId);
                    if (userGroup == null) return AccessCode.NotFound;
                    else userGroup.Status = status;
                    Data.UserGroupRepository.Edited(userGroup);
                    Data.SaveChanges();
                }
                return check.Code;
            }
        }

        public AccessCode RemoveUser(int groupId, int userId, UserDTO
userDTO)
        {
            using (var Data = DataFactory.Get())
            {
                var check = CheckRule(userDTO, groupId,
GroupRule.CanControlUsers, Data);
                if (check.Code == AccessCode.Succsess)
                {
                    var userGroup = check.Group.Users.FirstOrDefault(x =>
x.UserId == userId);
                    if (userGroup == null) return AccessCode.NotFound;
                    Data.UserGroupRepository.Delete(userGroup);
                    Data.SaveChanges();
                }
                return check.Code;
            }
        }

        public bool CanCreate(UserDTO userDTO)
        {
            using(var Data = DataFactory.Get())
            {
                var user = getUserService.Get(userDTO,Data);
                return GroupRule.CanCreate(user);
            }
        }
    }
```

```
}
```

Файл MessageService.cs

```
using Education.BLL.DTO.Forum;
using Education.BLL.DTO.Forum.Edit;
using Education.BLL.DTO.User;
using Education.BLL.Logic.Interfaces;
using Education.BLL.Services.ForumServices.Interfaces;
using Education.DAL.Entities;
using Education.DAL.Interfaces;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace Education.BLL.Services.ForumServices
{
    public class MessageService : IMessageService
    {

        private IMessageRules MessageRules;
        private IGetUserDTO GetUserService;
        private IUOWFactory DataFactory;
        private IForumDTOHelper forumDTOHelper;

        public MessageService(IMessageRules rules, IGetUserDTO getUserDTO,
IUOWFactory dataFactory, IForumDTOHelper forumHelper)
        {
            GetUserService = getUserDTO;
            DataFactory = dataFactory;
            forumDTOHelper = forumHelper;
            MessageRules = rules;
        }

        private void EditMessage(Message message, MessageEditDTO messageDTO,
User user, bool edit = true)
        {
            message.Text = messageDTO.Text;
            if (edit)
            {
                message.LastEditor = user;
                message.LastEditTime = DateTime.Now;
            }
            else
            {
                message.Owner = user;
                message.Time = DateTime.Now;
            }
        }

        private bool IsFirstMessage(Message message)
        {
            if (message.Theme == null) return false;
            if (message?.Theme?.Messages?.OrderBy(X => X.Id).FirstOrDefault()
== message) return true;
            return false;
        }

        private (AccessCode Code, Message Message, User User)
CheckMessage(UserDTO userDTO, int MessageId, Func checkFunc, IUOW Data)
        {
            var user = GetUserService.Get(userDTO, Data);
```

```
            var message = Data.MessageRepository.Get().FirstOrDefault(x =>
x.Id == MessageId);
            if (message == null) return (AccessCode.NotFound, null, user);
            if (checkFunc(user, message)) return (AccessCode.Succsess,
message, user);
            else return (AccessCode.NoPremision, null, user);
        }

        public CreateResultDTO Create(MessageEditDTO DTO, UserDTO userDTO)
        {
            using(var Data = DataFactory.Get())
            {
                var user = GetUserService.Get(userDTO, Data);
                var theme = Data.ThemeRepository.Get().FirstOrDefault(x =>
x.Id == DTO.ThemeId);
                if (theme == null) return CreateResultDTO.NotFound;
                if (MessageRules.CanCreate(user, theme))
                {
                    var message = new Message();
                    EditMessage(message, DTO, user, false);
                    message.Theme = theme;
                    Data.MessageRepository.Add(message);
                    Data.SaveChanges();
                    return new CreateResultDTO(message.Id,
AccessCode.Succsess);
                }
                else return CreateResultDTO.NoPremision;
            }
        }

        public AccessCode Delete(int id, UserDTO userDTO)
        {
            using (var Data = DataFactory.Get())
            {
                var check = CheckMessage(userDTO, id, MessageRules.CanDelete,
Data);
                if (check.Code == AccessCode.Succsess)
                {
                    if (IsFirstMessage(check.Message)) return
AccessCode.NoPremision;
                    Data.MessageRepository.Delete(check.Message);
                    Data.SaveChanges();
                }
                return check.Code;
            }
        }

        public (AccessCode, MessageDTO) Read(int id, UserDTO userDTO)
        {
            using (var Data = DataFactory.Get())
            {
                var check = CheckMessage(userDTO, id, MessageRules.CanRead,
Data);
                if (check.Code == AccessCode.Succsess)
                    return (check.Code, forumDTOHelper.GetDTO(check.Message,
check.User));
                else return (check.Code, null);
            }
        }

        public AccessCode Update(MessageEditDTO DTO, UserDTO userDTO)
        {
            using (var Data = DataFactory.Get())
```

```
            {
                var check = CheckMessage(userDTO, DTO.Id,
MessageRules.CanRead, Data);
                if (check.Code == AccessCode.Succsess)
                {
                    if (IsFirstMessage(check.Message)) return
AccessCode.NoPremision;
                    EditMessage(check.Message, DTO, check.User);
                    Data.MessageRepository.Edited(check.Message);
                    Data.SaveChanges();
                }
                return check.Code;
            }
        }

        public bool CanCreate(int ThemeId, UserDTO userDTO)
        {
            using (var Data = DataFactory.Get())
            {
                var user = GetUserService.Get(userDTO, Data);
                var theme = Data.ThemeRepository.Get().FirstOrDefault(x =>
x.Id == ThemeId);
                if (theme == null) return false;
                return MessageRules.CanCreate(user, theme);
            }
        }


        public MessagePreviewDTO Get(int id, UserDTO userDTO)
        {
            using (var Data = DataFactory.Get())
            {
                var user = GetUserService.Get(userDTO, Data);
                var message = Data.MessageRepository.Get().FirstOrDefault(x
=> x.Id == id);
                if (message == null || !MessageRules.CanRead(user, message))
return null;
                return forumDTOHelper.GetDTO(message);
            }
        }
    }
}
```

Файл SectionService.cs

```
using Education.BLL.DTO.Forum;
using Education.BLL.DTO.Forum.Edit;
using Education.BLL.DTO.User;
using Education.BLL.Logic.Interfaces;
using Education.BLL.Services.ForumServices.Interfaces;
using Education.DAL.Entities;
using Education.DAL.Interfaces;
using System;
using System.Collections.Generic;
using System.Linq;


namespace Education.BLL.Services.ForumServices
{
    public class SectionService : ISectionService
    {
        private ISectionRules SectionRules;
        private IGetUserDTO GetUserService;
        private IUOWFactory DataFactory;
```

```
        private IForumDTOHelper forumDTOHelper;

        public SectionService(ISectionRules rules, IGetUserDTO getUserDTO,
IUOWFactory dataFactory, IForumDTOHelper forumHelper)
        {
            GetUserService = getUserDTO;
            DataFactory = dataFactory;
            forumDTOHelper = forumHelper;
            SectionRules = rules;
        }

        private (AccessCode Code, Section Section, User User)
CheckSection(UserDTO userDTO, int SectionId, Func checkFunc, IUOW Data)
        {
            var user = GetUserService.Get(userDTO, Data);
            var section = Data.SectionRepository.Get().FirstOrDefault(x =>
x.Id == SectionId);
            if (section == null) return (AccessCode.NotFound, null, user);
            if (checkFunc(user, section)) return (AccessCode.Succsess,
section, user);
            else return (AccessCode.NoPremision, null, user);
        }

        private void EditSection(SectionEditDTO sectionDTO, Section section)
        {
            section.Name = sectionDTO.Name;
            section.Open = sectionDTO.Open;
        }

        public CreateResultDTO Create(SectionEditDTO DTO, UserDTO userDTO)
        {
            using(var Data = DataFactory.Get())
            {
                var user = GetUserService.Get(userDTO, Data);
                var group = Data.GroupRepository.Get().FirstOrDefault(x =>
x.Id == DTO.GroupId);
                if (group == null) return CreateResultDTO.NotFound;
                if (SectionRules.CanCreate(user, group))
                {
                    var section = new Section();
                    EditSection(DTO, section);
                    section.Group = group;
                    Data.SectionRepository.Add(section);
                    Data.SaveChanges();
                    return new CreateResultDTO(section.Id,
AccessCode.Succsess);
                }
                else return CreateResultDTO.NoPremision;
            }
        }

        public AccessCode Delete(int id, UserDTO userDTO)
        {
            using (var Data = DataFactory.Get())
            {
                var check = CheckSection(userDTO, id, SectionRules.CanDelete,
Data);
                if (check.Code == AccessCode.Succsess)
                {
                    Data.SectionRepository.Delete(check.Section);
                    Data.SaveChanges();
                }
                return check.Code;
```

```
            }
        }

        public (AccessCode, SectionDTO) Read(int id, UserDTO userDTO)
        {
            using (var Data = DataFactory.Get())
            {
                var check = CheckSection(userDTO, id, SectionRules.CanRead,
Data);
                if (check.Code == AccessCode.Succsess)
                    return (check.Code, forumDTOHelper.GetDTO(check.Section,
check.User));
                else return (check.Code, null);
            }
        }

        public AccessCode Update(SectionEditDTO DTO, UserDTO userDTO)
        {
            using (var Data = DataFactory.Get())
            {
                var check = CheckSection(userDTO, DTO.Id,
SectionRules.CanEdit, Data);
                if (check.Code == AccessCode.Succsess)
                {
                    EditSection(DTO, check.Section);
                    Data.SectionRepository.Edited(check.Section);
                    Data.SaveChanges();
                }
                return check.Code;
            }
        }

        public bool CanCreate(int GroupId, UserDTO userDTO)
        {
            using (var Data = DataFactory.Get())
            {
                var user = GetUserService.Get(userDTO, Data);
                var group = Data.GroupRepository.Get().FirstOrDefault(x =>
x.Id == GroupId);
                if (group == null) return false;
                return SectionRules.CanCreate(user, group);
            }
        }
    }
}
```

Файл ThemeService.cs

```
using Education.BLL.DTO.Forum;
using Education.BLL.DTO.Forum.Edit;
using Education.BLL.DTO.User;
using Education.BLL.Logic.Interfaces;
using Education.BLL.Services.ForumServices.Interfaces;
using Education.DAL.Entities;
using Education.DAL.Interfaces;
using System;
using System.Collections.Generic;
using System.Linq;

namespace Education.BLL.Services.ForumServices
{
    public class ThemeService : IThemeService
    {
        private IThemeRules ThemeRules;
```

```
        private IGetUserDTO GetUserService;
        private IUOWFactory DataFactory;
        private IForumDTOHelper forumDTOHelper;

        public ThemeService(IThemeRules rules, IGetUserDTO getUserDTO,
IUOWFactory dataFactory, IForumDTOHelper forumHelper)
        {
            GetUserService = getUserDTO;
            DataFactory = dataFactory;
            forumDTOHelper = forumHelper;
            ThemeRules = rules;
        }

        private (AccessCode Code,Theme Theme, User User) CheckTheme(UserDTO
userDTO, int ThemeId, Func checkFunc, IUOW Data)
        {
            var user = GetUserService.Get(userDTO, Data);
            var theme = Data.ThemeRepository.Get().FirstOrDefault(x => x.Id
== ThemeId);
            if (theme == null) return (AccessCode.NotFound,null,user);
            if (checkFunc(user, theme)) return
(AccessCode.Succsess,theme,user);
            else return (AccessCode.NoPremision,null,user);
        }

        private void EditTheme(Theme theme, ThemeEditDTO themeDTO)
        {
            theme.Name = themeDTO.Name;
            theme.Open = themeDTO.Open;
            var fs = theme?.Messages?.FirstOrDefault();
            if (fs == null) return;
            fs.Text = themeDTO.Text;
        }

        public CreateResultDTO Create(ThemeEditDTO DTO, UserDTO userDTO)
        {
            using(var Data = DataFactory.Get())
            {
                var user = GetUserService.Get(userDTO,Data);
                var section = Data.SectionRepository.Get().FirstOrDefault(x
=> x.Id == DTO.SectionId);
                if (section == null) return CreateResultDTO.NotFound;
                if (ThemeRules.CanCreate(user, section))
                {
                    var theme = new Theme();
                    theme.Messages.Add(
                        new Message
                        {
                            Owner = user,
                            Theme = theme,
                            Time = DateTime.Now
                        }
                    );

                    EditTheme(theme, DTO);
                    theme.Section = section;

                    Data.ThemeRepository.Add(theme);
                    Data.SaveChanges();
                    return new CreateResultDTO(theme.Id,
AccessCode.Succsess);
                }
                else return CreateResultDTO.NoPremision;
```

```
            }
        }

        public AccessCode Delete(int id, UserDTO userDTO)
        {
            using (var Data = DataFactory.Get())
            {
                var check = CheckTheme(userDTO, id, ThemeRules.CanDelete,
Data);
                if(check.Code == AccessCode.Succsess)
                {
                    Data.ThemeRepository.Delete(check.Theme);
                    Data.SaveChanges();
                }
                return check.Code;
            }
        }

        public (AccessCode, ThemeDTO) Read(int id, UserDTO userDTO)
        {
            using (var Data = DataFactory.Get())
            {
                var check = CheckTheme(userDTO, id, ThemeRules.CanRead,
Data);
                if (check.Code == AccessCode.Succsess)
                    return (check.Code, forumDTOHelper.GetDTO(check.Theme,
check.User));
                else return (check.Code, null);
            }
        }

        public AccessCode Update(ThemeEditDTO DTO, UserDTO userDTO)
        {
            using (var Data = DataFactory.Get())
            {
                var check = CheckTheme(userDTO, DTO.Id, ThemeRules.CanEdit,
Data);
                if (check.Code == AccessCode.Succsess)
                {
                    EditTheme(check.Theme,DTO);
                    Data.ThemeRepository.Edited(check.Theme);
                    Data.SaveChanges();
                }
                return check.Code;
            }
        }

        public bool CanCreate(int SectionId, UserDTO userDTO)
        {
            using (var Data = DataFactory.Get())
            {
                var user = GetUserService.Get(userDTO, Data);
                var section = Data.SectionRepository.Get().FirstOrDefault(x
=> x.Id == SectionId);
                if (section == null) return false;
                return ThemeRules.CanCreate(user, section);
            }
        }

        public (AccessCode, ThemeDTO) Read(int id, int page, UserDTO userDTO)
        {
            using (var Data = DataFactory.Get())
            {
```

```
                var check = CheckTheme(userDTO, id, ThemeRules.CanRead,
Data);
                if (check.Code == AccessCode.Succsess)
                    return (check.Code, forumDTOHelper.GetDTO(check.Theme,
check.User,page));
                else return (check.Code, null);
            }
        }
    }
}
```

Файл IImageService.cs

```
using Education.BLL.DTO;
using Education.BLL.DTO.Forum;
using Education.BLL.DTO.User;
using System;
using System.Collections.Generic;
using System.Text;

namespace Education.BLL.Services.ImageManager.Interfaces
{
    public interface IImageService
    {
        string ImageFolder { get; }
        IEnumerable AllowedImgTypes { get; }
        bool CanLoad(UserDTO userDTO);
        bool CheckFile(FileDTO fileDTO);
        string AvatarPath(UserDTO userDTO);
        IEnumerable Get();
        AccessCode Delete(string path, UserDTO user);
        bool CanDelete(UserDTO userDTO);
    }
}
```

Файл ImageService.cs

```
using Education.BLL.DTO;
using Education.BLL.DTO.Forum;
using Education.BLL.DTO.User;
using Education.BLL.Logic;
using Education.BLL.Logic.Interfaces;
using Education.BLL.Services.ImageManager.Interfaces;
using Education.DAL.Interfaces;
using System;
using System.Collections.Generic;
using System.IO;
using System.Text;

namespace Education.BLL.Services.ImageManager
{
    public class ImageService : IImageService
    {
        private IUOWFactory DataFactory;
        private IGetUserDTO GetUserService;
        private long MaxSize;

        public string ImageFolder { get; private set; } = @"wwwroot/images/";

        public IEnumerable AllowedImgTypes { get; private set; }
            = new string[] { ".jpg", ".jpeg", ".bmp", ".png" };

        public ImageService(IUOWFactory dataFactory, IGetUserDTO getUserDTO,
long maxSize = 1000000)
        {
```

```
    DataFactory = dataFactory;
    GetUserService = getUserDTO;
    MaxSize = maxSize;
}

public bool CanLoad(UserDTO userDTO)
{
    using(var Data = DataFactory.Get())
    {
        var user = GetUserService.Get(userDTO, Data);
        if (user == null || user.Level < 1) return false;
        else return true;
    }
}

public string AvatarPath(UserDTO userDTO)
{
    using (var Data = DataFactory.Get())
    {
        var user = GetUserService.Get(userDTO, Data);
        if (user == null) return null;
        else return ImageFolder + user.Login + "/";
    }
}

private bool CheckType(string type)
{
    foreach(var a in AllowedImgTypes)
        if (a == type.ToLower()) return true;
    return false;
}

public bool CheckFile(FileDTO fileDTO)
{
    if (fileDTO.Size > MaxSize) return false;
    return CheckType(fileDTO.Type);
}

public bool CanDelete(UserDTO userDTO)
{
    return CanLoad(userDTO);
}

public AccessCode Delete(string path, UserDTO user)
{
    if (!CanDelete(user)) return AccessCode.NoPremision;
    string realPath = "wwwroot" + path;
    if (!File.Exists("wwwroot" + path)) return AccessCode.NotFound;
    try
    {
        File.Delete(realPath);
        return AccessCode.Succsess;
    }
    catch
    {
        return AccessCode.Error;
    }
}

public IEnumerable Get()
{
    var Images = new List();
```

```
            foreach (var image in Directory.GetFiles(ImageFolder))
            {
                var fileInfo = new FileInfo(image);
                if (!CheckFile(new FileDTO { Size = fileInfo.Length, Type =
fileInfo.Extension }))
                    continue;
                Images.Add(new ImageInfoDTO { Path =
image.Replace(@"wwwroot", ""), Size = fileInfo.Length });

            }
            return Images;
        }
    }
}
```

Файл IBlogService.cs

```
using Education.BLL.DTO.Forum;
using Education.BLL.DTO.Pages;
using Education.BLL.DTO.User;
using System;
using System.Collections.Generic;
using System.Text;

namespace Education.BLL.Services.PageServices.Interfaces
{
    public interface IBlogService
    {
        CreateResultDTO Create(NoteEditDTO noteEditDTO, UserDTO userDTO);
        AccessCode Delete(int id, UserDTO userDTO);
        AccessCode Update(NoteEditDTO noteEditDTO, UserDTO userDTO);
        BlogDTO Get(UserDTO userDTO, int page);
        (AccessCode, NoteDTO) Get(int id, UserDTO userDTO);
        bool CanCreate(UserDTO userDTO);
    }
}
```

Файл IPageService.cs

```
using Education.BLL.DTO.Forum;
using Education.BLL.DTO.Pages;
using Education.BLL.DTO.User;
using Education.BLL.Logic.Interfaces;
using System;
using System.Collections.Generic;
using System.Text;

namespace Education.BLL.Services.PageServices.Interfaces
{
    public interface IPageService
    {
        bool CanCreate(UserDTO userDTO);
        IPageMap Map { get; }
        (AccessCode, PageDTO) Get(int id, UserDTO userDTO);
        AccessCode Update(PageEditDTO pageEditDTO, UserDTO userDTO);
        AccessCode Delete(int id, UserDTO userDTO);
        CreateResultDTO Create(PageEditDTO pageEditDTO, UserDTO userDTO);
    }
}
```

Файл BlogService.cs

```
using Education.BLL.DTO.Forum;
using Education.BLL.DTO.Pages;
using Education.BLL.DTO.User;
using Education.BLL.Logic.Interfaces;
```

```
using Education.BLL.Services.PageServices.Interfaces;
using Education.DAL.Entities;
using Education.DAL.Entities.Pages;
using Education.DAL.Interfaces;
using System;
using System.Collections.Generic;
using System.Linq;

namespace Education.BLL.Services.PageServices
{
    public class BlogService : IBlogService
    {
        private const int NotesPerPage = 10;
        private INoteRules NoteRules;
        private IUOWFactory DataFactory;
        private IGetUserDTO GetUserService;
        private IDTOHelper DTOHelper;

        public BlogService(INoteRules noteRules, IUOWFactory dataFactory,
IGetUserDTO getUserService, IDTOHelper dtoHelper)
        {
            NoteRules = noteRules;
            DataFactory = dataFactory;
            GetUserService = getUserService;
            DTOHelper = dtoHelper;
        }

        private NoteDTO GetDTO(Note note, User user)
        {
            return new NoteDTO
            {
                Id = note.Id,
                Preview = note.Preview,
                Name = note.Name,
                Text = note.Text,
                Time = note.Time,
                Published = note.Published,
                Owner = DTOHelper.GetUser(note.Owner),
                Access = new DTO.AccessDTO
                {
                    CanDelete = NoteRules.CanDelete(user, note),
                    CanRead = NoteRules.CanRead(user, note),
                    CanUpdate = NoteRules.CanEdit(user, note)
                }
            };
        }

        private void EditNote(Note note, NoteEditDTO noteEditDTO)
        {
            if (note == null) return;
            note.Name = noteEditDTO.Name;
            note.Preview = noteEditDTO.Preview;
            note.Published = noteEditDTO.Published;
            note.Text = noteEditDTO.Text;
            note.Time = DateTime.Now;
        }

        private IEnumerable Get(User user, IEnumerable notes)
        {
            var res = new List();
            foreach(var note in notes)
                if (NoteRules.CanRead(user, note))
                    res.Add(GetDTO(note, user));
```

```
            return res;
        }

        private IEnumerable GetNotes(int page, IUOW Data)
        {
            return Data.NoteRepository.Get().Skip((page - 1) * NotesPerPage)
            .Take(NotesPerPage);
        }

        public (AccessCode,NoteDTO) Get(int id, UserDTO userDTO)
        {
            using(var Data = DataFactory.Get())
            {
                var note = Data.NoteRepository.Get().FirstOrDefault(x => x.Id
== id);
                var user = GetUserService.Get(userDTO,Data);
                if (note == null) return (AccessCode.NotFound, null);
                if (!NoteRules.CanRead(user,note)) return
(AccessCode.NoPremision, null);
                return (AccessCode.Succsess, GetDTO(note, user));
            }
        }

        public BlogDTO Get(UserDTO userDTO, int page)
        {
            using (var Data = DataFactory.Get())
            {
                if (page < 1) page = 1;
                var user = GetUserService.Get(userDTO, Data);
                var notes = GetNotes(page, Data);

                return new BlogDTO {
                    Page = page,
                    Pages = Data.NoteRepository.Get().Count() / NotesPerPage
+ 1,
                    Notes = Get(user,notes),
                    CanCreate = NoteRules.CanCreate(user)
                };
            }
        }

        public AccessCode Update(NoteEditDTO noteEditDTO, UserDTO userDTO)
        {
            using (var Data = DataFactory.Get())
            {
                var note = Data.NoteRepository.Get().FirstOrDefault(x => x.Id
== noteEditDTO.Id);
                var user = GetUserService.Get(userDTO, Data);
                if (note == null) return AccessCode.NotFound;
                if (!NoteRules.CanEdit(user, note)) return
AccessCode.NoPremision;
                EditNote(note, noteEditDTO);
                Data.NoteRepository.Edited(note);
                Data.SaveChanges();
            }
            return AccessCode.Succsess;
        }

        public AccessCode Delete(int id, UserDTO userDTO)
        {
            using (var Data = DataFactory.Get())
            {
```

```
                var note = Data.NoteRepository.Get().FirstOrDefault(x => x.Id
== id);
                var user = GetUserService.Get(userDTO, Data);
                if (note == null) return AccessCode.NotFound;
                if (!NoteRules.CanEdit(user, note)) return
AccessCode.NoPremision;
                Data.NoteRepository.Delete(note);
                Data.SaveChanges();
                return AccessCode.Succsess;
            }
        }

        public CreateResultDTO Create(NoteEditDTO noteEditDTO, UserDTO
userDTO)
        {
            using (var Data = DataFactory.Get())
            {
                var user = GetUserService.Get(userDTO, Data);
                if (!NoteRules.CanCreate(user)) return
CreateResultDTO.NoPremision;
                var note = new Note { Owner = user };
                EditNote(note, noteEditDTO);
                Data.NoteRepository.Add(note);
                Data.SaveChanges();
                return new CreateResultDTO(note.Id, AccessCode.Succsess);
            }
        }

        public bool CanCreate(UserDTO userDTO)
        {
            using (var Data = DataFactory.Get())
                return NoteRules.CanCreate(GetUserService.Get(userDTO,
Data));
        }

    }
}
```

Файл PageService.cs

```
using Education.BLL.DTO.Forum;
using Education.BLL.DTO.Pages;
using Education.BLL.DTO.User;
using Education.BLL.Logic.Interfaces;
using Education.DAL.Interfaces;
using System.Linq;
using System.Collections.Generic;
using System.Text;
using Education.DAL.Entities.Pages;
using Education.DAL.Entities;
using Education.BLL.Services.PageServices.Interfaces;
using Education.BLL.Logic;

namespace Education.BLL.Services.PageServices
{
    public class PageService : IPageService
    {
        private IUOWFactory DataFactory;
        private IPageRules PageRules;
        private IDTOHelper DTOHelper;
        private IGetUserDTO GetUserService;

        public IPageMap Map { get; private set; }
```

```
        private void InitMap()
        {
            using (var Data = DataFactory.Get())
                Map = new PageMap(Data.PageRepository.Get());
        }

        public PageService(IPageRules pageRules, IUOWFactory dataFactory,
IGetUserDTO getUserService, IDTOHelper dtoHelper)
        {
            PageRules = pageRules;
            DataFactory = dataFactory;
            GetUserService = getUserService;
            DTOHelper = dtoHelper;
            //---------------------------
            InitMap();
        }

        private PagePreviewDTO GetPreviewDTO(Page page)
        {
            return new PagePreviewDTO { Id = page.Id, Name = page.Name };
        }

        private void EditPage(Page page, PageEditDTO pageEditDTO, IUOW Data)
        {
            if (page == null || pageEditDTO == null) return;
            page.Name = pageEditDTO.Name;
            if (pageEditDTO.ParentId >= 0)
                page.ParentPage = Data.PageRepository.Get().FirstOrDefault(x
=> x.Id == pageEditDTO.ParentId);
            else page.ParentPage = null;
            page.Published = pageEditDTO.Published;
            page.Text = pageEditDTO.Text;
        }

        private PageDTO GetDTO(Page page, User user)
        {
            var childPages = new List();
            foreach (var cp in page.ChildPages)
                if(PageRules.CanRead(user, cp))
                    childPages.Add(GetPreviewDTO(cp));

            var res = new PageDTO
            {
                ChildPages = childPages,
                Id = page.Id,
                Name = page.Name,
                Published = page.Published,
                Text = page.Text,
                Access = new DTO.AccessDTO
                {
                    CanDelete = PageRules.CanDelete(user, page),
                    CanRead = PageRules.CanRead(user, page),
                    CanUpdate = PageRules.CanEdit(user, page)
                }
            };

            if (page.ParentPage != null)
            {
                res.ParentId = page.ParentPage.Id;
                res.ParentName = page.ParentPage.Name;
            }
            else res.ParentId = -1;
```

```
            return res;
        }

        private IEnumerable GetMap(Page page)
        {
            var res = new List();
            foreach (var ch in page.ChildPages)
            {
                res.Add(new PageInfo { Id = ch.Id, Name = ch.Name, Childs =
GetMap(ch) });
            }
            return res;
        }

        private bool CheckCycle(Page parent, Page child)
        {
            if (parent == child) return false;
            Page curPage = parent.ParentPage;
            while (true)
            {
                if (curPage == null) return false;
                if (curPage == parent) return false; // Чтобы не зацикл.
                if (curPage == child) return true;
                curPage = curPage.ParentPage;
            }

        }

        public (AccessCode,PageDTO) Get(int id, UserDTO userDTO)
        {
            using (var Data = DataFactory.Get())
            {
                var page = Data.PageRepository.Get().FirstOrDefault(X => X.Id
== id);
                var user = GetUserService.Get(userDTO, Data);
                if (page == null) return (AccessCode.NotFound, null);
                if (!PageRules.CanRead(user, page)) return
(AccessCode.NoPremision, null);
                return (AccessCode.Succsess,GetDTO(page, user));
            }
        }

        public AccessCode Update(PageEditDTO pageEditDTO, UserDTO userDTO)
        {
            using(var Data = DataFactory.Get())
            {
                var page = Data.PageRepository.Get().FirstOrDefault(X => X.Id
== pageEditDTO.Id);
                var user = GetUserService.Get(userDTO, Data);
                if (page == null) return AccessCode.NotFound;
                if (!PageRules.CanEdit(user, page)) return
AccessCode.NoPremision;
                //-------------------------
                if(pageEditDTO.ParentId > 0) // Проверка на цикличность
                {
                    var parent = Data.PageRepository.Get().FirstOrDefault(x
=> x.Id == pageEditDTO.ParentId);
                    if (parent != null && CheckCycle(parent, page)) return
AccessCode.Error;
                }
                //-------------------------
                EditPage(page, pageEditDTO, Data);
                Data.PageRepository.Edited(page);
```

```
                Data.SaveChanges();
                Map.Update(page);
                return AccessCode.Succsess;
            }
        }

        public AccessCode Delete(int id, UserDTO userDTO)
        {
            using (var Data = DataFactory.Get())
            {
                var page = Data.PageRepository.Get().FirstOrDefault(X => X.Id
== id);
                var user = GetUserService.Get(userDTO, Data);
                if (page == null) return AccessCode.NotFound;
                if (!PageRules.CanDelete(user, page)) return
AccessCode.NoPremision;
                Data.PageRepository.Delete(page);
                Data.SaveChanges();
                Map.Delete(id);
                return AccessCode.Succsess;
            }
        }

        public CreateResultDTO Create(PageEditDTO pageEditDTO, UserDTO
userDTO)
        {
            using (var Data = DataFactory.Get())
            {
                var user = GetUserService.Get(userDTO, Data);
                if (!PageRules.CanCreate(user)) return
CreateResultDTO.NoPremision;
                var page = new Page();
                EditPage(page, pageEditDTO, Data);
                Data.PageRepository.Add(page);
                //Добавить проверку на цикл
                Data.SaveChanges();
                Map.Add(page);
                return new CreateResultDTO(page.Id, AccessCode.Succsess);
            }
        }

        public bool CanCreate(UserDTO userDTO)
        {
            using (var Data = DataFactory.Get())
                return PageRules.CanCreate(GetUserService.Get(userDTO,
Data));
        }
    }
}
```

Файл AuthKeyService.cs

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Security.Claims;
using System.Text;
using Education.BLL.DTO.User;
using Education.BLL.Services.UserServices.Interfaces;
using Education.DAL.Entities;
using Education.BLL.Logic;
using Education.DAL.Interfaces;
```

```
namespace Education.BLL.Services.UserServices.Auth
{
    public class AuthKeyService : IAuthKeyService
    {
        private IMessenger Messager;
        private IKeyGenerator KeyGenerator;

        public AuthKeyService(IKeyGenerator kg, IMessenger msgr)
        {
            Messager = msgr;
            KeyGenerator = kg;
        }

        private void RemoveKey(Contact contact, IUOW Data)
        {
            Data.AuthKeyRepository.Delete(contact.Key);
            Data.SaveChanges();
        }

        public KeyStatus Check(Contact contact, string Key, IUOW Data)
        {
            if (contact == null) throw new
KeyException(KeyError.ContactNotFound);
            if (contact.Key == null) throw new
KeyException(KeyError.KeyNotFound);
            if (contact.Key.EndTime < DateTime.Now)
            {
                RemoveKey(contact, Data);
                return KeyStatus.KeyTimeEnded;
            }
            if (contact.Key.Value == Key)
            {
                RemoveKey(contact, Data);
                return KeyStatus.Success;
            }
            return KeyStatus.Fail;
        }

        public DateTime Generate(Contact contact, IUOW Data)
        {
            if (contact == null) throw new
KeyException(KeyError.ContactNotFound);
            if (contact.Key != null)
            {
                if (contact.Key.EndTime > DateTime.Now)
                {
                    return contact.Key.EndTime;
                }
            }
            contact.Key = KeyGenerator.Get();
            Data.ContactRepository.Edited(contact);
            Data.SaveChanges();
            Messager.Send(contact.Value, "Your key: " + contact.Key.Value);
            return contact.Key.EndTime;
        }
    }
}
```

Файл ClaimService.cs

```
using Education.BLL.DTO.User;
using Education.BLL.Services.UserServices.Interfaces;
using Education.DAL.Entities;
using Education.DAL.Interfaces;
using System;
```

```
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Security.Claims;
using System.Text;

namespace Education.BLL.Services.UserServices.Auth
{
    public class ClaimService : IClaimService
    {
        private IUOWFactory DataFactory;

        class ClaimNames
        {
            public static string UserID = "UserId";
            public static string SessionID = "SessionID";
            public static string SessionValue = "SessionValue";
        }
        public ClaimService(IUOWFactory uowf)
        {
            DataFactory = uowf;
        }

        private UserClaim GenerateClaim(User user, LoginInfoDTO loginInfoDTO)
        {
            var now = DateTime.Now;
            var claim = new UserClaim {
                LoginBrowser = loginInfoDTO.Browser,
                User = user,
                LoginTime = DateTime.Now,
                LoginIp = loginInfoDTO.IP
            };
            claim.Value = user.Login + "-" + now.ToString();
            return claim;
        }

        private UserDTO Generate(IEnumerable claims)
        {
            var idClaim = claims.FirstOrDefault(x => x.Type ==
ClaimNames.UserID);
            var sIdClaim = claims.FirstOrDefault(x => x.Type ==
ClaimNames.SessionID);
            if (idClaim == null || sIdClaim == null) return null;
            return new UserDTO { Id = int.Parse(idClaim.Value), ClaimId =
int.Parse(sIdClaim.Value) };
        }

        public void RemoveAllClaims(User user, IUOW Data, params string[]
without)
        {
            if (user == null) return;
            var claims = Data.UserClaimRepository.Get().Where(x => x.User ==
user).ToList();
            foreach (var claim in without)
            {
                var res = claims.FirstOrDefault(x => x.Value == claim);
                if (res != null) claims.Remove(res);
            }

            Data.UserClaimRepository.Delete(claims);
            Data.SaveChanges();

        }
```

```
        //-------------------------------------------------
        public UserDTO GetUser(IEnumerable claims)
        {
            try
            {
                return Generate(claims);
            }
            catch
            {
                return null;
            }
        }
        //-------------------------------------------------
        public ClaimsIdentity Generate(User user, IUOW Data, LoginInfoDTO
loginInfoDTO)
        {
            if (user == null || loginInfoDTO == null) return null;
            var claim = GenerateClaim(user, loginInfoDTO);
            Data.UserClaimRepository.Add(claim);
            Data.SaveChanges();
            var claims = new List
            {
                new Claim(ClaimNames.UserID, user.Id.ToString()),
                new Claim(ClaimNames.SessionID, claim.Id.ToString()),
                new Claim(ClaimNames.SessionValue, claim.Value.ToString()),
            };
            return new ClaimsIdentity(claims, "ApplicationCookie");

        }

        public void Logout(IEnumerable claims)
        {
            if (claims == null) return;
            using (var Data = DataFactory.Get())
            {
                var a = claims.FirstOrDefault(x => x.Type ==
ClaimNames.SessionValue);
                var claim = Data.UserClaimRepository.Get().FirstOrDefault(x
=> x.Value == a.Value);
                if (claim == null) return;
                Data.UserClaimRepository.Delete(claim);
                Data.SaveChanges();
            }
        }

        public IEnumerable GetInfo(User user, IUOW Data)
        {
            if (user == null) return null;

            var claims = Data.UserClaimRepository.Get().Where(x => x.User ==
user);
            var result = new List();
            foreach (var claim in claims)
                result.Add(new ClaimInfoDTO { Browser = claim.LoginBrowser,
Ip = claim.LoginIp, LoginTime = claim.LoginTime });
            return result;

        }

    }
}
```

Файл RegValidator.cs

```
using System;
using System.Linq;
using System.Text.RegularExpressions;
using Education.BLL.DTO.User;
using Education.BLL.Services.UserServices.Interfaces;
using Education.DAL.Interfaces;
using Education.BLL.Logic;

namespace Education.BLL.Services.UserServices.Auth
{
    public class RegValidator : IRegValidator
    {
        private Regex PhoneRegex = new
Regex(regRegularExpressions.PhoneRegex);
        private Regex EmailRegex = new
Regex(regRegularExpressions.EmailRegex);
        private Regex LoginRegex = new
Regex(regRegularExpressions.LoginRegex);
        private Regex PassRegex = new
Regex(regRegularExpressions.PasswordRegex);
        private Regex FullNameRegex = new
Regex(regRegularExpressions.FullNameRegex);

        public RegValidator()
        {

        }

        public CheckResult checkEmail(string email, IUOW Data)
        {
            if (email == null || !EmailRegex.IsMatch(email)) return
CheckResult.WrongValue;
            if (Data.UserRepository.Get().FirstOrDefault(x => x.Email.Value
== email) != null)
                return CheckResult.AlreadyExists;
            return CheckResult.Ok;
        }

        public CheckResult checkPhone(string phone, IUOW Data)
        {
            if (phone == null || !PhoneRegex.IsMatch(phone)) return
CheckResult.WrongValue;
            if (Data.UserRepository.Get()
                .FirstOrDefault(x => x.Phone.Value == phone) != null)
                return CheckResult.AlreadyExists;
            return CheckResult.Ok;
        }

        public CheckResult checkLogin(string login, IUOW Data)
        {
            if (login == null || !LoginRegex.IsMatch(login))
                return CheckResult.WrongValue;
            if (Data.UserRepository.Get().FirstOrDefault(x => x.Login ==
login) != null)
                return CheckResult.AlreadyExists;
            return CheckResult.Ok;
        }

        public CheckResult checkPassword(string password)
        {
            if (password == null || !PassRegex.IsMatch(password))
                return CheckResult.WrongValue;
            return CheckResult.Ok;
```

```
        }

        public CheckResult checkFullName(string fullname)
        {
            if (fullname == null || !FullNameRegex.IsMatch(fullname))
                return CheckResult.WrongValue;
            return CheckResult.Ok;
        }

        public RegisterResult Check(RegUserInfo userDTO, IUOW Data)
        {
            bool emailExists = !String.IsNullOrEmpty(userDTO.Email);
            bool phoneExists = !String.IsNullOrEmpty(userDTO.Phone);

            if (!emailExists && !phoneExists) return
RegisterResult.NeedContact;

            var Check = checkLogin(userDTO.Login, Data);
            if (Check == CheckResult.AlreadyExists) return
RegisterResult.LoginAlreadyExists;
            else if (Check == CheckResult.WrongValue) return
RegisterResult.WrongLogin;

            Check = checkPassword(userDTO.Password);
            if (Check == CheckResult.WrongValue) return
RegisterResult.WrongPassword;

            if (phoneExists)
            {
                Check = checkPhone(userDTO.Phone, Data);
                if (Check == CheckResult.AlreadyExists) return
RegisterResult.PhoneAlreadyExists;
                else if (Check == CheckResult.WrongValue) return
RegisterResult.WrongPhone;
            }

            if (emailExists)
            {
                Check = checkEmail(userDTO.Email, Data);
                if (Check == CheckResult.AlreadyExists) return
RegisterResult.EmailAlreadyExists;
                else if (Check == CheckResult.WrongValue) return
RegisterResult.WrongEmail;
            }

            if (checkFullName(userDTO.Name) != CheckResult.Ok) return
RegisterResult.WrongFullName;

            return RegisterResult.Confirm;

        }
    }
}
```

Файл SHA256Hasher.cs

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text;
using System.Security.Cryptography;

namespace Education.BLL.Services.UserServices.Auth
{
```

```
    public class SHA256Hasher : Interfaces.IPassHasher
    {
        SHA256 SHA256 = new SHA256Managed();

        public string Get(string input)
        {
            if (input == null) return null;
            var bytes = Encoding.Default.GetBytes(input);
            var hashbytes = SHA256.ComputeHash(bytes);
            return BitConverter.ToString(hashbytes).ToLower();
        }
    }
}
```

Файл UserAuthService.cs

```
using System;
using System.Collections.Generic;
using System.Security.Claims;
using System.Linq;
using Education.BLL.DTO.User;
using Education.BLL.Logic;
using Education.BLL.DTO;
using Education.DAL.Entities;
using Education.DAL.Interfaces;
using Education.BLL.Services.UserServices.Interfaces;
using Education.BLL.Logic.Interfaces;

namespace Education.BLL.Services.UserServices.Auth
{
    public class UserAuthService : IUserService
    {
        private IUOWFactory DataFactory;
        private IAuthKeyService EmailAuthService;
        private IAuthKeyService PhoneAuthService;
        private IPassHasher PassHasher;
        private IRegValidator RegValidator;
        private IKeyGenerator KeyGenerator;
        private IGetUserDTO GetUserService;
        public IClaimService ClaimService { get; private set; }

        public UserAuthService(IUOWFactory iuowf,
            IAuthKeyService phoneAuthService,
            IAuthKeyService emailAuthService,
            IPassHasher passHasher,
            IRegValidator regValidator,
            IClaimService claimService,
            IKeyGenerator keyGenerator,
            IGetUserDTO getUserDTO)
        {
            DataFactory = iuowf;
            EmailAuthService = emailAuthService;
            PhoneAuthService = phoneAuthService;
            PassHasher = passHasher;
            RegValidator = regValidator;
            ClaimService = claimService;
            KeyGenerator = keyGenerator;
            GetUserService = getUserDTO;

        }

        private User GetUser(string login, string pass, IUOW Data)
        {
            return Data.UserRepository.Get().FirstOrDefault(
                x =>
```

```
                x.Password == PassHasher.Get(pass)
                &&
                x.Login == login.ToLower()
                ||
                x.Phone != null && x.Phone.Confirmed && x.Phone.Value ==
login.ToLower()
                ||
                x.Email != null && x.Email.Confirmed && x.Email.Value ==
login.ToLower());
        }



        private AuthResult KeyLogin(User user, LoginInfoDTO loginInfoDTO,
IUOW Data)
        {
            KeyStatus keyStatus;
            if (user.authType == AuthType.Email) keyStatus =
EmailAuthService.Check(user.Email, loginInfoDTO.Key, Data);
            else if (user.authType == AuthType.Phone) keyStatus =
PhoneAuthService.Check(user.Phone, loginInfoDTO.Key, Data);
            else throw new UserAuthException(AuthError.AuthTypeNotFound);
            if (keyStatus == KeyStatus.Success)
                return new AuthResult { Status = AuthStatus.Succsess,
Identity = ClaimService.Generate(user, Data, loginInfoDTO) };
            else if (keyStatus == KeyStatus.KeyTimeEnded) return new
AuthResult { Status = AuthStatus.NeedNewKey };
            return new AuthResult { Status = AuthStatus.WrongKey };

        }

        private AuthResult SendKey(User user, IUOW Data)
        {
            DateTime keyTime;
            if (user.authType == AuthType.Email) keyTime =
EmailAuthService.Generate(user.Email, Data);
            else if (user.authType == AuthType.Phone) keyTime =
PhoneAuthService.Generate(user.Phone, Data);
            else throw new UserAuthException(AuthError.AuthTypeNotFound);
            return new AuthResult { Status = AuthStatus.KeySent, authType =
user.authType, KeyTime = keyTime };
        }


        public AuthResult Login(LoginInfoDTO loginInfoDTO)
        {
            using (var Data = DataFactory.Get())
            {
                User user = GetUser(loginInfoDTO.Login,
loginInfoDTO.Password, Data);
                if (user == null) return new AuthResult { Status =
AuthStatus.UserNotFound };
                if (user.Ban != null)
                {
                    if (user.Ban.EndTime < DateTime.Now) return new
AuthResult { Status = AuthStatus.UserBanned, KeyTime = user.Ban.EndTime,
Comment = user.Ban.Reason };
                    else
                    {
                        Data.BanRepository.Delete(user.Ban);
                        Data.SaveChanges();
                    }
                }
```

```
                if (user.authType == AuthType.Simple)
                    return new AuthResult { Status = AuthStatus.Succsess,
Identity = ClaimService.Generate(user, Data, loginInfoDTO) };
                else if (!String.IsNullOrEmpty(loginInfoDTO.Key)) return
KeyLogin(user, loginInfoDTO, Data);
                else return SendKey(user, Data);
            }
        }

        public void Logout(IEnumerable claims)
        {
            ClaimService.Logout(claims);
        }

        public RegisterResult Register(RegUserInfo userInfo)
        {
            using (var Data = DataFactory.Get())
            {
                var info = new UserInfo { FullName = userInfo.Name };
                var check = RegValidator.Check(userInfo, Data);
                if (check != RegisterResult.Confirm) return check;
                var newUser = new User
                {
                    Info = info,
                    Login = userInfo.Login.ToLower(),
                    Password = PassHasher.Get(userInfo.Password)
                };

                Contact email = null;
                Contact phone = null;

                if (!String.IsNullOrEmpty(userInfo.Email)) {
                    email = new Contact { Value = userInfo.Email.ToLower(),
Confirmed = false };
                }
                if (!String.IsNullOrEmpty(userInfo.Phone))
                {
                    phone = new Contact { Value = userInfo.Phone, Confirmed =
false };
                }

                newUser.Phone = phone;
                newUser.Email = email;
                Data.UserRepository.Add(newUser);
                Data.SaveChanges();
            }
            return RegisterResult.Confirm;
        }

        public void ResetClaims(UserDTO userDTO)
        {
            using (var Data = DataFactory.Get())
            {

ClaimService.RemoveAllClaims(GetUserService.Get(userDTO,Data),Data);
            }
        }

    }
}
```

Файл ConfirmKeyService.cs

```
using System;
```

```
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Security.Claims;
using System.Text;
using Education.BLL.DTO.User;
using Education.BLL.Services.UserServices.Interfaces;
using Education.DAL.Entities;
using Education.BLL.Logic;
using Education.DAL.Interfaces;

namespace Education.BLL.Services.UserServices.Confirm
{
    public class ConfirmKeyService : IAuthKeyService
    {
        private IMessenger Messager;
        private IKeyGenerator KeyGenerator;

        public ConfirmKeyService(IKeyGenerator kg, IMessenger msgr)
        {
            Messager = msgr;
            KeyGenerator = kg;
        }

        private void RemoveKey(Contact contact, IUOW Data)
        {
            Data.AuthKeyRepository.Delete(contact.ConfirmKey);
            Data.SaveChanges();
        }

        public KeyStatus Check(Contact contact, string Key, IUOW Data)
        {
            if (contact == null) throw new
KeyException(KeyError.ContactNotFound);
            if (contact.ConfirmKey == null) return KeyStatus.Fail;
            if (contact.ConfirmKey.EndTime < DateTime.Now)
            {
                RemoveKey(contact, Data);
                return KeyStatus.KeyTimeEnded;
            }
            if (contact.ConfirmKey.Value == Key)
            {
                RemoveKey(contact, Data);
                return KeyStatus.Success;
            }
            return KeyStatus.Fail;
        }

        public DateTime Generate(Contact contact, IUOW Data)
        {
            if (contact == null) throw new
KeyException(KeyError.ContactNotFound);
            if (contact.ConfirmKey != null)
            {
                if (contact.ConfirmKey.EndTime > DateTime.Now)
                {
                    return contact.ConfirmKey.EndTime;
                }
            }
            contact.ConfirmKey = KeyGenerator.Get();
            Data.ContactRepository.Edited(contact);
            Data.SaveChanges();
```

```
            Messager.Send(contact.Value, "Your key: " +
contact.ConfirmKey.Value);
            return contact.ConfirmKey.EndTime;
        }
    }
}
```

Файл ConfirmService.cs

```
using Education.BLL.DTO;
using Education.BLL.Services.UserServices.Interfaces;
using Education.DAL.Entities;
using Education.DAL.Interfaces;
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text;

namespace Education.BLL.Services.UserServices.Confirm
{
    public class ConfirmService : IConfirmService
    {
        protected IAuthKeyService KeyService;

        public ConfirmService(IAuthKeyService keyService)
        {
            KeyService = keyService;
        }

        private void RemoveContact(Contact contact, IUOW Data)
        {
            Data.ContactRepository.Delete(contact);
            Data.SaveChanges();
        }

        private void ConfirmContact(Contact contact, IUOW Data)
        {
            contact.Confirmed = true;
            Data.ContactRepository.Edited(contact);
            Data.SaveChanges();
        }

        public ConfirmResult DoIfSuccsess(Contact contact, IUOW Data, Action
action = null, string key = null)
        {
            if (contact == null) return new ConfirmResult { Status =
ConfirmCode.ContactNotFound };
            if (String.IsNullOrEmpty(key))
            {
                var keytime = KeyService.Generate(contact, Data);
                return new ConfirmResult { Status = ConfirmCode.KeySend,
KeyTime = keytime };
            }
            var res = KeyService.Check(contact, key, Data);
            if (res == KeyStatus.Success)
            {
                action?.Invoke(contact, Data);
                return new ConfirmResult { Status = ConfirmCode.Success };
            }
            else if (res == KeyStatus.KeyTimeEnded) return new ConfirmResult
{ Status = ConfirmCode.NeedNewKey };
            return new ConfirmResult { Status = ConfirmCode.Fail };
        }
```

```
        public virtual ConfirmResult Confirm(Contact contact, IUOW Data,
string key = null)
        {
            if (contact == null) return new ConfirmResult { Status =
ConfirmCode.ContactNotFound };
            if (contact.Confirmed) return new ConfirmResult { Status =
ConfirmCode.AlreadyConfimed };
            return DoIfSuccsess(contact, Data, ConfirmContact, key);
        }

        public virtual ConfirmResult Remove(Contact contact, IUOW Data,
string key = null)
        {
            if (contact == null) return new ConfirmResult { Status =
ConfirmCode.ContactNotFound };
            if (!contact.Confirmed)
            {
                RemoveContact(contact, Data);
                return new ConfirmResult { Status = ConfirmCode.Success };
            }
            return DoIfSuccsess(contact, Data, RemoveContact, key);
        }
    }
}
```

Файл InitDBService.cs

```
using Education.BLL.Logic;
using Education.DAL.Interfaces;
using System.Linq;
using Education.DAL.Entities;
using Education.BLL.Services.UserServices.Auth;

namespace Education.BLL.Services.UserServices.Init
{
    public class InitDBService : Interfaces.IInitDBService
    {
        private IUOW Data;
        public InitDBService(IUOW data)
        {
            Data = data;
        }

        private void CheckAdmin(string login)
        {
            if (Data.UserRepository.Get().FirstOrDefault(x => x.Login ==
login || x.Level == 2) != null)
                throw new DBInitException("AdminAlreadyExists");
        }

        public void InitAdmin(string login, string password)
        {
            //-----------------------

            //-----------------------
            var hasher = new SHA256Hasher();
            CheckAdmin(login);
            Data.UserRepository.Add(new DAL.Entities.User
            {
                Login = login,
                Password = hasher.Get(password),
                Info = new DAL.Entities.UserInfo
                {
                    FullName = ""
```

```
                },
                Level = 2,
                Email = new Contact
                {
                    Value = "talerok@gmail.com",
                    Confirmed = true
                },
                Phone = new Contact
                {
                    Value = "+79995701322",
                    Confirmed = true
                },
                authType = AuthType.Phone
            });
        }
    }
}
```

Файл IAuthKeyService.cs

```
using Education.BLL.DTO.User;
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text;
using Education.DAL.Entities;
using Education.DAL.Interfaces;

namespace Education.BLL.Services.UserServices.Interfaces
{
    public enum KeyStatus
    {
        Success,
        Fail,
        KeyTimeEnded
    }


    public interface IAuthKeyService
    {
        DateTime Generate(Contact contact, IUOW Data);
        KeyStatus Check(Contact contact, string Key, IUOW Data);
    }
}
```

Файл IClaimService.cs

```
using Education.BLL.DTO.User;
using Education.DAL.Entities;
using Education.DAL.Interfaces;
using System;
using System.Collections.Generic;
using System.Security.Claims;
using System.Text;

namespace Education.BLL.Services.UserServices.Interfaces
{
    public interface IClaimService
    {
        ClaimsIdentity Generate(User user, IUOW Data, LoginInfoDTO
loginInfoDTO);
        IEnumerable GetInfo(User user, IUOW Data);
        void RemoveAllClaims(User user, IUOW Data, params string[] without);
        void Logout(IEnumerable claims);
        UserDTO GetUser(IEnumerable claims);
```

```
    }
}
```

Файл IConfirmService.cs

```
using Education.BLL.DTO;
using Education.DAL.Entities;
using Education.DAL.Interfaces;
using System;

namespace Education.BLL.Services.UserServices.Interfaces
{
    public enum ConfirmCode
    {
        Success,
        KeySend,
        Fail,
        NeedNewKey,
        UserNotFound,
        AlreadyConfimed,
        ContactNotFound,
        NeedContact
    };

    public interface IConfirmService
    {
        ConfirmResult Confirm(Contact contact, IUOW Data, string key = null);
        ConfirmResult Remove(Contact contact, IUOW Data, string key = null);
        ConfirmResult DoIfSuccsess(Contact contact, IUOW Data, Action action
= null, string key = null);
    }
}
```

Файл IGetUserService.cs

```
using Education.BLL.DTO.User;
using Education.DAL.Entities;
using Education.DAL.Interfaces;
using System;
using System.Collections.Generic;
using System.Text;

namespace Education.BLL.Services.UserServices.Interfaces
{
    interface IGetUserService
    {
        User Get(UserDTO userDTO, IUOW Data);
        User Get(string login, IUOW Data);
    }
}
```

Файл IInitDBService.cs

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text;

namespace Education.BLL.Services.UserServices.Interfaces
{
    public interface IInitDBService
    {
        void InitAdmin(string login, string password);
    }
}
```

Файл IKeyGenerator.cs

```
using Education.DAL.Entities;
using System;
using System.Collections.Generic;
using System.Text;

namespace Education.BLL.Services.UserServices.Interfaces
{
    public interface IKeyGenerator
    {
        Key Get();
    }
}
```

Файл IMessenger.cs

```
using System;
using System.Collections.Generic;
using System.Text;

namespace Education.BLL.Services.UserServices.Interfaces
{
    public interface IMessenger
    {
        void Send(string to, string text);
    }
}
```

Файл IPassHasher.cs

```
using System;
using System.Collections.Generic;
using System.Text;

namespace Education.BLL.Services.UserServices.Interfaces
{
    public interface IPassHasher
    {
        string Get(string input);
    }
}
```

Файл IProfileService.cs

```
using Education.BLL.DTO.User;
using Education.BLL.DTO;
using Education.DAL.Entities;
using System;
using System.Collections.Generic;
using System.Text;

namespace Education.BLL.Services.UserServices.Interfaces
{
    public enum SetContactCode
    {
        Succsess,
        UserNotFound,
        AlreadySet,
        AlreadyExists,
        WrongValue
    };

    public interface IProfileService
    {
        IClaimService ClaimService { get; }
        ConfirmResult ResetAuthType(UserDTO userDTO, string key = null);
```

```
        ConfirmResult SetAuthType(UserDTO userDTO, AuthType authType, string
key = null);
        ConfirmResult SetPassword(UserDTO userDTO, string oldpassword, string
newPassword, string key = null);
        UserProfileDTO GetUserProfile(UserDTO userDTO);
        ConfirmResult ConfirmEmail(UserDTO userDTO, string key = null);
        ConfirmResult ConfirmPhone(UserDTO userDTO, string key = null);
        ConfirmResult RemoveEmail(UserDTO userDTO, string key = null);
        ConfirmResult RemovePhone(UserDTO userDTO, string key = null);
        SetContactCode SetEmail(UserDTO userDTO, string email);
        SetContactCode SetPhone(UserDTO userDTO, string phone);
        ConfirmCode SetAvatar(UserDTO userDTO, string path);
    }
}
```

Файл IRegValidator.cs

```
using Education.BLL.DTO.User;
using Education.DAL.Interfaces;
using System;
using System.Collections.Generic;
using System.Text;

namespace Education.BLL.Services.UserServices.Interfaces
{
    public enum RegisterResult
    {
        Confirm,
        WrongLogin,
        WrongEmail,
        WrongPhone,
        WrongPassword,
        WrongFullName,
        LoginAlreadyExists,
        EmailAlreadyExists,
        PhoneAlreadyExists,
        NeedContact,
        InternalError

    };

    public enum CheckResult
    {
        Ok,
        AlreadyExists,
        WrongValue
    };

    public interface IRegValidator
    {
        CheckResult checkEmail(string email, IUOW Data);
        CheckResult checkPhone(string phone, IUOW Data);
        CheckResult checkLogin(string login, IUOW Data);
        CheckResult checkPassword(string password);
        CheckResult checkFullName(string fullname);
        RegisterResult Check(RegUserInfo userDTO, IUOW Data);
    }
}
```

Файл IRestorePasswordService.cs

```
using Education.BLL.DTO;
using System;
using System.Collections.Generic;
using System.Text;
```

```
namespace Education.BLL.Services.UserServices.Interfaces
{
    public interface IRestorePasswordService
    {
        RestoreResult Restore(string login, string password = null, string
key = null);
    }
}
```

Файл IUserService.cs

```
using System;
using System.Collections.Generic;
using System.Text;
using Education.BLL.DTO.User;
using System.Security.Claims;
using Education.BLL.DTO;

namespace Education.BLL.Services.UserServices.Interfaces
{
    public enum AuthStatus
    {
        Succsess,
        UserNotFound,
        UserBanned,
        WrongKey,
        NeedNewKey,
        KeySent,
        InternalError
    }

    public interface IUserService
    {
        IClaimService ClaimService { get; }
        AuthResult Login(LoginInfoDTO loginInfoDTO);
        RegisterResult Register(RegUserInfo user);
        void Logout(IEnumerable claims);
        void ResetClaims(UserDTO userDTO);

    }
}
```

Файл BigKeyGenerator.cs

```
using Education.BLL.Services.UserServices.Interfaces;
using Education.DAL.Entities;
using System;
using System.Collections.Generic;
using System.Text;

namespace Education.BLL.Services.UserServices.KeyGenerators
{
    public class BigKeyGenerator : IKeyGenerator
    {
        private static Random Random = new Random(05460546);
        private const short KeyLenght = 5;
        private string GeneratePart()
        {
            return Random.Next(10000, 99999).ToString();
        }

        private string GenerateKey()
        {
            string key = "";
```

```
            for(int i = 0; i < KeyLenght; i++)
            {
                if (i != 0) key += "-";
                key += GeneratePart();
            }
            return key;
        }

        public Key Get()
        {
            return new Key
            {
                Value = GenerateKey(),
                EndTime = DateTime.Now.AddDays(1)
            };
        }
    }
}
```

Файл SmallKeyGenerator.cs

```
using System;
using System.Collections.Generic;
using System.Text;
using Education.BLL.Services.UserServices.Interfaces;
using Education.DAL.Entities;

namespace Education.BLL.Services.UserServices.KeyGenerators
{
    public class SmallKeyGenerator : IKeyGenerator
    {
        private static Random Random = new Random(05470547);

        public SmallKeyGenerator()
        {

        }

        public Key Get()
        {
            return new Key
            {
                Value = Random.Next(10000, 99999).ToString(),
                EndTime = DateTime.Now.AddMinutes(5)
            };
        }
    }
}
```

Файл EmailMessenger.cs

```
using System;
using System.Collections.Generic;
using System.Text;

namespace Education.BLL.Services.UserServices.Messengers
{
    public class EmailMessenger : Interfaces.IMessenger
    {
        public void Send(string to, string text)
        {
            int i = 1;
        }
    }
}
```

Файл SmsMessenger.cs

```
using System;
using System.Collections.Generic;
using System.Text;

namespace Education.BLL.Services.UserServices.Messagers
{
    public class SmsMessenger : Interfaces.IMessenger
    {
        public void Send(string to, string text)
        {
            int i = 1;
        }
    }
}
```

Файл ProfileService.cs

```
using Education.BLL.DTO.User;
using Education.BLL.DTO;
using Education.BLL.Services.UserServices.Interfaces;
using Education.DAL.Entities;
using Education.DAL.Interfaces;
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Reflection;
using Education.BLL.Services.ImageManager.Interfaces;
using Education.BLL.Logic.Interfaces;

namespace Education.BLL.Services.UserServices.Profile
{
    public class ProfileService : IProfileService
    {
        private IPassHasher PassHasher;

        private static string EmailPropertyName = "Email";
        private static string PhonePropertyName = "Phone";

        private IUOWFactory DataFactory;
        private IRegValidator RegValidator;

        private IConfirmService EmailService;
        private IConfirmService PhoneService;
        private IGetUserDTO GetUserService;

        public IClaimService ClaimService { get; private set; }

        public ProfileService(
            IUOWFactory uowf,
            IRegValidator regValidator,
            IConfirmService emailCS,
            IConfirmService phoneCS,
            IPassHasher passHasher,
            IClaimService claimService,
            IGetUserDTO getUserDTO)
        {
            DataFactory = uowf;
            RegValidator = regValidator;
            EmailService = emailCS;
            PhoneService = phoneCS;
            PassHasher = passHasher;
            ClaimService = claimService;
            GetUserService = getUserDTO;
```

```
        }

        private void SetPassword(User user, string password, IUOW Data)
        {
            user.Password = PassHasher.Get(password);
            Data.UserRepository.Edited(user);
            Data.SaveChanges();
        }

        #region ContactLogic

        private ConfirmResult ConfirmContact(UserDTO userDTO, string
propertyName, Func func, IUOW Data, string key = null)
        {
            var User = GetUserService.Get(userDTO, Data);
            if (User == null) return new ConfirmResult { Status =
ConfirmCode.UserNotFound };
            var Property = User.GetType().GetProperty(propertyName);
            var Contact = Property.GetValue(User) as Contact;
            if (Contact == null) return new ConfirmResult { Status =
ConfirmCode.ContactNotFound };
            return func(Contact, Data, key);
        }

        private ConfirmResult RemoveContact(User User, string propertyName,
Func func, AuthType checkType, IUOW Data, string key = null)
        {
            if (User == null) return new ConfirmResult { Status =
ConfirmCode.UserNotFound };
            var Property = User.GetType().GetProperty(propertyName);
            var Contact = Property.GetValue(User) as Contact;
            if (Contact == null) return new ConfirmResult { Status =
ConfirmCode.ContactNotFound };
            var res = func(Contact, Data, key);
            if (res.Status == ConfirmCode.Success && User.authType ==
checkType)
            {
                Property.SetValue(User, null);
                User.authType = AuthType.Simple;
                Data.UserRepository.Edited(User);
                Data.SaveChanges();
            }
            return res;
        }

        private SetContactCode SetContact(UserDTO userDTO, string value,
string propertyName, Func checkFunc, IUOW Data)
        {
            var User = GetUserService.Get(userDTO, Data);
            if (User == null) return SetContactCode.UserNotFound;
            if (User.GetType().GetProperty(propertyName).GetValue(User) !=
null) return SetContactCode.AlreadySet;
            var check = checkFunc(value, Data);
            if (check == CheckResult.AlreadyExists)
                return SetContactCode.AlreadyExists;
            else if (check == CheckResult.WrongValue)
                return SetContactCode.WrongValue;

            User.GetType().GetProperty(propertyName).SetValue(User, new
Contact { Confirmed = false, Value = value });
            Data.UserRepository.Edited(User);
            Data.SaveChanges();
            return SetContactCode.Succsess;
```

```
        }

        #endregion

        #region ContactServices

        public ConfirmResult ConfirmEmail(UserDTO userDTO, string key = null)
        {
            using (var Data = DataFactory.Get())
            {
                return ConfirmContact(userDTO, EmailPropertyName,
EmailService.Confirm, Data, key);
            }
        }

        public ConfirmResult ConfirmPhone(UserDTO userDTO, string key = null)
        {
            using (var Data = DataFactory.Get())
            {
                return ConfirmContact(userDTO, PhonePropertyName,
PhoneService.Confirm, Data, key);
            }
        }

        public ConfirmResult RemoveEmail(UserDTO userDTO, string key = null)
        {
            using (var Data = DataFactory.Get())
            {
                var User = GetUserService.Get(userDTO, Data);
                if (User == null) return new ConfirmResult { Status =
ConfirmCode.UserNotFound };
                return RemoveContact(User, EmailPropertyName,
EmailService.Remove, AuthType.Email, Data, key);
            }
        }

        public ConfirmResult RemovePhone(UserDTO userDTO, string key = null)
        {
            using (var Data = DataFactory.Get())
            {
                var User = GetUserService.Get(userDTO, Data);
                if (User == null) return new ConfirmResult { Status =
ConfirmCode.UserNotFound };
                return RemoveContact(User, PhonePropertyName,
PhoneService.Remove, AuthType.Phone, Data, key);
            }
        }

        public SetContactCode SetEmail(UserDTO userDTO, string email)
        {
            using (var Data = DataFactory.Get())
            {
                return SetContact(userDTO, email, EmailPropertyName,
RegValidator.checkEmail,Data);
            }
        }

        public SetContactCode SetPhone(UserDTO userDTO, string phone)
        {
            using (var Data = DataFactory.Get())
            {
                return SetContact(userDTO, phone, PhonePropertyName,
RegValidator.checkPhone, Data);
```

```
            }
        }

        #endregion

        public UserProfileDTO GetUserProfile(UserDTO userDTO)
        {
            using (var Data = DataFactory.Get())
            {
                var user = GetUserService.Get(userDTO, Data);
                if (user == null) return null;
                return new UserProfileDTO
                {
                    Avatar = user?.Info?.Avatar,
                    authType = user.authType,
                    email = user.Email?.Value,
                    emailConfirm = user.Email?.Confirmed,
                    phone = user.Phone?.Value,
                    phoneConfirm = user.Phone?.Confirmed,
                    name = user.Info?.FullName,
                    Claims = ClaimService.GetInfo(user, Data)
                };
            }
        }

        public ConfirmResult ResetAuthType(UserDTO userDTO, string key =
null)
        {
            using (var Data = DataFactory.Get())
            {
                var User = GetUserService.Get(userDTO, Data);
                ConfirmResult res;
                if (User == null) return new ConfirmResult { Status =
ConfirmCode.UserNotFound };
                if (User.authType == AuthType.Simple) return new
ConfirmResult { Status = ConfirmCode.Success };

                else if (User.authType == AuthType.Email) res =
EmailService.DoIfSuccsess(User.Email, Data, null, key);
                else if (User.authType == AuthType.Phone) res =
PhoneService.DoIfSuccsess(User.Phone, Data, null, key);
                else return new ConfirmResult { Status =
ConfirmCode.ContactNotFound };

                if (res.Status == ConfirmCode.Success)
                {
                    User.authType = AuthType.Simple;
                    Data.UserRepository.Edited(User);
                    Data.SaveChanges();
                    return new ConfirmResult { Status = ConfirmCode.Success
};
                }
                return res;
            }
        }

        public ConfirmResult SetAuthType(UserDTO userDTO, AuthType authType,
string key = null)
        {
            using (var Data = DataFactory.Get())
            {
                ConfirmResult res;
                var User = GetUserService.Get(userDTO, Data);
```

```
                if (User == null) return new ConfirmResult { Status =
ConfirmCode.UserNotFound };
                if (User.authType != AuthType.Simple) return new
ConfirmResult { Status = ConfirmCode.NeedContact };
                if (authType == AuthType.Simple) return new ConfirmResult {
Status = ConfirmCode.Success };
                else if (authType == AuthType.Phone && User.Phone != null &&
User.Phone.Confirmed)
                {
                    res = PhoneService.DoIfSuccsess(User.Phone, Data, null,
key);
                }
                else if (authType == AuthType.Email && User.Email != null &&
User.Email.Confirmed)
                {
                    res = EmailService.DoIfSuccsess(User.Email, Data, null,
key);
                }
                else return new ConfirmResult { Status =
ConfirmCode.ContactNotFound };

                if (res.Status == ConfirmCode.Success)
                {
                    User.authType = authType;
                    Data.UserRepository.Edited(User);
                    Data.SaveChanges();
                }
                return res;
            }
        }

        public ConfirmCode SetAvatar(UserDTO userDTO, string path)
        {
            using (var Data = DataFactory.Get())
            {
                var user = GetUserService.Get(userDTO, Data);
                if (user == null) return ConfirmCode.UserNotFound;
                if (user.Info == null) return ConfirmCode.Fail;
                user.Info.Avatar = path;
                Data.UserInfoRepository.Edited(user.Info);
                Data.SaveChanges();
                return ConfirmCode.Success;
            }
        }

        public ConfirmResult SetPassword(UserDTO userDTO, string oldpassword,
string newPassword, string key = null)
        {
            using (var Data = DataFactory.Get())
            {
                var User = GetUserService.Get(userDTO, Data);
                ConfirmResult res;
                if (User == null || PassHasher.Get(oldpassword) !=
User.Password) return new ConfirmResult { Status = ConfirmCode.UserNotFound
};
                if (User.authType == AuthType.Simple)
                {
                    SetPassword(User, newPassword,Data);
                    return new ConfirmResult { Status = ConfirmCode.Success
};
                }
```

```
                else if (User.authType == AuthType.Email) res =
EmailService.DoIfSuccsess(User.Email, Data, null, key);
                else if (User.authType == AuthType.Phone) res =
PhoneService.DoIfSuccsess(User.Phone, Data, null, key);
                else return new ConfirmResult { Status =
ConfirmCode.ContactNotFound };

                if (res.Status == ConfirmCode.Success)
                {
                    SetPassword(User, newPassword,Data);
                    return new ConfirmResult { Status = ConfirmCode.Success
};
                }
                return res;
            }
        }

    }
}
```

Файл RestorePasswordService.cs

```
using Education.BLL.DTO;
using Education.BLL.Services.UserServices.Confirm;
using Education.BLL.Services.UserServices.Interfaces;
using Education.DAL.Entities;
using Education.DAL.Interfaces;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace Education.BLL.Services.UserServices.Restore
{

    public class RestorePasswordService : IRestorePasswordService
    {
        private IAuthKeyService emailService;
        private IAuthKeyService phoneService;
        private IUOWFactory DataFactory;
        private IRegValidator regValidator;
        private IPassHasher passHasher;
        public RestorePasswordService(IUOWFactory uowf, IAuthKeyService
emailKeyService, IAuthKeyService phoneKeyService, IRegValidator reg,
IPassHasher pHasher)
        {
            DataFactory = uowf;
            emailService = emailKeyService;
            phoneService = phoneKeyService;
            regValidator = reg;
            passHasher = pHasher;
        }

        private User GetUser(string login, IUOW Data)
        {
            return Data.UserRepository.Get().FirstOrDefault(x =>
                x.Login == login.ToLower()
                ||
                x.Phone != null && x.Phone.Confirmed && x.Phone.Value ==
login.ToLower()
                ||
                x.Email != null && x.Email.Confirmed && x.Email.Value ==
login.ToLower());
        }
```

```
        public RestoreResult Restore(string login, string password = null,
string key = null)
        {
            using (var Data = DataFactory.Get())
            {
                var user = GetUser(login.ToLower(), Data);
                if (user == null) return new RestoreResult { Status =
RestoreCode.UserNotFound };
                if (key == null)
                {
                    if (user.Phone != null && user.Phone.Confirmed)
                        return new RestoreResult { Status =
RestoreCode.KeySent, SendTo = AuthType.Phone, KeyTime =
phoneService.Generate(user.Phone,Data) };
                    else if (user.Email != null && user.Email.Confirmed)
                        return new RestoreResult { Status =
RestoreCode.KeySent, SendTo = AuthType.Email, KeyTime =
emailService.Generate(user.Email,Data) };
                    else return new RestoreResult { Status =
RestoreCode.NeedContact };
                }

                KeyStatus status;

                if (user.Phone != null && user.Phone.Confirmed)
                    status = phoneService.Check(user.Phone, key, Data);
                else if (user.Email != null && user.Email.Confirmed)
                    status = emailService.Check(user.Email, key, Data);
                else return new RestoreResult { Status =
RestoreCode.NeedContact };

                if (status == KeyStatus.Success)
                {
                    if (String.IsNullOrEmpty(password) ||
regValidator.checkPassword(password) != CheckResult.Ok)
                        return new RestoreResult { Status =
RestoreCode.WrongPassword };

                    user.Password = passHasher.Get(password);
                    Data.UserRepository.Edited(user);
                    Data.SaveChanges();
                    return new RestoreResult { Status = RestoreCode.Succsess
};
                }
                else if (status == KeyStatus.KeyTimeEnded)
                    return new RestoreResult { Status =
RestoreCode.NeedNewKey };
                else return new RestoreResult { Status = RestoreCode.WorngKey
};
            }
        }
    }
}
```

Файл CaptchaInfo.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace Education.Captcha
{
    public static class CaptchaInfo
```

```
    {
        public const string SecretKey =
"6LfO_kwUAAAAANhsypTW1Joe2g9wA7f5I_DwF2EK";
        public const string Key = "6LfO_kwUAAAAAOlV53ySRy3E-L-caAqJ1er3Iumw";
    }
}
```

Файл ValidateRecaptchaAttribute.cs

```
using System.Linq;
using System.Collections.Generic;
using System.Net.Http;
using Education.Models;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Filters;
using Microsoft.AspNetCore.Mvc.ModelBinding;
using Microsoft.AspNetCore.Mvc.ViewFeatures;
using Newtonsoft.Json;
using Microsoft.Extensions.Primitives;

namespace Education.Captcha
{
    public class ValidateRecaptchaAttribute : ActionFilterAttribute
    {
        private readonly string _propertyName;
        private readonly string _secretKey;
        private readonly string _errorViewName;
        private readonly string _errorMessage;
        private const string GoogleRecaptchaUrl =
"https://www.google.com/recaptcha/api/siteverify?secret={0}&response={1}";

        public ValidateRecaptchaAttribute(string propertyName =
"RepatchaValue", string secretKey = CaptchaInfo.SecretKey, string
errorViewName = "Error", string errorMessage = "Invalid captcha!")
        {
            _propertyName = propertyName;
            _secretKey = secretKey;
            _errorViewName = errorViewName;
            _errorMessage = errorMessage;
        }

        public override void OnActionExecuting(ActionExecutingContext
context)
        {
            StringValues Headers;
            context.HttpContext.Request.Headers.TryGetValue("Captcha", out
Headers);
            if (Headers.Count != 0)
            {
                string Captcha = Headers[0];
                var captchaValidationResult = ValidateRecaptcha(Captcha,
_secretKey);
                if (captchaValidationResult.Success)
                {
                    base.OnActionExecuting(context);
                    return;
                }
            }

            SetInvalidResult(context);
            /*var model = context.ActionArguments.First().Value;
            var propertyInfo = model.GetType().GetProperty(_propertyName);
            if (propertyInfo != null)
            {
```

```
                var repatchaValue = propertyInfo.GetValue(model, null) as
string;
                var captchaValidationResult =
ValidateRecaptcha(repatchaValue, _secretKey);
                if (captchaValidationResult.Success)
                {
                    base.OnActionExecuting(context);
                    return;
                }
            }*/
        }

        private void SetInvalidResult(ActionExecutingContext context)
        {
            context.Result = new JsonResult(new ErrorMessage { Error =
_errorMessage } );
        }

        private static RecaptchaResponse ValidateRecaptcha(string
userEnteredCaptcha, string secretKey)
        {
            if (string.IsNullOrEmpty(userEnteredCaptcha))
            {
                return new RecaptchaResponse
                {
                    Success = false,
                    ErrorCodes = new[] { "missing-input-response" }
                };
            }

            using (var client = new HttpClient())
            {
                var result =
client.GetStringAsync(string.Format((string)GoogleRecaptchaUrl, secretKey,
userEnteredCaptcha)).Result;
                var captchaResponse = JsonConvert.DeserializeObject(result);
                return captchaResponse;
            }
        }

        public class RecaptchaResponse
        {
            [JsonProperty("success")]
            public bool Success { get; set; }

            [JsonProperty("challenge_ts")]
            public string ChallengeTs { get; set; }   // timestamp of the
challenge load (ISO format yyyy-MM-dd'T'HH:mm:ssZZ)

            [JsonProperty("hostname")]
            public string Hostname { get; set; }      // the hostname of the
site where the reCAPTCHA was solved

            [JsonProperty("error-codes")]
            public string[] ErrorCodes { get; set; }  // optional
        }
    }
}
```

Файл MenuViewComponent.cs

```
using Education.BLL.Services.PageServices.Interfaces;
using Microsoft.AspNetCore.Html;
using System;
```

```
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace Education.Components
{
    public class MenuViewComponent
    {
        private IPageService PageService { get; set; }
        public MenuViewComponent(IPageService pageService)
        {
            PageService = pageService;
        }

        public HtmlString Invoke()
        {
            string res = "";
            foreach(var page in PageService.Map.Get)
                if(page.Published)
                    res += String.Format(@"{1}", page.Id, page.Name);
            return new HtmlString(res);
        }

    }

}
```

Файл NavigationViewComponent.cs

```
using Education.BLL.DTO.Forum;
using Education.BLL.DTO.Pages;
using Microsoft.AspNetCore.Html;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace Education.Components
{
    public class NavigationViewComponent
    {
        private const string sep = @">>";

        private string MainPageNavigation()
        {
            return @"Главная страница";
        }

        private string GroupNavigation(int id, string name)
        {
            return MainPageNavigation() + sep + @"Группа " + name + "";
        }

        private string SectionNavigation(int id, string name, SectionRoute
route)
        {
            return GroupNavigation(route.GroupId, route.GroupName)
                + sep + @"Секция " + name + "";
        }

        private string ThemeNavigation(int id, string name, ThemeRoute route)
        {
            return SectionNavigation(route.SectionId, route.SectionName,
route.SectionRoute)
```

```
                + sep + @"Тема " + name + "";
        }

        private string PageNavigation(PageDTO pageDTO)
        {
            string res;
            if (pageDTO.ParentId > 0)
                res = @"" + pageDTO.ParentName + "" + sep;
            else res = "";
            res += @"" + pageDTO.Name + "";
            return res;
        }

        public HtmlString Invoke(Object DTO)
        {
            if (DTO is GroupDTO)
            {
                var _DTO = DTO as GroupDTO;
                return new HtmlString(GroupNavigation(_DTO.Id, _DTO.Name));
            }
            else if (DTO is SectionDTO)
            {
                var _DTO = DTO as SectionDTO;
                return new HtmlString(SectionNavigation(_DTO.Id, _DTO.Name,
_DTO.Route));
            }
            else if (DTO is ThemeDTO)
            {
                var _DTO = DTO as ThemeDTO;
                return new HtmlString(ThemeNavigation(_DTO.Id, _DTO.Name,
_DTO.Route));
            }
            else if(DTO is PageDTO)
            {
                return new HtmlString(PageNavigation(DTO as PageDTO));
            }
            else return new HtmlString("");
        }


    }
}
```

Файл PageMapViewComponent.cs

```
using Education.BLL.DTO.Pages;
using Microsoft.AspNetCore.Html;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace Education.Components
{
    public class PageMapViewComponent
    {
        private string Offset(int level)
        {
            return new string('-', level * 3);
        }

        private string Generate(PageInfo pageInfo, int SelectedId, int Id,
int level = 0)
        {
```

```
            if (pageInfo.Id == Id) return "";

            string res = "";

            foreach (var page in pageInfo.Childs)
                res += Generate(page, SelectedId, Id, level + 1);
            return res;
        }


        public HtmlString Invoke(IEnumerable<PageInfo> map, string name, int
SelectedId, int Id)
        {
            var res = "<select name=\"" + name + "\"><option value=\"-
1\">Нет</option>";
            foreach(var page in map)
                res += Generate(page, SelectedId, Id, 0);
            return new HtmlString(res + "</select>");
        }
    }
}
```

Файл PagingViewComponent.cs

```
using Microsoft.AspNetCore.Html;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Education.Components
{
    public class PagingViewComponent
    {
        private static int PagesCount = 3;

        private string GetRef(string url, string name)
        {
            return "" + name + "";
        }

        private string GetSelectedRef(string url, string name)
        {
            return "" + name + "";
        }

        public HtmlString Invoke(string url, int page, int pages)
        {
            var res = new StringBuilder();

            res.Append("
");

            if (page != 1)
                res.Append(GetRef(url, "<<Первая"));

            int startpage = page <= PagesCount ? 1 : page - PagesCount;
            int endpage = pages - page <= PagesCount ? pages : pages +
PagesCount;

            for(int i = startpage; i<=endpage; i++)
            {
```

```
                if (i == page) res.Append(GetSelectedRef(url + i,
i.ToString()));
                else res.Append(GetRef(url + i, i.ToString()));
            }

            if (page != pages)
                res.Append(GetRef(url + pages, "Последняя>>"));

            return new HtmlString(res.ToString());
        }
    }
}
```

Файл ProfileViewComponent.cs

```
using Education.BLL.DTO.User;
using Education.BLL.Logic.Interfaces;
using Education.BLL.Services.UserServices.Interfaces;
using Education.DAL.Interfaces;
using Microsoft.AspNetCore.Html;
using Microsoft.AspNetCore.Mvc;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace Education.Components
{
    public class ProfileViewComponent : ViewComponent
    {
        private IClaimService ClaimService;
        private IGetUserDTO GetUserService;
        private IUOWFactory DataFactory { get; set; }
        public ProfileViewComponent(IClaimService claimService, IGetUserDTO
getUserDTO, IUOWFactory dataFactory)
        {
            ClaimService = claimService;
            GetUserService = getUserDTO;
            DataFactory = dataFactory;
        }

        private UserDTO GetUser()
        {
            return ClaimService.GetUser(HttpContext.User.Claims);
        }

        private string GetUserName(UserDTO userDTO)
        {
            using(var Data = DataFactory.Get())
            {
                var user = GetUserService.Get(userDTO, Data);
                if (user == null) return "";
                else return user.Info.FullName;
            }
        }

        public HtmlString Invoke()
        {
            var user = GetUser();
            if(user == null)
                return new HtmlString("<div class=\"Login\"><a href =
\"/Account/Login\">Войти</a><a href =
\"/Account/Login\">Зарегистрироваться</a></div>");
            else
```

```
                return new HtmlString("<div class=\"Login\"><text>" +
GetUserName(user) + "</text><a href = \"/Profile/\">Профиль</a><a
href=\"/Account/Logout\">Выйти</a></div>");
        }

    }
}
```

Файл UserGroupViewComponent.cs

```
using Education.BLL.DTO.Forum;
using Education.DAL.Entities;
using Microsoft.AspNetCore.Html;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace Education.Components
{
    public class UserGroupViewComponent
    {
        private const string reqButton = @"";
        private const string chButton = @"";

        private string GetInfo(UserGroupDTO userGroupDTO)
        {
            return @"
"
                + userGroupDTO.UserInfo.Name
                + @"


";
        }

        private string GetEdit(string inside)
        {
            return @"
" + inside + "
";
        }

        private string GetUserFormInfo(UserGroupDTO userGroupDTO, int
GroupId)
        {
            return @"";
        }

        private string GetNotInGroup(UserGroupDTO DTO, int GroupId)
        {
            return GetInfo(DTO) + GetEdit(GetUserFormInfo(DTO, GroupId) +
reqButton);
        }

        private string GetOption(UserGroupStatus id, int vid, string name,
UserGroupStatus sel)
        {
            var res = "";
            else return res + " selected>" + name + "";
        }

        private string GetSelect(UserGroupDTO user)
        {
```

```
            return @"<select name=""Status"">" +
GetOption(UserGroupStatus.request, 0, "Запрос", user.Status)
            + GetOption(UserGroupStatus.member, 1, "Участник", user.Status)
            + GetOption(UserGroupStatus.owner, 2, "Модератор", user.Status)
            + @"<option value=""4"">Удалить</option></select>";
        }

        private string GetInGroup(UserGroupDTO DTO, int GroupId)
        {
            return GetInfo(DTO) + GetEdit(GetUserFormInfo(DTO, GroupId) +
GetSelect(DTO) + chButton);
        }

        public HtmlString Invoke(UserGroupDTO DTO, int GroupId, bool inGroup)
        {
            if (inGroup) return new HtmlString(GetInGroup(DTO, GroupId));
            else return new HtmlString(GetNotInGroup(DTO, GroupId));
        }
    }
}
```

Файл DefaultController.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Education.BLL.DTO.Forum;
using Microsoft.AspNetCore.Mvc;

namespace Education.Controllers.Base
{
    public class DefaultController : Controller
    {
        protected IActionResult Redirect(AccessCode code)
        {
            switch (code)
            {
                case AccessCode.Error:
                    return RedirectToAction("Index", "Error");
                case AccessCode.NoPremision:
                    return RedirectToAction("NoPremision", "Error");
                case AccessCode.NotFound:
                    return RedirectToAction("NoResult", "Error");
                default:
                    return RedirectToAction("Index", "Home");
            }
        }

        protected IActionResult ErrorCode(AccessCode code)
        {
            switch (code)
            {
                case AccessCode.Error:
                    return BadRequest();
                case AccessCode.NoPremision:
                    return Unauthorized();
                case AccessCode.NotFound:
                    return NotFound();
                default:
                    return Ok();
            }
        }
    }
```

```
}
```
Файл AccountController.cs
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using Education.BLL.Services.UserServices.Interfaces;
using Education.BLL.DTO.User;
using System.Security.Claims;
using Microsoft.AspNetCore.Authentication;
using Microsoft.AspNetCore.Authentication.Cookies;
using Microsoft.AspNetCore.Authorization;
using Education.BLL.Logic;
using Education.Models;

namespace Education.Controllers
{
    public class AccountController : Controller
    {
        [Authorize]
        public IActionResult Index()
        {
            return RedirectToAction("Index", "Profile");
        }

        public IActionResult Login()
        {
            if (User.Identity.IsAuthenticated) return
RedirectToAction("Index");

            return View();
        }

        public IActionResult Register()
        {
            if (User.Identity.IsAuthenticated) return
RedirectToAction("Index");

            return View();
        }

        public IActionResult Restore()
        {
            if (User.Identity.IsAuthenticated) return
RedirectToAction("Index");

            return View();
        }


        public IActionResult Logout()
        {
            return RedirectToAction("Logout","AuthApi");
        }

    }
}
```

Файл AdminController.cs
```
using System;
using System.Collections.Generic;
using System.IO;
```

```
using System.Linq;
using System.Threading.Tasks;
using Education.BLL.DTO.Forum;
using Education.BLL.DTO.User;
using Education.BLL.Services.AdminService.Interfaces;
using Education.BLL.Services.ConfigService.Interfaces;
using Education.BLL.Services.UserServices.Interfaces;
using Education.Models;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;

namespace Education.Controllers
{
    public class AdminController : Base.DefaultController
    {
        private IClaimService ClaimService;
        private IAdminService AdminService;
        private IConfigService ConfigService;
        private UserDTO GetUser()
        {
            return ClaimService.GetUser(User.Claims);
        }

        public AdminController(IClaimService claimService, IAdminService
adminService, IConfigService configService)
        {
            ClaimService = claimService;
            AdminService = adminService;
            ConfigService = configService;
        }

        public IActionResult Index()
        {
            if (AdminService.IsAdmin(GetUser()))
                return View();
            else return Redirect(AccessCode.NoPremision);
        }

        #region Users

        public IActionResult Users()
        {
            if (AdminService.IsAdmin(GetUser()))
                return View();
            else return Redirect(AccessCode.NoPremision);
        }

        public IActionResult UserInfo(int id)
        {
            var user = AdminService.GetUser(GetUser(), id);
            if (user.Code != AccessCode.Succsess)
                return Redirect(user.Code);
            return View(user.Result);
        }

        [HttpPost]
        public IActionResult EditUser(AdminUserInfoDTO dto)
        {
            var res = AdminService.EditUser(GetUser(), dto);
            if (res == AccessCode.Succsess)
                return RedirectToAction("Users");
            return Redirect(res);
        }
```

```
        [HttpPost]
        public IActionResult ClearUserSessions(int id)
        {
            var res = AdminService.ResetClaims(GetUser(), id);
            if (res == AccessCode.Succsess)
                return RedirectToAction("Users");
            return Redirect(res);
        }

        [HttpPost]
        public IActionResult SearchUsers(string search = null)
        {
            var user = GetUser();
            if (!AdminService.IsAdmin(user))
                return Redirect(AccessCode.NoPremision);
            IEnumerable users;
            AccessCode code;
            if(search == null)
            {
                var res = AdminService.GetAll(user);
                code = res.Code;
                users = res.Result;
            }
            else
            {
                var res = AdminService.Search(user, search);
                code = res.Code;
                users = res.Result;
            }
            if (code != AccessCode.Succsess)
                return ErrorCode(code);

            return PartialView(users);

        }

        #endregion

        #region Config
        public IActionResult Config()
        {
            if (AdminService.IsAdmin(GetUser()))
                return View(new Config { ConnString =
ConfigService.ConnectionString, Icon = ConfigService.IconPath, Name =
ConfigService.Name });
            else return Redirect(AccessCode.NoPremision);
        }

        public async Task ChangeIcon(IFormFile uploadedFile)
        {
            if (!AdminService.IsAdmin(GetUser()))
                return Redirect(AccessCode.NoPremision);
            using (var fileStream = new FileStream(ConfigService.IconPath,
FileMode.Create))
                await uploadedFile.CopyToAsync(fileStream);
            return RedirectToAction("Index");
        }

        public IActionResult ChangeConfig(Config config)
        {
            if (!AdminService.IsAdmin(GetUser()))
                return Redirect(AccessCode.NoPremision);
```

```
            ConfigService.ConnectionString = config.ConnString;
            ConfigService.Name = config.Name;
            return RedirectToAction("Index");
        }

        #endregion
    }
}
```

Файл AuthApiController.cs

```
using System;
using System.Collections.Generic;
using System.Web;
using System.Security.Claims;
using System.Threading.Tasks;
using Education.BLL.DTO.User;
using Education.BLL.DTO;
using Education.BLL.Services.UserServices.Interfaces;
using Education.Models;
using Education.Captcha;
using Microsoft.AspNetCore.Authentication;
using Microsoft.AspNetCore.Authentication.Cookies;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Authorization;

namespace Education.Controllers
{
    [Produces("application/json")]
    public class AuthApiController : Controller
    {
        private IUserService userService;

        public AuthApiController(IUserService service)
        {
            userService = service;
        }

        private UserDTO GetUser()
        {
            return userService.ClaimService.GetUser(User.Claims);
        }

        #region Auth

        private async Task ClearCookie()
        {
            await
HttpContext.SignOutAsync(CookieAuthenticationDefaults.AuthenticationScheme);
            return RedirectToAction("Index", "Home");
        }

        [Authorize]
        [Route("api/Logout")]
        public async Task Logout()
        {
            userService.Logout(User.Claims);
            return await ClearCookie();
        }

        [Authorize]
        [Route("api/ResetClaims")]
```

```
        public async Task ResetClaims()
        {
            userService.ResetClaims(GetUser());
            return await ClearCookie();
        }

        [Route("api/Auth")]
        [HttpPost]
        public async Task Auth(AuthRequest request)
        {

            AuthResult res = userService.Login(
                new LoginInfoDTO {
                    Login = request.Login,
                    Password = request.Password,
                    Key = request.SecretKey,
                    Browser = Request.Headers["User-Agent"],
                    IP =
Request.HttpContext.Connection.RemoteIpAddress.ToString()
                }
           );
            if (res.Status == AuthStatus.Succsess)
                await
HttpContext.SignInAsync(CookieAuthenticationDefaults.AuthenticationScheme,
new ClaimsPrincipal(res.Identity));
            return new JsonResult(
                new AuthResponse { AuthType = res.authType, KeyTime =
res.KeyTime, Status = res.Status });

        }

        [Route("api/Register")]
        [HttpPost]
        [ValidateRecaptcha]
        public IActionResult Register(RegRequest regRequest)
        {
            var userDTO = new RegUserInfo
            {
                Email = regRequest.Email,
                Phone = regRequest.PhoneNumber,
                Name = regRequest.FullName,
                Login = regRequest.Login,
                Password = regRequest.Password
            };
            return new JsonResult(new RegResponse { Status =
userService.Register(userDTO) });
        }
        #endregion
    }
}
```

Файл BlogController.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Education.BLL.DTO.Forum;
using Education.BLL.DTO.Pages;
using Education.BLL.DTO.User;
using Education.BLL.Services.PageServices.Interfaces;
using Education.BLL.Services.UserServices.Interfaces;
using Education.Controllers.Base;
using Microsoft.AspNetCore.Mvc;
```

```
namespace Education.Controllers
{
    public class BlogController : DefaultController
    {
        private IBlogService BlogService;
        private IClaimService ClaimService;

        public BlogController(IBlogService blogService, IClaimService
claimService)
        {
            BlogService = blogService;
            ClaimService = claimService;
        }

        private UserDTO GetUser()
        {
            return ClaimService.GetUser(User.Claims);
        }

        public IActionResult Page(int id = 1)
        {
            var notes = BlogService.Get(GetUser(), id);
            return View(notes);
        }

        public IActionResult Control(int id = -1)
        {
            var user = GetUser();
            if (id == -1)
            {
                if (BlogService.CanCreate(user))
                    return View(null);
                else return Redirect(AccessCode.NoPremision);
            }

            var note = BlogService.Get(id, user);
            if (note.Item1 == AccessCode.Succsess)
                return View(note.Item2);
            else return Redirect(note.Item1);
        }

        public IActionResult Index(int id)
        {
            var res = BlogService.Get(id, GetUser());
            if(res.Item1 == AccessCode.Succsess)
                return View(res.Item2);
            return Redirect(res.Item1);
        }

        [HttpPost]
        public IActionResult Create(NoteEditDTO noteEditDTO)
        {
            var res = BlogService.Create(noteEditDTO, GetUser());
            if (res.Code == AccessCode.Succsess) return
RedirectToAction("Index", new { res.Id });
            return Redirect(res.Code);
        }

        [HttpPost]
        public IActionResult Update(NoteEditDTO noteEditDTO)
        {
            var res = BlogService.Update(noteEditDTO, GetUser());
```

```
            if (res == AccessCode.Succsess) return RedirectToAction("Index",
new { noteEditDTO.Id });
            return Redirect(res);
        }

        [HttpPost]
        public IActionResult Remove(int id)
        {
            var res = BlogService.Delete(id, GetUser());
            if (res == AccessCode.Succsess) return RedirectToAction("Page");
            return Redirect(res);
        }
    }
}
```

Файл ErrorController.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;

namespace Education.Controllers
{
    public class ErrorController : Controller
    {
        public IActionResult Index()
        {
            return View("Index","Ошибка доступа к данным");
        }

        public IActionResult NoPremision()
        {
            return View("Index", "Ошибка: отказано в доступе");
        }

        public IActionResult NoResult()
        {
            return View("Index", "Ошибка: запись не найдена");
        }


    }
}
```

Файл GroupController.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Education.BLL.DTO.Forum;
using Education.BLL.DTO.Forum.Edit;
using Education.BLL.DTO.User;
using Education.BLL.Services.ForumServices.Interfaces;
using Education.BLL.Services.UserServices.Interfaces;
using Education.Controllers.Base;
using Education.DAL.Entities;
using Education.Models;
using Microsoft.AspNetCore.Mvc;

namespace Education.Controllers
{
    public class GroupController : DefaultController
```

```
    {
        private IGroupService GroupService;
        private IClaimService ClaimService;

        public GroupController(IGroupService groupService, IClaimService
claimService)
        {
            GroupService = groupService;
            ClaimService = claimService;
        }

        private UserDTO GetUser()
        {
            return ClaimService.GetUser(User.Claims);
        }

        public IActionResult Index(int id)
        {
            var res = GroupService.Read(id, GetUser());
            if (res.Item1 == AccessCode.Succsess) return View(res.Item2);
            else return Redirect(res.Item1);
        }

        [HttpPost]
        public IActionResult Delete(int id)
        {
            var res = GroupService.Delete(id, GetUser());
            return Redirect(res);
        }

        [HttpPost]
        public IActionResult Create(GroupRequest groupRequest)
        {
            var groupDTO = new GroupEditDTO
            {
                Name = groupRequest.Name,
                Open = groupRequest.Open,
                Logo = groupRequest.Logo
            };
            var res = GroupService.Create(groupDTO, GetUser());
            if (res.Code == AccessCode.Succsess) return
RedirectToAction("Index", new { id = res.Id });
            else return Redirect(res.Code);
        }

        [HttpPost]
        public IActionResult Update(GroupRequest groupRequest)
        {
            var section = new GroupEditDTO
            {
                Name = groupRequest.Name,
                Open = groupRequest.Open,
                Id = groupRequest.Id,
                Logo = groupRequest.Logo
            };
            var res = GroupService.Update(section, GetUser());
            if (res == AccessCode.Succsess)
                return RedirectToAction("Index", new { id = groupRequest.Id
});
            else return Redirect(res);
        }

        public IActionResult Users(int id)
```

```
        {
            var res = GroupService.GetUsers(id, GetUser());
            if (res.Code == AccessCode.Succsess)
                return View(new UsersOfGroup
                {
                    Data = res.Data,
                    GroupId = id
                });
            else return Redirect(res.Code);
        }

        public IActionResult ControlUser(ControlUser request)
        {
            AccessCode res;
            if (request.Status == Status.Delete)
                res = GroupService.RemoveUser(request.GroupId,
request.UserId, GetUser());
            else if (request.Status == Status.Member)
                res = GroupService.ChangeUserRole(request.GroupId,
request.UserId, UserGroupStatus.member, GetUser());
            else if (request.Status == Status.Owner)
                res = GroupService.ChangeUserRole(request.GroupId,
request.UserId, UserGroupStatus.owner, GetUser());
            else if (request.Status == Status.Request)
                res = GroupService.ChangeUserRole(request.GroupId,
request.UserId, UserGroupStatus.request, GetUser());
            else return Redirect(AccessCode.NotFound);
            if (res != AccessCode.Succsess) return Redirect(res);
            else return RedirectToAction("Users", new { id = request.GroupId
});
        }

        public IActionResult Enter(int id)
        {
            var res = GroupService.Request(GetUser(), id);
            return Redirect(res);
        }

        public IActionResult Leave(int id)
        {
            var res = GroupService.Leave(GetUser(), id);
            return Redirect(res);
        }

        public IActionResult Control(int id = -1)
        {
            if (id == -1)
            {
                if(GroupService.CanCreate(GetUser()))
                    return View(null);
                else return Redirect(AccessCode.NoPremision);
            }
            var res = GroupService.Read(id, GetUser());
            if (res.Item1 == AccessCode.Succsess &&
res.Item2.Access.CanUpdate)
                return View(res.Item2);
            else return Redirect(res.Item1);
        }
    }
}
```

Файл HomeController.cs

```
using System;
```

```
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Education.BLL.DTO.User;
using Education.BLL.Services.ForumServices.Interfaces;
using Education.BLL.Services.PageServices.Interfaces;
using Education.BLL.Services.UserServices.Interfaces;
using Education.Models;
using Microsoft.AspNetCore.Mvc;

namespace Education.Controllers
{
    public class HomeController : Controller
    {
        private IForumService ForumService;
        private IBlogService BlogService;

        private IClaimService ClaimService;
        public HomeController(IForumService forumService, IClaimService
claimService, IBlogService blogService)
        {
            ForumService = forumService;
            BlogService = blogService;
            ClaimService = claimService;
        }

        private UserDTO GetUser()
        {
            return ClaimService.GetUser(User.Claims);
        }

        public IActionResult Index()
        {
            var user = GetUser();
            return View(new MainPage {
                BlogDTO = BlogService.Get(user,1),
                ForumDTO = ForumService.Get(user)
            } );
        }


    }
}
```

Файл ImageManagerController.cs

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Threading.Tasks;
using Education.BLL.DTO;
using Education.BLL.DTO.Forum;
using Education.BLL.DTO.User;
using Education.BLL.Services.ImageManager.Interfaces;
using Education.BLL.Services.UserServices.Interfaces;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;

namespace Education.Controllers
{
    public class ImageManagerController : Controller
    {
```

```
        private IClaimService ClaimService;
        private IImageService ImageService;

        public ImageManagerController(IClaimService claimService,
IImageService imageService)
        {
            ClaimService = claimService;
            ImageService = imageService;
        }

        private UserDTO GetUser()
        {
            return ClaimService.GetUser(User.Claims);
        }

        [HttpPost]
        public IActionResult Load()
        {
            var user = GetUser();
            if (!ImageService.CanLoad(user)) return new UnauthorizedResult();
            foreach(var file in Request.Form.Files)
            {
                if (!ImageService.CheckFile(new FileDTO { Size = file.Length,
Type = Path.GetExtension(file.FileName) }))
                    continue;
                file.CopyTo(new FileStream(ImageService.ImageFolder +
file.FileName, FileMode.CreateNew));
            }
            return new OkResult();
        }

        [HttpPost]
        public IActionResult Delete(string path)
        {
            var res = ImageService.Delete(path, GetUser());
            if (res == AccessCode.NoPremision) return new
UnauthorizedResult();
            if (res == AccessCode.NotFound) return new
NotFoundObjectResult(path);
            if (res == AccessCode.Error) return new BadRequestResult();
            return new OkResult();

        }

        public IActionResult Images()
        {
            return PartialView(ImageService.Get());
        }

    }
}
```

Файл MessageController.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Education.BLL.DTO.Forum;
using Education.BLL.DTO.Forum.Edit;
using Education.BLL.DTO.User;
using Education.BLL.Services.ForumServices.Interfaces;
using Education.BLL.Services.UserServices.Interfaces;
using Education.Controllers.Base;
```

```
using Microsoft.AspNetCore.Mvc;

// For more information on enabling MVC for empty projects, visit
https://go.microsoft.com/fwlink/?LinkID=397860

namespace Education.Controllers
{
    public class MessageController : DefaultController
    {
        private IMessageService MessageService;
        private IClaimService ClaimService;

        private UserDTO GetUser()
        {
            return ClaimService.GetUser(User.Claims);
        }

        public MessageController(IMessageService messageService,
IClaimService claimService)
        {
            MessageService = messageService;
            ClaimService = claimService;
        }

        public IActionResult Index(int id) // Read
        {
            var res = MessageService.Read(id, GetUser());
            if (res.Item1 == AccessCode.Succsess) return View(res.Item2);
            else return Redirect(res.Item1);
        }

        private IActionResult GetMessageTheme(int id, UserDTO userDTO)
        {

            var message = MessageService.Get(id, userDTO);
            if (message != null)
                return RedirectToAction("Index", "Theme", new { id =
message.ThemeId, page = message.Page });
            else return Redirect(AccessCode.NotFound);
        }

        [HttpPost]
        public IActionResult Create(int themeId, string text)
        {
            var messageDTO = new MessageEditDTO
            {
                ThemeId = themeId,
                Text = text
            };
            var user = GetUser();
            var res = MessageService.Create(messageDTO, user);
            if (res.Code == AccessCode.Succsess)
                return GetMessageTheme(res.Id, user);
            else return Redirect(res.Code);
        }

        [HttpPost]
        public IActionResult Remove(int id)
        {
            var res = MessageService.Delete(id, GetUser());
            return Redirect(res);
        }
```

```
        [HttpPost]
        public IActionResult Update(int id, string text)
        {
            var messageDTO = new MessageEditDTO
            {
                Id = id,
                Text = text
            };
            var user = GetUser();
            var res = MessageService.Update(messageDTO, user);
            if (res == AccessCode.Succsess)
                return GetMessageTheme(id, user);
            return Redirect(res);
        }

        public IActionResult Control(int id)
        {
            var res = MessageService.Read(id, GetUser());
            if(res.Item1 == AccessCode.Succsess &&
res.Item2.Access.CanUpdate)
                return View(res.Item2);

            return Redirect(res.Item1);
        }


    }
}
```

Файл PageController.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Education.BLL.DTO.Forum;
using Education.BLL.DTO.Pages;
using Education.BLL.DTO.User;
using Education.BLL.Services.PageServices.Interfaces;
using Education.BLL.Services.UserServices.Interfaces;
using Education.Controllers.Base;
using Education.Models;
using Microsoft.AspNetCore.Mvc;

namespace Education.ControllersPageService
{
    public class PageController : DefaultController
    {
        private IPageService PageService;
        private IClaimService ClaimService;

        public PageController(IPageService pageService, IClaimService
claimService)
        {
            PageService = pageService;
            ClaimService = claimService;
        }

        private UserDTO GetUser()
        {
            return ClaimService.GetUser(User.Claims);
        }

        public IActionResult All()
```

```
        {
            if (PageService.CanCreate(GetUser()))
                return View(PageService.Map.Get);
            else return Redirect(AccessCode.NoPremision);
        }

        public IActionResult Index(int id)
        {
            var res = PageService.Get(id, GetUser());
            if (res.Item1 == AccessCode.Succsess) return View(res.Item2);
            return Redirect(res.Item1);
        }

        public IActionResult Control(int id = -1)
        {
            var user = GetUser();
            if(id == -1)
            {
                if (PageService.CanCreate(user))
                    return View(new PageControl { PageDTO = null, Map =
PageService.Map.Get });
                else return Redirect(AccessCode.NoPremision);
            }
            var page = PageService.Get(id, user);
            if(page.Item1 == AccessCode.Succsess &&
page.Item2.Access.CanUpdate)
                return View(new PageControl { PageDTO = page.Item2, Map =
PageService.Map.Get });
            return Redirect(page.Item1);
        }

        [HttpPost]
        public IActionResult Create(PageEditDTO pageEditDTO)
        {
            var res = PageService.Create(pageEditDTO, GetUser());
            if (res.Code == AccessCode.Succsess)
                return RedirectToAction("index", new { id = res.Id });
            return Redirect(res.Code);
        }

        [HttpPost]
        public IActionResult Edit(PageEditDTO pageEditDTO)
        {
            var res = PageService.Update(pageEditDTO, GetUser());
            if (res == AccessCode.Succsess)
                return RedirectToAction("index", new { id = pageEditDTO.Id
});
            return Redirect(res);
        }

        [HttpPost]
        public IActionResult Remove(int id)
        {
            return Redirect(PageService.Delete(id, GetUser()));
        }
    }
}
```

Файл ProfileController.cs

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Threading.Tasks;
```

```
using Education.BLL.DTO.User;
using Education.BLL.DTO;
using Education.BLL.Services.UserServices.Interfaces;
using Education.DAL.Entities;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using Education.BLL.Services.ImageManager.Interfaces;

namespace Education.Controllers
{
    public class ProfileController : Controller
    {
        private IProfileService ProfileService;
        private IImageService ImageService;
        private UserDTO GetUser()
        {
            return ProfileService.ClaimService.GetUser(User.Claims);
        }

        [Authorize]
        public IActionResult Index()
        {
            return View(ProfileService.GetUserProfile(GetUser()));
        }

        public ProfileController(IProfileService profileService,
IImageService imageService)
        {
            ProfileService = profileService;
            ImageService = imageService;
        }

        [Authorize]
        [HttpPost]
        [Route("api/ConfirmEmail")]
        public ConfirmResult ConfirmEmail(string key = null)
        {
            return ProfileService.ConfirmEmail(GetUser(),key);
        }

        [Authorize]
        [HttpPost]
        [Route("api/ConfirmPhone")]
        public ConfirmResult ConfirmPhone(string key = null)
        {
            return ProfileService.ConfirmPhone(GetUser(),key);
        }

        [Authorize]
        [HttpPost]
        [Route("api/RemovePhone")]
        public ConfirmResult RemovePhone(string key = null)
        {
            return ProfileService.RemovePhone(GetUser(),key);
        }
        [HttpPost]
        [Route("api/RemoveEmail")]
        public ConfirmResult RemoveEmail(string key = null)
        {
            return ProfileService.RemoveEmail(GetUser(), key);
        }
```

```
        [Authorize]
        [HttpPost]
        [Route("api/SetEmail")]
        public SetContactCode SetEmail(string value)
        {
            return ProfileService.SetEmail(GetUser(), value);
        }

        [Authorize]
        [HttpPost]
        [Route("api/SetPhone")]
        public SetContactCode SetPhone(string value)
        {
            return ProfileService.SetPhone(GetUser(), value);
        }

        [Authorize]
        [HttpPost]
        [Route("api/ResetAuthType")]
        public ConfirmResult ResetAuthType(string key = null)
        {
            return ProfileService.ResetAuthType(GetUser(), key);
        }

        [Authorize]
        [HttpPost]
        [Route("api/SetAuthType")]
        public ConfirmResult SetAuthType(AuthType value, string key = null)
        {
            return ProfileService.SetAuthType(GetUser(), value, key);
        }

        [Authorize]
        [HttpPost]
        [Route("api/SetPassword")]
        public ConfirmResult SetPassword(string oldpassword, string
newPassword, string key = null)
        {
            return ProfileService.SetPassword(GetUser(), oldpassword,
newPassword, key);
        }

        [Authorize]
        [HttpPost]
        public IActionResult SetAvatar(IFormFile file)
        {
            var user = GetUser();
            if (file == null || user == null) return Redirect("/");

            var type = Path.GetExtension(file.FileName);

            if (!ImageService.CheckFile(new FileDTO { Size = file.Length,
Type = type }))
                return RedirectToAction("Index");
            var path = ImageService.AvatarPath(user);
            if (!Directory.Exists(path)) Directory.CreateDirectory(path);
            if (path != null) path+= Path.GetRandomFileName() + type;
            file.CopyTo(new FileStream(path, FileMode.Create));
            ProfileService.SetAvatar(user, path.Replace(@"wwwroot", ""));
            return RedirectToAction("Index");
        }

    }
```

```
}
```

Файл RestoreApiController.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using Education.Captcha;
using Education.BLL.Services.UserServices.Interfaces;
using Education.Models;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;

namespace Education.Controllers
{
    [Produces("application/json")]
    public class RestoreApiController : Controller
    {
        IRestorePasswordService RestorePasswordService;
        public RestoreApiController(IRestorePasswordService
restorePasswordService)
        {
            RestorePasswordService = restorePasswordService;
        }

        [HttpPost]
        [ValidateRecaptcha]
        [Route("api/Restore")]
        public IActionResult Restore(RestoreRequest request)
        {
            var result = RestorePasswordService.Restore(request.Login,
request.Password, request.Key);
            return new JsonResult(result);
        }

    }
}
```

Файл SectionController.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Education.BLL.DTO.Forum;
using Education.BLL.DTO.Forum.Edit;
using Education.BLL.DTO.User;
using Education.BLL.Services.ForumServices.Interfaces;
using Education.BLL.Services.UserServices.Interfaces;
using Education.Controllers.Base;
using Education.Models;
using Microsoft.AspNetCore.Mvc;

namespace Education.Controllers
{
    public class SectionController : DefaultController
    {
        private ISectionService SectionService;
        private IClaimService ClaimService;

        public SectionController(ISectionService sectionService,
IClaimService claimService)
        {
            SectionService = sectionService;
            ClaimService = claimService;
```

```
        }


        private UserDTO GetUser()
        {
            return ClaimService.GetUser(User.Claims);
        }

        public IActionResult Index(int id)
        {
            var res = SectionService.Read(id, GetUser());
            if (res.Item1 == AccessCode.Succsess) return View(res.Item2);
            else return Redirect(res.Item1);
        }

        [HttpPost]
        public IActionResult Delete(int id)
        {
            var res = SectionService.Delete(id, GetUser());
            return Redirect(res);
        }

        [HttpPost]
        public IActionResult Create(SectionRequest sectionRequest)
        {
            var section = new SectionEditDTO
            {
                Name = sectionRequest.Name,
                Open = sectionRequest.Open,
                GroupId = sectionRequest.GroupId
            };

            var res = SectionService.Create(section, GetUser());
            if (res.Code == AccessCode.Succsess)
                return RedirectToAction("Index", new { id = res.Id });
            else return Redirect(res.Code);
        }

        [HttpPost]
        public IActionResult Update(SectionRequest sectionRequest)
        {
            var section = new SectionEditDTO
            {
                Name = sectionRequest.Name,
                Open = sectionRequest.Open,
                Id = sectionRequest.Id
            };
            var res = SectionService.Update(section, GetUser());
            if (res == AccessCode.Succsess)
                return RedirectToAction("Index", new { id = sectionRequest.Id
});
            else return Redirect(res);
        }

        public IActionResult Control(int id)
        {
            var res = SectionService.Read(id, GetUser());
            if (res.Item1 == AccessCode.Succsess &&
res.Item2.Access.CanUpdate)
                return View(new SectionControl(id, res.Item2));
            else return Redirect(res.Item1);
        }
```

```
        public IActionResult Add(int id)
        {
            if (SectionService.CanCreate(id, GetUser()))
                return View("Control", new SectionControl(id, null));
            else return Redirect(AccessCode.NoPremision);
        }

    }
}
```

Файл ThemeController.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Education.BLL.DTO.Forum;
using Education.BLL.DTO.Forum.Edit;
using Education.BLL.DTO.User;
using Education.BLL.Services.ForumServices.Interfaces;
using Education.BLL.Services.UserServices.Interfaces;
using Education.Controllers.Base;
using Education.Models;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;

namespace Education.Controllers
{
    public class ThemeController : DefaultController
    {
        private IThemeService ThemeService;
        private IClaimService ClaimService;

        public ThemeController(IThemeService themeService, IClaimService
claimService)
        {
            ThemeService = themeService;
            ClaimService = claimService;
        }

        private UserDTO GetUser()
        {
            return ClaimService.GetUser(User.Claims);
        }

        public IActionResult Index(int id, int page = 1)
        {
            var res = ThemeService.Read(id, page, GetUser());
            if (res.Item1 == AccessCode.Succsess) return View(res.Item2);
            else return Redirect(res.Item1);
        }

        [HttpPost]
        public IActionResult Remove(int id)
        {
            var res = ThemeService.Delete(id, GetUser());
            return Redirect(res);
        }

        [HttpPost]
        public IActionResult Create(ThemeRequest themeRequest)
        {
            var theme = new ThemeEditDTO
            {
```

```
                Name = themeRequest.Name,
                SectionId = themeRequest.SectionId,
                Open = themeRequest.Open,
                Text = themeRequest.Text
            };

            var res = ThemeService.Create(theme, GetUser());
            if (res.Code == AccessCode.Succsess)
                return RedirectToAction("Index", new { id = res.Id });
            else return Redirect(res.Code);
        }

        [HttpPost]
        public IActionResult Update(ThemeRequest themeRequest)
        {
            var theme = new ThemeEditDTO
            {
                Name = themeRequest.Name,
                Open = themeRequest.Open,
                Text = themeRequest.Text,
                Id = themeRequest.Id
            };

            var res = ThemeService.Update(theme, GetUser());
            if (res == AccessCode.Succsess)
                return RedirectToAction("Index", new { id = themeRequest.Id
});
            else return Redirect(res);
        }

        public IActionResult Control(int id)
        {
            var res = ThemeService.Read(id, GetUser());
            if(res.Item1 == AccessCode.Succsess &&
res.Item2.Access.CanUpdate)
            {
                return View(new ThemeControl(-1, res.Item2));
            }
            return Redirect(res.Item1);
        }

        public IActionResult Add(int id)
        {
            if (ThemeService.CanCreate(id, GetUser()))
                return View("Control", new ThemeControl(id, null));
            else return Redirect(AccessCode.NoPremision);
        }

    }
}
```

Файл AuthRequest.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace Education.Models
{
    public class AuthRequest
    {
        public string Login { get; set; }
        public string Password { get; set; }
```

```
        public string SecretKey { get; set; }

    }
}
```

Файл AuthResponse.cs

```
using Education.BLL.Services.UserServices.Interfaces;
using Education.DAL.Entities;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace Education.Models
{
    public class AuthResponse
    {
        public AuthStatus Status { get; set; }
        public AuthType AuthType { get; set; }
        public DateTime? KeyTime { get; set; }
    }
}
```

Файл Config.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace Education.Models
{
    public class Config
    {
        public string Name { get; set; }
        public string Icon { get; set; }
        public string ConnString { get; set; }
    }
}
```

Файл ControlUser.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace Education.Models
{
    public enum Status
    {
        Request,
        Member,
        Owner,
        Delete
    }

    public class ControlUser
    {
        public int GroupId { get; set; }
        public int UserId { get; set; }
        public Status Status { get; set; }
    }
}
```

Файл ErrorMessage.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace Education.Models
{
    public class ErrorMessage
    {
        public string Error { get; set; }
    }
}
```

Файл GroupRequest.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace Education.Models
{
    public class GroupRequest
    {
        public int Id { get; set; }
        public string Name { get; set; }
        public string Logo { get; set; }
        public bool Open { get; set; }
    }
}
```

Файл HomeModel.cs

```
using Education.BLL.DTO.Forum;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace Education.Models
{
    public class HomeModel
    {
        public IEnumerable Groups { get; set; }
        public bool CanCreateGroups { get; set; }
    }
}
```

Файл MainPage.cs

```
using Education.BLL.DTO.Forum;
using Education.BLL.DTO.Pages;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace Education.Models
{
    public class MainPage
    {
        public ForumDTO ForumDTO { get; set;}
        public BlogDTO BlogDTO { get; set; }
    }
}
```

Файл PageControl.cs

```
using Education.BLL.DTO.Pages;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace Education.Models
{
    public class PageControl
    {
        public IEnumerable Map { get; set; }
        public PageDTO PageDTO { get; set; }
    }
}
```

Файл RegRequest.cs

```
using Education.BLL.DTO.User;
using Education.BLL.Services.UserServices.Interfaces;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace Education.Models
{
    public class RegRequest
    {
        public string Login { get; set; }
        public string FullName { get; set; }
        public string Email { get; set; }
        public string PhoneNumber { get; set; }
        public string Password { get; set; }
    }
}
```

Файл RegResponse.cs

```
using Education.BLL.Services.UserServices.Interfaces;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace Education.Models
{
    public class RegResponse
    {
        public RegisterResult Status { get; set; }
    }
}
```

Файл RestoreRequest.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace Education.Models
{
    public class RestoreRequest
    {
        public string Login { get; set; }
```

```
        public string Key { get; set; }
        public string Password { get; set; }
    }
}
```

Файл SectionControl.cs

```
using Education.BLL.DTO.Forum;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace Education.Models
{
    public class SectionControl
    {
        public int GroupId { get; set; }
        public SectionDTO SectionDTO { get; set; }

        public SectionControl(int groupId, SectionDTO sectionDTO)
        {
            GroupId = groupId;
            SectionDTO = sectionDTO;
        }
    }
}
```

Файл SectionRequest.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace Education.Models
{
    public class SectionRequest
    {
        public int Id { get; set; }
        public int GroupId { get; set; }
        public string Name { get; set; }
        public bool Open { get; set; }
    }
}
```

Файл ThemeControl.cs

```
using Education.BLL.DTO.Forum;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace Education.Models
{
    public class ThemeControl
    {
        public int SectionId { get; set; }
        public ThemeDTO ThemeDTO { get; set; }

        public ThemeControl(int sectionId, ThemeDTO themeDTO)
        {
            SectionId = sectionId;
            ThemeDTO = themeDTO;
        }
```

```
    }
}
```

Файл ThemeRequest.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace Education.Models
{
    public class ThemeRequest
    {
        public int Id { get; set; }
        public int SectionId { get; set; }
        public string Name { get; set; }
        public string Text { get; set; }
        public bool Open { get; set; }
    }
}
```

Файл UsersOfGroup.cs

```
using Education.BLL.DTO.Forum;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace Education.Models
{
    public class UsersOfGroup
    {
        public int GroupId { get; set; }
        public IEnumerable Data { get; set; }
    }
}
```

Файл Program.cs

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore;
using Microsoft.AspNetCore.Hosting;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.Logging;

namespace Education
{
    public class Program
    {
        public static void Main(string[] args)
        {
            BuildWebHost(args).Run();
        }

        public static IWebHost BuildWebHost(string[] args) =>
            WebHost.CreateDefaultBuilder(args)
                .UseStartup()
                .Build();
    }
}
```

Файл Startup.cs

```
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Hosting;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using Education.BLL.Services.UserServices.Interfaces;
using Education.BLL.Services.UserServices.Auth;
using Microsoft.AspNetCore.Authentication.Cookies;
using Education.DAL.Interfaces;
using Education.DAL.Repositories;
using Education.BLL.Services.UserServices.KeyGenerators;
using Education.BLL.Services.UserServices.Messagers;
using Education.BLL.Services.UserServices.Messengers;
using Education.BLL.Services.UserServices.Confirm;
using Education.BLL.Services.UserServices.Profile;
using Education.BLL.Services.UserServices.Restore;
using System.Threading.Tasks;
using Education.BLL.Logic.Interfaces;
using Education.BLL.Logic.Rules;
using Education.BLL.Logic;
using Education.BLL.Services.ForumServices.Interfaces;
using Education.BLL.Services.ForumServices;
using Education.BLL.Services.ImageManager;
using Education.BLL.Services.ImageManager.Interfaces;
using Microsoft.AspNetCore.Http.Features;
using Education.BLL.Services.PageServices.Interfaces;
using Education.BLL.Services.PageServices;
using Education.BLL.Services.AdminService.Interfaces;
using Education.BLL.Services.AdminService;
using Education.BLL.Services.ConfigService.Interfaces;
using Education.BLL.Services;

namespace Education
{
    public class Startup
    {
        public Startup(IConfiguration configuration)
        {
            Configuration = configuration;
        }

        public IConfiguration Configuration { get; }

        // This method gets called by the runtime. Use this method to add
services to the container.
        public void ConfigureServices(IServiceCollection services)
        {
            //-------------------------------------------------------
            IConfigService configService = new
XMLConfigService("Config.xml");
            IMessenger messenger_sms = new SmsMessenger();
            IMessenger messenger_email = new EmailMessenger();
            IPassHasher passHasher = new SHA256Hasher();
            IKeyGenerator smallKeyGenerator = new SmallKeyGenerator();
            IKeyGenerator bigKeyGenerator = new BigKeyGenerator();
            IRegValidator regValidator = new RegValidator();
            IUOWFactory UOWFactory = new
EFUOWFactory(configService.ConnectionString);
            IGetUserDTO getUserDTO = new GetUserDTO();

            //-------------------------------------------------------
            IClaimService claimService = new ClaimService(UOWFactory);
```

```
            //-------------------------------------------------------
            services.AddSingleton(
                   serviceProvider =>
                   {
                       return configService;
                   }
               );
            //-------------------------------------------------------
            services.AddSingleton(
                   serviceProvider =>
                   {
                       return getUserDTO;
                   }
               );
            //-------------------------------------------------------
            services.AddSingleton(
                serviceProvider =>
                {
                    return UOWFactory;
                }
            );

            services.AddSingleton();
            //-------------------------------------------------------

            services.AddSingleton(
                serviceProvider =>
                {
                    return new UserAuthService(
                        UOWFactory,
                        new AuthKeyService(smallKeyGenerator, messenger_sms),
                        new
AuthKeyService(smallKeyGenerator,messenger_email),
                        passHasher,
                        regValidator,
                        claimService,
                        bigKeyGenerator,
                        getUserDTO
                        );
                }
            );

            services.AddSingleton(
               serviceProvider =>
               {
                   IConfirmService emailCS = new ConfirmService(
                       new ConfirmKeyService(bigKeyGenerator,
messenger_email)
                       );
                   IConfirmService phoneCS = new ConfirmService(
                       new ConfirmKeyService(smallKeyGenerator,
messenger_sms)
                       );
                   return new ProfileService(
                       UOWFactory,
                       regValidator,
                       emailCS,
                       phoneCS,
                       passHasher,
                       claimService,
                       getUserDTO
                      );
               }
```

```
            );

             services.AddSingleton(
                serviceProvider =>
                {
                    var emaiCKS = new ConfirmKeyService(bigKeyGenerator,
messenger_email);
                    var phoneCKS = new ConfirmKeyService(smallKeyGenerator,
messenger_sms);
                    return new RestorePasswordService(UOWFactory, emaiCKS,
phoneCKS, new RegValidator(), passHasher);
                }
             );
             //---------Forum Services--------------------------------
             IGroupRules groupRules = new GroupRules();
             ISectionRules sectionRules = new SectionRules(groupRules);
             IThemeRules themeRules = new ThemeRules(sectionRules);
             IMessageRules messageRules = new MessageRules(themeRules,
sectionRules);
             IDTOHelper dtoHelper = new DTOHelper();
             IForumDTOHelper forumDTOHelper = new ForumDTOHelper(messageRules,
themeRules, sectionRules, groupRules, dtoHelper);

             services.AddSingleton(
               serviceProvider =>
               {
                   return new GroupService(groupRules, getUserDTO, UOWFactory,
forumDTOHelper);
               }
             );

             services.AddSingleton(
               serviceProvider =>
               {
                   return new SectionService(sectionRules, getUserDTO,
UOWFactory, forumDTOHelper);
               }
             );

             services.AddSingleton(
               serviceProvider =>
               {
                   return new ThemeService(themeRules, getUserDTO, UOWFactory,
forumDTOHelper);
               }
             );

             services.AddSingleton(
               serviceProvider =>
               {
                   return new MessageService(messageRules, getUserDTO,
UOWFactory, forumDTOHelper);
               }
             );

             services.AddSingleton(
               serviceProvider =>
               {
                  return new ForumService(getUserDTO, UOWFactory,
forumDTOHelper, groupRules);
               }
             );
             //--------------------Page Services--------------------
```

```
            IPageRules pageRules = new PageRules();
            INoteRules noteRules = new NoteRules();

            services.AddSingleton(
                 serviceProvider =>
                    {
                        return new PageService(pageRules, UOWFactory,
getUserDTO, dtoHelper);
                    }
            );

            services.AddSingleton(
                 serviceProvider =>
                 {
                     return new BlogService(noteRules, UOWFactory,
getUserDTO, dtoHelper);
                 }
            );

            //-------------------------------------------------------
            services.AddSingleton(
                serviceProvider =>
                {
                    return new ImageService(UOWFactory, getUserDTO);
                }
            );
            //----------------------------------------
            services.AddSingleton(
                serviceProvider =>
                {
                    return new AdminService(getUserDTO, UOWFactory);
                }
            );

            services.Configure(x =>
            {
                x.ValueCountLimit = int.MaxValue;
                x.MemoryBufferThreshold = int.MaxValue;
                x.ValueLengthLimit = int.MaxValue;
                x.MultipartBodyLengthLimit = int.MaxValue; // In case of
multipart
            });
            //-------------------------------------------------------

services.AddAuthentication(CookieAuthenticationDefaults.AuthenticationScheme)
                 .AddCookie(options =>
                 {
                     options.LoginPath = new
Microsoft.AspNetCore.Http.PathString("/Account/Login");
                     options.LogoutPath = new
Microsoft.AspNetCore.Http.PathString("/Account/Logout");
                     options.Events.OnValidatePrincipal =
PrincipalValidator.ValidateAsync;
                 });

            services.AddMvc();
        }

        // This method gets called by the runtime. Use this method to
configure the HTTP request pipeline.
        public void Configure(IApplicationBuilder app, IHostingEnvironment
env)
        {
```

```
            if (env.IsDevelopment())
            {
                app.UseDeveloperExceptionPage();
                app.UseBrowserLink();
            }
            else
            {
                app.UseExceptionHandler("/Home/Error");
            }

            app.UseStaticFiles();
            app.UseAuthentication();
            app.UseMvc(routes =>
            {
                routes.MapRoute(
                    name: "default",
                    template:
"{controller=Home}/{action=Index}/{id?}/{page?}");
            });
        }
    }
}

public static class PrincipalValidator
{
    public static Task ValidateAsync(CookieValidatePrincipalContext context)
    {
        Task task = new Task(() =>
        {
            if (context == null) throw new
System.ArgumentNullException(nameof(context));
            var ClaimService =
context.HttpContext.RequestServices.GetRequiredService();
            var userDTO = ClaimService.GetUser(context.Principal.Claims);
            if (userDTO == null) return;

            var GetUserService =
context.HttpContext.RequestServices.GetRequiredService();
            var DataFactory =
context.HttpContext.RequestServices.GetRequiredService();

            using (var Data = DataFactory.Get())
            {
                var user = GetUserService.Get(userDTO, Data);
                if (user == null) context.RejectPrincipal();
            }
        });
        task.Start();
        return task;
    }
}
```

Файл Login.cshtml

```
<h2>Авторизация</h2>
@Html.Partial("LoginPanel");
```

Файл LoginPanel.cshtml

```
@using Education.BLL.Services.UserServices.Interfaces;
@using Education.DAL.Entities;

<script src="/lib/jquery/dist/jquery.js"></script>
<script src='https://www.google.com/recaptcha/api.js'></script>
```

```
<script>
    function LoginAjax(login, pass, key) {
        $.ajax({
            url: '/api/Auth',
            method: 'POST',
            data: {
                Login: login,
                Password: pass,
                SecretKey: key
            },
            success: function (data) {
                CheckAuthResponse(data);
            },
            error: function (jqxhr, status, errorMsg) {
                ErrorMessage("Внутреняя ошибка сервера");
                HideButton();
            }

        });
    }


    function CheckAuthResponse(data) {
        if (data.status == 0) SuccsessStatus();
        else if (data.status == 5) KeySentStatus(data);
        else if (data.error != null) ErrorMessage(data.error);
        else if (data.status == 4) NeedNewKeyStatus();
        else if (data.status == 3) WrongKey();
        else ErrorStatus(data);
    }

    function SuccsessStatus() {
        window.location.href = '/';
    }

    function WrongKey() {
        InfoMessage("Неправильный ключ!");

    }

    function NeedNewKeyStatus() {
        $("#key").val("");
        ErrorMessage("Срок действия ключа истек!");
    }

    function KeySentStatus(data) {

        var SendTo;
        if (data.authType == 1) SendTo = "Почту";
        else if (data.authType == 2) SendTo = "Телефон";
        InfoMessage("Ключ отправлен на " + SendTo + ", срок действия: " +
data.keyTime);
        $("#DarkSpace").show();

    }

    function ErrorStatus(data) {
        if (data.status == 1) ErrorMessage("Пользователь не найден!");
        else if (data.status == 2) ErrorMessage("Пользователь забанен, дата
разбана: !" + data.keyTime);
        else if (data.status == 6) ErrorMessage("Внутреняя ошибка, обратитесь
к администратору!");
    }
```

```
    function LoginClick() {
        LoginAjax($("#login").val(), $("#password").val(), $("#key").val());
    }

    function ErrorMessage(text) {
        HideDarkSpace();
        $('#ErrorMessage').text(text);

    }

    function InfoMessage(text) {
        $("#info").text(text);

    }

    function HideDarkSpace() {
        $("#key").val("");
        $("#DarkSpace").hide();

    }

    //-----------------------
    $(document).ready(function () {
        HideDarkSpace();
    });
</script>

<div class="Panel">
    <div class="DataFields">
        <input id="login" type="text" placeholder="Имя
акканута/email/телефон" /><br />
        <input id="password" type="password" placeholder="Пароль" /><br />
    </div>
    <div class="ErrorMessage" id="ErrorMessage">
    </div>
    <input class="Button" type="button" id="Login" onclick="LoginClick()"
value="Войти" />
    <a href="/Account/Restore">Востановить пароль</a>
    <a href="/Account/Register">Регистрация</a>
</div>

<div class="DarkSpace" id="DarkSpace">
    <div class="SecretKeyPanel">
        <div class="PanelLabel">
            <text>Двухфакторная авторизация</text>
        </div>
        <div class="PanelBody">
            <text id="info"></text><br />
            <input id="key" type="text" />
            <text id="keyTime"> </text>
            <div class="Buttons">
                <input type="button" onclick="LoginClick()" value="Войти" />
                <input type="button" onclick="HideDarkSpace()" value="Отмена"
/>
            </div>
        </div>
    </div>
</div>
```

Файл Register.cshtml

```
@{
    ViewData["Title"] = "Register";
```

```
}

<h2>Регистрация</h2>

@Html.Partial("RegisterPanel");
```

Файл RegisterPanel.cshtml

```
<script src="/lib/jquery/dist/jquery.js"></script>
<script src='https://www.google.com/recaptcha/api.js'></script>

@using  Education.BLL.Logic;
@using Education.BLL.Services.UserServices.Interfaces;
@using Education.Captcha;

<script>

    var Patterns = {};
    var RegExs = {};
    var Checks = {};
    Patterns["login"] =
"@Html.Raw(regRegularExpressions.LoginRegex.Replace("\\","\\\\"))";
    Patterns["email"] =
"@Html.Raw(regRegularExpressions.EmailRegex.Replace("\\","\\\\"))";
    Patterns["phone"] =
"@Html.Raw(regRegularExpressions.PhoneRegex.Replace("\\","\\\\"))";
    Patterns["password"] =
"@Html.Raw(regRegularExpressions.PasswordRegex.Replace("\\","\\\\"))";
    Patterns["fullname"] =
"@Html.Raw(regRegularExpressions.FullNameRegex.Replace("\\","\\\\"))";


    function RegisterAjax(login, pass, email, phone, fullname) {
        $.ajax({
            url: '/api/Register',
            method: 'POST',
            data: {
                Login: login,
                Password: pass,
                Email: email,
                PhoneNumber: phone,
                Fullname: fullname,
            },
            dataType: "json",
            headers: {
                'Captcha': grecaptcha.getResponse()
            },
            success: function (data) {
                CheckRegResponse(data);
                HideButton();
                grecaptcha.reset();
            },
            error: function (xhr, ajaxOptions, thrownError) {
                ErrorMessage("Ошибка сервера, обратитесь к администратору");
                HideButton();
                grecaptcha.reset();
            }
        });

    }

    function TryReg() {
        RegisterAjax($("#login").val(),
            $("#password").val(),
```

```
        $("#email").val(),
        $("#phone").val(),
        $("#fullname").val());
}

function CheckRegResponse(data) {
    if (data.error != null) {
        ErrorMessage(data.error);
    }
    var code = data.status;
    if (code == @((int)RegisterResult.Confirm)){
        window.location.href = '/Account/Login';
    } else if (code == @((int)RegisterResult.EmailAlreadyExists)){
        ErrorMessage("Аккаунт с таким email уже существует");
    } else if (code == @((int)RegisterResult.LoginAlreadyExists)){
        ErrorMessage("Аккаунт с таким Именем уже существует");
    } else if (code == @((int)RegisterResult.NeedContact)){
        ErrorMessage("Необходим хотябы 1 контакт (телефон или email)");
    } else if (code == @((int)RegisterResult.PhoneAlreadyExists)){
        ErrorMessage("Аккаунт с таким телефоном уже существует");
    } else if (code == @((int)RegisterResult.WrongEmail)){
        ErrorMessage("Неправильный формат email");
    } else if (code == @((int)RegisterResult.WrongFullName)){
        ErrorMessage("Неправильный формат ФИО");
    } else if (code == @((int)RegisterResult.WrongLogin)){
        ErrorMessage("Неправильный формат имени аккаунты");
    } else if (code == @((int)RegisterResult.WrongPassword)){
        ErrorMessage("Неправильный формат пароля");
    } else if (code == @((int)RegisterResult.WrongPhone)){
        ErrorMessage("Неправильный формат телефона");
    }

}

function ErrorMessage(text) {
    $("#error").text(text);

}

function checkInput(name) {
    if (RegExs[name] == null) {
        if (Patterns[name] == null) return true;
        RegExs[name] = new RegExp(Patterns[name]);
    }
    var inpt = $('#' + name).val();
    return RegExs[name].test(inpt);
}

function InputListener(input) {

    if ($(input).val() == "") {
        $(input).css("border-color", "");
        Checks[input.id] = 0;
    }else if (!checkInput(input.id)) {
        $(input).css("border-color", "red");
        Checks[input.id] = 1;
    }
    else {
        $(input).css("border-color", "lightgreen");
        Checks[input.id] = 2;
    }
    ShowButton();
}
```

```
    function PasswordConfirmListener(password, confirmpassword) {
        var cfpv = confirmpassword.val();
        var cfvid = confirmpassword.get(0).id;
        var psv = password.val();
        if (cfpv == "") {
            Checks[cfvid] = 0;
            confirmpassword.css("border-color", "");
        } else if (cfpv != psv) {
            confirmpassword.css("border-color", "red");
            Checks[cfvid] = 1;
        }
        else {
            confirmpassword.css("border-color", "lightgreen");
            Checks[cfvid] = 2
        }
        ShowButton();
    }

    function ShowButton() {
        var button = $('#register');
        if (CheckInputs()) button.show();
        else HideButton();

    }

    function HideButton() {
        $('#register').hide();
    }

    function isCaptchaChecked() {
        return grecaptcha && grecaptcha.getResponse().length !== 0;
    }

    function CheckInputs() {
        if (!isCaptchaChecked()) return false;
        var res = 0;
        var count = 0;
        $.each(Checks, function (index, value) {
            count++;
            res += value;
        });
        if (res == 2 * count) return true;
        else if (res == 2 * (count - 1) && Checks['email'] == 2 ^
Checks['phone'] == 2) return true;
        else return false;
    }

    $(document).ready(function () {
        var inputs =
$("input[type='text'],input[type='email'],input[type='password']");
        $.each(inputs, function (index, value) {
            if (Patterns[value.id] == null) return;
            $(value).bind("input", function () { InputListener(value) });
            Checks[value.id] = 0;
        });
        var cfp = $("#password_confirm");
        cfp.bind("input", function () {
PasswordConfirmListener($("#password"), cfp); });
        $("#password").bind("input", function () {
PasswordConfirmListener($("#password"), cfp); });
        Checks[cfp.get(0).id] = 0;
        $('#register').hide();
```

```
    });


</script>

<div class="RegInfo">

    <hr>
    <p><strong>Требования к аккаунту:</strong></p>
    <ul>
        <li><strong>Имя пользователя:</strong> длина от 4 до 20 символов,
только латинские символы и символ '-'</li>
        <li><strong>Контактные данные:</strong> необходимо указать телефон
или email, или и то и другое</li>
        <li><strong>ФИО:</strong> Только кирилические символы, пример:
"Иванов Иван Иванович"</li>
        <li><strong>Пароль:</strong> длина от 6 символов, наличие минимум
заглавной и строчной латинских символов и одной цифры</li>
    </ul>
    <hr>
</div>

<div class="Panel">
    <div class="DataFields">
        <input id="login" type="text" placeholder="Имя" />
        <input id="email" type="email" placeholder="Почта" />
        <input id="phone" type="text" placeholder="Номер телефона" />
        <input id="fullname" type="text" placeholder="ФИО Полностью" />
        <input id="password" type="password" placeholder="Пароль" />
        <input id="password_confirm" type="password" placeholder="Повторите
пароль" />
    </div>
    <div class="g-recaptcha" data-callback="ShowButton" data-
sitekey="@Html.Raw(CaptchaInfo.Key) "></div>
    <text id="error" class="ErrorMessage">  </text>
    <input class="Button" id="register" type="button" onclick="TryReg()"
value="Зарегестрироваться" />
</div>
```

Файл Restore.cshtml

```
@{
    ViewData["Title"] = "Restore";
}

<h2>Востановить пароль</h2>

@Html.Partial("RestorePanel");
```

Файл RestorePanel.cshtml

```
@using  Education.BLL.DTO;
@using Education.Captcha;
@using Education.DAL.Entities;

<script src="/lib/jquery/dist/jquery.js"></script>
<script src='https://www.google.com/recaptcha/api.js'></script>

<script>

    var LoginRegexs = {};

    LoginRegexs["login"] = new
RegExp("@Html.Raw(regRegularExpressions.LoginRegex.Replace("\\","\\\\"))");
```

```
    LoginRegexs["email"] = new
RegExp("@Html.Raw(regRegularExpressions.EmailRegex.Replace("\\","\\\\"))");
    LoginRegexs["phone"] = new
RegExp("@Html.Raw(regRegularExpressions.PhoneRegex.Replace("\\","\\\\"))");
    var PasswordRegex = new
RegExp("@Html.Raw(regRegularExpressions.PasswordRegex.Replace("\\","\\\\"))")
;

    function RestoreAjax(login, pass, key) {
        $.ajax({
            url: '/api/Restore',
            method: 'POST',
            data: {
                Login: login,
                Password: pass,
                Key: key
            },
            dataType: "json",
            headers: {
                'Captcha': grecaptcha.getResponse()
            },
            success: function (data) {
                CheckResponse(data)
                grecaptcha.reset();
            },
            error: function (xhr, ajaxOptions, thrownError) {
                CheckResponse(xhr);
                grecaptcha.reset();
            }
        });

    }

    function TryRestore() {
        RestoreAjax($("#login").val(), $("#password").val(),
$("#key").val());
    }

    function GetSendTo(code) {
        if (code == @((int)AuthType.Email)) return "email";
        else if (code == @((int)AuthType.Phone)) return "телефон";
        return "";
    }

    function CheckResponse(data) {
        if (data.error != null) {
            WriteMessage(data.error);
            return;
        }
        if (data.status == @((int)RestoreCode.Succsess)){
            window.location.href = '/Account/Login';
        } else if (data.status == @((int)RestoreCode.KeySent)){
            WriteMessage("Ключ отправлен на " + GetSendTo(data.sendTo) + ",
действителен до: " + data.keyTime);
            restorPanel.SetStatus(true, true);
        }else if (data.status == @((int)RestoreCode.NeedContact)){
            WriteMessage("На этом аккаунте нет подтвержденных контактных
данных, обратитесь к администратору");
        }else if (data.status == @((int)RestoreCode.NeedNewKey)){
            WriteMessage("Ключ не действителен");
            restorPanel.SetStatus(false, true);
        }else if (data.status == @((int)RestoreCode.UserNotFound)){
            WriteMessage("Пользователь не найден");
```

```
        }else if (data.status == @((int)RestoreCode.WorngKey)){
            WriteMessage("Неправильный ключ");
        }else if (data.status == @((int)RestoreCode.WrongPassword)){
            WriteMessage("Неправильный формат пароля");
        }

    }

    function WriteMessage(text) {
        $("#Message").text(text);
    }



    function RestorePanel(loginInput, passwordInput, cfpasswordInput,
keyInput, buttonInput) {
        var Status;
        var Login = loginInput;
        var Password = passwordInput;
        var CfPassword = cfpasswordInput;
        var Key = keyInput;
        var Button = buttonInput;

        function ShowAndClear(object) {
            object.val("");
            object.show();
        }

        function HideAndClear(object) {
            object.val("");
            object.hide();
        }

        function CheckLoginInput(login) {
            function CheckLogin(login) {
                var result;
                $.each(LoginRegexs, function (index, value) {
                    result = value.test(login);
                    if (result) return false;
                });
                return result;
            }

            var result = CheckLogin(login.val());
            if (login.val() == "") login.css("border-color", "");
            else if (result) login.css("border-color", "lightgreen");
            else login.css("border-color", "red");
            return result;
        }

        function CheckPasswordInputs(password, confpassword) {
            var checkP = false;
            var checkCP = false;
            if (password.val() == "") password.css("border-color", "");
            else if (PasswordRegex.test(password.val())) {
                checkP = true;
                password.css("border-color", "lightgreen");
            }
            else password.css("border-color", "red");

            if (confpassword.val() == "") confpassword.css("border-color",
"");
            else if (confpassword.val() == password.val()) {
```

```
                var checkCP = true;
                confpassword.css("border-color", "lightgreen");
            }
            else confpassword.css("border-color", "red");

            return checkP && checkCP;
        }

        this.CheckInputs = function () {
            var Check;
            if (Status) {
                Check = CheckPasswordInputs(Password, CfPassword);
            } else {
                Check = CheckLoginInput(Login);
            }
            if (Check && isCaptchaChecked()) Button.show();
            else Button.hide();
        }

        this.GetStatus = function () {
            return Status;
        }

        this.SetStatus = function (status, resetCaptcha) {
            Status = status;
            if (Status) {
                Login.hide();

                ShowAndClear(Key);
                ShowAndClear(Password);
                ShowAndClear(CfPassword);
            } else {
                Login.show();
                HideAndClear(Key);
                HideAndClear(Password);
                HideAndClear(CfPassword);

            }
            if (resetCaptcha) grecaptcha.reset();
            this.CheckInputs();
        }

    }



    function isCaptchaChecked() {
        return grecaptcha && grecaptcha.getResponse().length !== 0;
    }

    var restorPanel;

    function checkInputs() {
        restorPanel.CheckInputs();
    }

    $(document).ready(function () {
        restorPanel = new RestorePanel($("#login"), $("#password"),
$("#password_confirm"), $("#key"), $("#restore"));
        restorPanel.SetStatus(false, false);
        $("#login").bind("input", function () {
            restorPanel.CheckInputs(restorPanel);
        });
```

```
        $("#password").bind("input", function () {
            restorPanel.CheckInputs(restorPanel);
        });

        $("#password_confirm").bind("input", function () {
            restorPanel.CheckInputs(restorPanel);
        });
    });

</script>

<div class="Panel">
    <div class="DataFields">
        <input id="login" type="text" placeholder="Имя
аккаунта/Email/Телефон" />
        <input id="key" type="text" placeholder="Проверочный код" />
        <input id="password" type="password" placeholder="Новый пароль" />
        <input id="password_confirm" type="password" placeholder="Повторите
пароль" />
    </div>
    <div class="g-recaptcha"  data-callback="checkInputs" data-
sitekey="@Html.Raw(CaptchaInfo.Key)"></div>
    <text id="Message" class="ErrorMessage">  </text>
    <input class="Button" id="restore" type="button" onclick="TryRestore()"
value="Отправить" />
</div>
```

Файл Config.cshtml

```
@model Education.Models.Config

<div class="Description">
    <div class="Name">
        Конфигурация сайта
    </div>
</div>
<div class="Panel">
    <h3>
        favicon
    </h3>
    @using (Html.BeginForm("ChangeIcon", "Admin", FormMethod.Post))
    {
        <img src="@Model.Icon" />
        <div class="DataFields">
            <input type="file" />
            <input type="submit" value="Изменить favicon" />
        </div>
    }
    @using (Html.BeginForm("ChangeConfig", "Admin", FormMethod.Post))
    {
        <h3>
            Название учреждения
        </h3>
        <div class="DataFields">
            @Html.EditorFor(x => x.Name)
        </div>
        <h3>
            Строка подключения
        </h3>
        <div class="DataFields">
            @Html.EditorFor(x => x.ConnString)
            <input type="submit" value="Сохранить" />
```

```
        </div>
    }
</div>
```

Файл Index.cshtml

```
<div class="Description">
    <div class="Name">
        Админ панель
    </div>
</div>
<div class="AdminMenu">
    @Html.ActionLink("Конфигурация", "Config", "Admin")
    @Html.ActionLink("Пользователи", "Users", "Admin")
    @Html.ActionLink("Страницы", "All", "Page")
</div>
```

Файл SearchUsers.cshtml

```
@model IEnumerable<Education.BLL.DTO.User.AdminUserInfoDTO>

<table class="table">
    <thead>
        <tr>
            <th>
                Аватар
            </th>
            <th>
                Id
            </th>
            <th>
                Login
            </th>
            <th>
                ФИО
            </th>
            <th>
                Email
            </th>
            <th>
                Подверждение Email
            </th>
            <th>
                Телефон
            </th>
            <th>
                Подверждение телефона
            </th>
            <th>
                Метод авторизации
            </th>
            <th>
                Уровень привилегий
            </th>
            <th>
                Заблокирован
            </th>
            <th>
                Причина блокировки
            </th>
            <th>
                Время окончания блокировки
            </th>
        </tr>
    </thead>
```

```
    <tbody>
@foreach (var item in Model) {
        <tr>
            <td>
                <img class="Avatar" src="@item.Avatar" />
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.Id)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.Login)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.name)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.email)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.emailConfirm)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.phone)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.phoneConfirm)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.authType)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.Level)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.Banned)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.BanReason)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.BanTime)
            </td>
            <td>
                @Html.ActionLink("Редактировать", "UserInfo", new { id =
item.Id })
            </td>
        </tr>
}
    </tbody>
</table>
```

Файл UserInfo.cshtml

```
@model Education.BLL.DTO.User.AdminUserInfoDTO

@{
    Microsoft.AspNetCore.Html.IHtmlContent Option(int val, int selval, string
text)
    {
        return Html.Raw(String.Format("<option value=\"{0}\"" + (val ==
selval ? "selected" : "") + ">{1}</option>", val, text));
    }
}
```

```
<script src="/lib/jquery/dist/jquery.js"></script>

<script>
    function Extra(val, name, cond, clearf) {
        if (cond(val))
            $("#Extra" + name).show();
        else {
            $("#Extra" + name).hide();
            clearf();
        }

    }

    $(document).ready(function () {
        var phonef = () => Extra($("#phone").val(), "Phone", function (text)
{ return text != ""; }, function () { $("#phoneConfirm").prop('checked',
false); });
        var emailf = () => Extra($("#email").val(), "Email", function (text)
{ return text != ""; }, function () { $("#emailConfirm").prop('checked',
false); });
        var banf = () => Extra($("#Banned").prop('checked'), "Ban", function
(val) { return val; }, function () { $("#BanReason").val("");
$("#BanTime").val(""); });
        phonef();
        emailf();
        banf();
        $("#phone").on("input", function () {
            phonef();
        });
        $("#email").on("input", function () {
            emailf();
        });
        $("#Banned").change(function () {
            banf();
        });
    });
</script>
<div class="Navigation">
    @Html.ActionLink("Админ панель", "Index", "Admin")>>
    @Html.ActionLink("Пользователи", "Users", "Admin")>>
    @Html.ActionLink(Model.name, "UserInfo", "Admin", new { id = Model.Id })
</div>
<div class="Description">
    <div class="Name">
        Пользователь
    </div>
</div>
<div class="Panel">
    @using (Html.BeginForm("EditUser", "Admin", FormMethod.Post))
    {
        <input type="hidden" name="Id" value="@Model.Id" />
        <h3>Аватар</h3>
        <div class="DataFields">
            <img class="Avatar" src="@Model.Avatar" />
            @Html.TextBox("Avatar", Model.Avatar)
        </div>
        <h3>Login</h3>
        <div class="DataFields">
            @Html.TextBox("Login", Model.Login)
        </div>
        <h3>ФИО</h3>
        <div class="DataFields">
            @Html.TextBox("name", Model.name)
```

```
        </div>
        <h3>Уровень пользователя</h3>
        <div class="DataFields">
            <select name="Level">
                @Option(0, Model.Level, "Пользователь")
                @Option(1, Model.Level, "Модератор")
                @Option(2, Model.Level, "Админ")
            </select>
        </div>
        <h3>Email</h3>
        <div class="DataFields">
            @Html.TextBox("email", Model.email)
            <span id="ExtraEmail">
                <label>Подтвержден</label>
                @Html.CheckBox("emailConfirm", Model.emailConfirm == true)
            </span>
        </div>
        <h3>Телефон</h3>
        <div class="DataFields">
            @Html.TextBox("phone", Model.phone)
            <span id="ExtraPhone">
                <label>Подтвержден</label>
                @Html.CheckBox("phoneConfirm", Model.phoneConfirm == true)
            </span>
        </div>
        <h3>Метод авторизации</h3>
        <div class="DataFields">
            <select name="authType">
                @Option(0, (int)Model.authType, "Обычный")
                @Option(1, (int)Model.authType, "Email")
                @Option(2, (int)Model.authType, "Телефон")
            </select>
        </div>

        <h3>Блокировка</h3>
        <div class="DataFields">
            <label>Заблокирован</label>
            @Html.CheckBox("Banned", Model.Banned)
            <span id="ExtraBan">
                <h4>Причина</h4>
                @Html.TextBox("BanReason", Model.BanReason)
                <h4>Дата разблокировки</h4>
                @Html.TextBox("BanTime", Model.BanTime)
            </span>
        </div>
        <div class="DataFields">
            <input type="submit" value="Сохранить" />
        </div>
    }
    <div class="DataFields">
        @using (Html.BeginForm("ClearUserSessions", "Admin",
FormMethod.Post))
        {
            <input type="hidden" value="@Model.Id" />
            <input type="submit" value="Сбросить все сессии" />
        }
    </div>
</div>
```

Файл Users.cshtml

```
<script src="/lib/jquery/dist/jquery.js"></script>

<script>
```

```
    function LoadUsers(name) {
        var info;
        if (name == "")
            info = {};
        else info = {
            search: name
        };

        $.ajax({
            url: '@Url.Action("SearchUsers","Admin")',
            method: 'POST',
            data: info,
            success: function (data) {
                ShowUsers(data);
            },
            error: function (jqxhr, status, errorMsg) {
                ShowError();
            }
        });

    }

    function ShowUsers(data) {
        $("#Result").html(data);
        $("#ResultMessage").text("");
    }

    function ShowError() {
        $("#Result").html("");
        $("#ResultMessage").text("Ошибка загрузки");

    }
    $(document).ready(function () {
        $("#Send").click(function () {
            LoadUsers($("#Name").val());
        });
    });
</script>

<div class="Navigation">
    @Html.ActionLink("Админ панель", "Index", "Admin")>>
    @Html.ActionLink("Пользователи", "Users", "Admin")
</div>
<div class="Description">
    <div class="Name">
        Пользователи
    </div>
</div>

<div class="SearchPanel">
    <input id="Name" placeholder="Поиск пользователя"/>
    <input type="button" value="Поиск" id="Send"  />
</div>
<div class="ResultMessage" id="ResultMessage">

</div>
<div class="Result" id="Result">

</div>
```

Файл Control.cshtml

```
@model Education.BLL.DTO.Pages.NoteEditDTO
```

```
@{
    string Name = "", Text = "", Action = "Create", ButtonText = "Создать";
    string Preview = "";
    bool Published = false;
    int Id = -1;
    if (Model != null)
    {
        Id = Model.Id;
        Preview = Model.Preview;
        Name = Model.Name;
        Text = Model.Text;
        Action = "Edit";
        ButtonText = "Сохранить";
        Published = Model.Published;
    }
}

<script src="/lib/jquery/dist/jquery.js"></script>

<img class="BlogPreview" id="Preview" src="@Preview" />
<input type="button" onclick="ShowArea()" value="Сменить превью" />

@Html.Partial("~/Views/ImageManager/Index.cshtml")

<script type="text/javascript" src="~/editor/nicEdit.js"></script>

<script type="text/javascript">
    bkLib.onDomLoaded(function () {
        new nicEditor({ iconsPath: '/nicEditorIcons.gif', fullPanel: true
}).panelInstance('TextArea');
    });
</script>

<script>
    var OnSelect = function (url) {
        $("#Preview").attr("src", url);
        $("#SendPreview").attr("value", url);
        HideArea();
    }
</script>

@using (Html.BeginForm(Action, "Blog", FormMethod.Post))
{
    /*
     public int Id { get; set; }
        public string Preview { get; set; }
        public string Name { get; set; }
        public string Text { get; set; }
        public bool Published { get; set; }
     */
    @Html.Hidden("Id", Id)
    <input type="hidden" name="Preview" id="SendPreview" value="@Preview" />
    <input type="text" name="Name" placeholder="Название страницы"
value="@Name" />
    <textarea cols="50" id="TextArea" name="Text" style="width:100%;
height:300px">@Text</textarea>
    <label>
        Опубликовать
        @Html.CheckBox("Published", Published)
    </label>
    <input type="submit" value="@ButtonText" />
}
```

```
@if (Model != null)
{
    @using (Html.BeginForm("Remove", "Blog", FormMethod.Post))
    {
        @Html.Hidden("Id", Id)
        <input type="submit" value="Удалить запись" />
    }
}
```

Файл Index.cshtml

```
@model Education.BLL.DTO.Pages.NoteDTO

<div class="Note">
    <div class="Description">
        <div class="Name">
            @Html.ActionLink(Model.Name, "Index", "Blog", new { id = Model.Id
})
        </div>
        <div class="Edit">
            @if (Model.Access.CanUpdate)
            {
                @Html.ActionLink("Редактировать", "Control", "Blog", new { id
= Model.Id })
            }
        </div>
    </div>
    <img class="Prev" src="@Model.Preview" />
    <div class="Text">
        @Html.Raw(Model.Text)
    </div>
    <div class="Edited">@Model.Owner.Name, @Model.Time</div>
</div>
```

Файл Page.cshtml

```
@model Education.BLL.DTO.Pages.BlogDTO

<div class="Description">
    <div class="Name">
        @Html.ActionLink("Блог", "Page", "Blog")
    </div>
    <div class="Edit">
        @if (Model.CanCreate)
        {
            @Html.ActionLink("Создать запись", "Control", "Blog")
        }
    </div>
</div>
@foreach(var note in Model.Notes)
{
    @Html.Partial("~/Views/Blog/Index.cshtml", note)
    <hr />
}
@await Component.InvokeAsync("Paging", new { url = "/Blog/Page/", page =
Model.Page, pages = Model.Pages })
```

Файл Index.cshtml

```
@model string

<!DOCTYPE html>

<html>
<head>
    <meta name="viewport" content="width=device-width" />
```

```
    <title>Ошибка сервера</title>
</head>
<body>
    <h1>@Model</h1>
    <a href="/"><h2>Вернутся на главную страницу</h2></a>
</body>
</html>
```

Файл All.cshtml

```
@model Education.BLL.DTO.Forum.ForumDTO

@{
    ViewData["Title"] = "Группы";
}

<div class="Groups">
    <div class="Description">
        <div class="Name">
            @Html.ActionLink("Группы", "Index", "Home")
        </div>
        <div class="Edit">
            @if (Model.CanCreateGroup)
            {
                @Html.ActionLink("Создать группу", "Control", "Group")
            }
        </div>
    </div>
    <div class="All">
        @foreach (var group in Model.Groups)
        {
            var id = new { id = group.Id };
            <div class="Group">
                <img class="Logo" src="@group.Logo" />
                <div class="Info">
                    @if (group.Access.CanRead)
                    {
                        <div class="Name">
                            @Html.ActionLink("Группа " + group.Name, "Index",
"Group", id)
                        </div>
                    }
                    else
                    {
                        <div class="Name">Группа @group.Name</div>
                    }
                    <div>
                        @(group.Open ? "Открытая группа" : "Закрытая группа")

                        @if (group.Access.CanUpdate)
                        {
                            <strong>
                                @Html.ActionLink("Редактировать", "Control",
"Group", id)
                            </strong>
                        }
                    </div>
                    <div>
                        <strong>
                            @if (!group.Open && group.Status != null)
                            {
                                @if (!group.Status.Member)
                                {
                                    if (!group.Access.CanRead)
```

```
                                    {
                                        @Html.ActionLink("Подать заявку",
"Enter", "Group", id)
                                    }
                                }
                                else
                                {
                                    <text>Роль: </text>
                                    if (group.Status.Status ==
Education.DAL.Entities.UserGroupStatus.member)
                                    {
                                        <text>Участник</text>
                                    }
                                    else if (group.Status.Status ==
Education.DAL.Entities.UserGroupStatus.owner)
                                    {
                                        <text>Модератор</text>
                                    }
                                    else if (group.Status.Status ==
Education.DAL.Entities.UserGroupStatus.request)
                                    {
                                        <text>Отправлена заявка</text>
                                    }
                                    @Html.ActionLink("Покинуть группу",
"Leave", "Group", id)
                                }
                            }
                        </strong>
                    </div>

                </div>
            </div>

        }
    </div>
</div>
```

Файл Control.cshtml

```
@model Education.BLL.DTO.Forum.GroupDTO

@{
    string Desctiprion, Action, ButtonText;

    if(Model == null)
    {
        Desctiprion = "Создать группу";
        Action = "/Group/Create";
        ButtonText = "Создать";
    }
    else
    {
        Desctiprion = "Редактировать группу " + Model.Name;
        Action = "/Group/Update";
        ButtonText = "Сохранить";

    }
    string Logo = Model != null ? Model.Logo : "";
}

<!DOCTYPE html>

<html>
<head>
```

```
    <meta name="viewport" content="width=device-width" />
    <script src="/lib/jquery/dist/jquery.js"></script>
    <title>View</title>
</head>
<body>
    <h4>@Desctiprion</h4>

    <img class="GroupLogo" id="Logo" src="@Logo" />
    <input type="button" onclick="ShowArea()" value="Сменить логотип" />

    <form action="@Action" method="post">
        <input type="hidden" name="Id" value="@(Model != null ? Model.Id : -
1)" />
        <input type="hidden" name="Logo" id="SendLogo" value="@Logo" />
        <input type="text" name="Name" placeholder="Имя группы"
value="@(Model != null ? Model.Name : "")" />
        <label><input type="checkbox" name="Open" @(Model != null &&
Model.Open ? "checked" : "") value="true" /> Открытая группа</label>
        <input type="submit" value="@ButtonText" />
    </form>

    @if (Model != null && Model.Access.CanDelete)
    {
        <form action="/Group/Delete" method="post">
            <input type="hidden" name="Id" value="@(Model != null ? Model.Id
: -1)" />
            <input type="submit" value="Удалить" />
        </form>
    }

    @Html.Partial("~/Views/ImageManager/Index.cshtml")

    <script>
        var OnSelect = function (url) {
            $("#Logo").attr("src", url);
            $("#SendLogo").attr("value", url);
            HideArea();
        }
    </script>

</body>



</html>
```

Файл Index.cshtml

```
@model Education.BLL.DTO.Forum.GroupDTO

@{
    var id = new { id = Model.Id };
}

<head>
    <meta name="viewport" content="width=device-width" />
    <title>Группа @Model.Name</title>
</head>

<div class="Navigation">
    @await Component.InvokeAsync("Navigation", Model)
</div>
<div class="Description">
    <div class="Name">
```

```
        @Html.ActionLink("Группа " + Model.Name, "Index", "Group", id)
    </div>
    <div class="Edit">
        @if (Model.Access.CanUpdate)
        {
            @Html.ActionLink("Редактировать", "Control", "Group", id)
        }
        @if (Model.Access.CanControlUsers)
        {
            @Html.ActionLink("Пользователи", "Users", "Group", id)
        }
        @if (Model.Access.CanCreateElements)
        {
            @Html.ActionLink("Добавить секцию", "Add", "Section", id)
        }
    </div>
</div>
<div class="Sections">
    @foreach (var section in Model.Sections)
    {
        @Html.Partial("~/Views/Section/View.cshtml", section)
    }
</div>
```

Файл Users.cshtml

```
@model UsersOfGroup

<!DOCTYPE html>

<html>
<head>
    <meta name="viewport" content="width=device-width" />
    <title>Users</title>
</head>
<body>
<div class="UserList">
    <h1>Запросы:</h1>
    @foreach(var user in Model.Data.Where(x => x.Status ==
Education.DAL.Entities.UserGroupStatus.request))
    {
        <div class="User">
            @await Component.InvokeAsync("UserGroup", new { DTO = user,
GroupId = Model.GroupId, inGroup = false })
        </div>
    }
    <h1>Участники:</h1>
    @foreach(var user in Model.Data.Where(x => x.Status !=
Education.DAL.Entities.UserGroupStatus.request))
    {
        <div class="User">
            @await Component.InvokeAsync("UserGroup", new { DTO = user,
GroupId = Model.GroupId, inGroup = true })
        </div>
    }
</div>
</body>
</html>
```

Файл Index.cshtml

```
@model Education.Models.MainPage

@{
    ViewData["Title"] = "Главная";
```

```
}

@Html.Partial("~/Views/Group/All.cshtml", Model.ForumDTO)

@Html.Partial("~/Views/Blog/Page.cshtml", Model.BlogDTO)
```

Файл Images.cshtml

```
@model IEnumerable<Education.BLL.DTO.ImageInfoDTO>


<div class="Images">
    @{
        int i = 0;
        @foreach (var image in Model)
        {
            <img id="@i" src="@image.Path" onclick='Select(@i);'/>
            i++;
        }
    }
</div>
```

Файл Index.cshtml

```
@*
    For more information on enabling MVC for empty projects, visit
http://go.microsoft.com/fwlink/?LinkID=397860
*@

<script>
    var load = false;

    function StartLoad() {
        load = true;
        $("#ImageManager").hide();
        $("#LoadCounter").show();
    }

    function EndLoad() {
        load = false;
        $("#ImageManager").show();
        $("#LoadCounter").hide();
    }

    function IsLoad() {
        return load;
    }

    function LoadFiles(files) {
        if (IsLoad()) return;
        StartLoad();
        $.ajax({
            xhr: function () {
                var xhr = new window.XMLHttpRequest();
                // прогресс загрузки на сервер
                xhr.upload.addEventListener("progress", function (evt) {
                    if (evt.lengthComputable) {
                        var percentComplete = evt.loaded / evt.total * 100;
                        var percent = percentComplete.toFixed(0) + "%";
                        $("#LoadCounter").css("width", percent);
                        $("#LoadCounter").text(percent);
                        console.log(percentComplete);
                    }
                }, false);
                return xhr;
```

```
        },
        url: '/ImageManager/Load',
        method: 'POST',
        data: files,
        processData: false,
        contentType: false,
        success: function (data) {
            LoadImages();
            EndLoad();
        },
        error: function (jqxhr, status, errorMsg) {
            Message("Ошибка загрузки");
            EndLoad();
        }

    });
}

function Message(text) {
    $("#Message").text(text);

}

function LoadImages() {
    $.ajax({
        url: '/ImageManager/Images',
        method: 'Get',
        success: function (data) {
            $("#Images").html(data);
        }
    });
}

$(document).ready(function () {
    ImageManager.ondragover = function () {
        ImageManager.style.backgroundcolor = "aliceblue";
        return false;
    };

    ImageManager.ondragleave = function () {
        ImageManager.style.backgroundcolor = "";
        return false;
    };

    ImageManager.ondrop = function (event) {
        event.preventDefault();
        var formData = new FormData();
        var files_list = event.dataTransfer.files;
        for (var i = 0; i < files_list.length; i++) {
            formData.append('file', files_list[i]);
        }
        LoadFiles(formData);
    };

    SelectFile.onchange = function () {
        var files_list = SelectFile.files;
        var formData = new FormData();
        for (var i = 0; i < files_list.length; i++) {
            formData.append('file', files_list[i]);
        }
        LoadFiles(formData);
    };
```

```
        LoadImages();

    });

    var Selected = "";
    var LastSelect = null;
    function Select(id) {
        if (LastSelect != null) {
            LastSelect.removeClass("SelectedImage");
        }
        LastSelect = $("#" + id);
        LastSelect.addClass("SelectedImage");
    }

    function DeleteImage() {
        if (LastSelect == null) return;
        $.ajax({
            url: '/ImageManager/Delete',
            method: 'POST',
            data: { path: LastSelect.attr('src') },
            success: function (data) {
                LoadImages();
            },
            error: function (jqxhr, status, errorMsg) {
                Message("Ошибка удаления");
            }
        });
    }

    function ShowArea() {
        $("#Area").show();

    }

    function HideArea() {
        $("#Area").hide();

    }

    function Cls() {
        HideArea();
    }

    function Del() {
        DeleteImage();
    }

    function Sel() {
        if (OnSelect == null) return;
        if (LastSelect == null) return;
        OnSelect(LastSelect.attr('src'));
    }

</script>
<div class="DarkSpace" id="Area" hidden>
    <div class="SelectImage">
        <div class="LoadCounter" id="LoadCounter" hidden></div>

        <div class="ImageManager" id="ImageManager">
            <div class="Label">Перетащите файлы сюда</div>
            <label class="SelectButton">
                <input type="file" id="SelectFile" name="Select"
multiple="multiple" />
```

```
                Выбрать файлы
            </label>
        </div>
        <div class="ErrorMessage" id="Message"></div>
        <hr />
        <div id="Images">

        </div>
        <div class="ImagesControl">
            <input type="button" onclick="Cls()" value="Отмена" />
            <input type="button" onclick="Del()" value="Удалить" />
            <input type="button" onclick="Sel()" value="Выбрать" />
        </div>
    </div>
</div>
```

Файл All.cshtml

```
@model IEnumerable<Education.BLL.DTO.Pages.PageInfo>

@{
        string GenerateMap(Education.BLL.DTO.Pages.PageInfo page)
        {
            string res = string.Format("<div class=\"Page\">--{0} <a
href=\"{1}\">Редактировать</a><div class=\"Childs\">", page.Name,
Url.Action("Control", "Page", new { id = page.Id }));
        foreach(var child in page.Childs)
            res += GenerateMap(child);
        return res + "</div></div>";
    }

    string Map()
    {
        var res = "";
        foreach (var page in Model)
            res += GenerateMap(page);
        return res;

    }
}

<div class="Navigation">
    @Html.ActionLink("Админ панель", "Index", "Admin")>>
    @Html.ActionLink("Страницы", "All", "Page")
</div>
<div class="Description">
    <div class="Name">
        Все Страницы
    </div>
    <div class="Edit">
        @Html.ActionLink("Создать страницу","Control","Page")
    </div>
</div>
<div class="AllPages">
    @Html.Raw(Map())
</div>
```

Файл Control.cshtml

```
@model Education.Models.PageControl

@{
    string Name = "", Text = "", Action = "Create", ButtonText = "Создать";
    bool Published = false;
    int Id = -1;
```

```
    int? ParrentId = -1;
    if (Model.PageDTO != null)
    {
        Id = Model.PageDTO.Id;
        ParrentId = Model.PageDTO.ParentId;
        Name = Model.PageDTO.Name;
        Text = Model.PageDTO.Text;
        Action = "Edit";
        ButtonText = "Сохранить";
        Published = Model.PageDTO.Published;
    }
}

<script type="text/javascript" src="~/editor/nicEdit.js"></script>
<script type="text/javascript">
    bkLib.onDomLoaded(function () {
        new nicEditor({ iconsPath: '/nicEditorIcons.gif', fullPanel: true
}).panelInstance('Area');
    });
</script>

@using (Html.BeginForm(Action, "Page", FormMethod.Post))
{
    @Html.Hidden("Id", Id)
    <input type="text" name="Name" placeholder="Название страницы"
value="@Name" />
    <textarea cols="50" id="Area" name="Text" style="width:100%;
height:300px">@Text</textarea>
    <label>
        Опубликовать
        @Html.CheckBox("Published", Published)
    </label>
    <label>
        Родительская страница
        @await Component.InvokeAsync("PageMap", new { map = Model.Map, name =
"ParentId", SelectedId = ParrentId, Id = Id })
    </label>
    <input type="submit" value="@ButtonText" />
}

@if (Model.PageDTO != null)
{
    @using (Html.BeginForm("Remove", "Page", FormMethod.Post))
    {
        @Html.Hidden("Id", Id)
        <input type="submit" value="Удалить страницу" />
    }
}
```

Файл Index.cshtml

```
@model Education.BLL.DTO.Pages.PageDTO

<div class="Navigation">
    @await Component.InvokeAsync("Navigation", Model)
</div>
<div class="Page">
    <div class="Description">
        <div class="Name">
            @Html.ActionLink(Model.Name, "Index", new { id = Model.Id })
        </div>
        <div class="Edit">
            @if (Model.Access.CanUpdate)
            {
```

```
                @Html.ActionLink("Редактировать", "Control", "Page", new { id
= Model.Id })
            }
        </div>
    </div>
    <div class="Text">
        @Html.Raw(Model.Text)
    </div>
    @if (Model.ChildPages != null && Model.ChildPages.Any())
    {
        <div class="ChildPages">
            <text>Допонительная информация:</text>
            <ol>
                @foreach (var page in Model.ChildPages)
                {
                    <li>@Html.ActionLink(page.Name, "Index", new { id =
page.Id })</li>
                }
            </ol>
        </div>
    }
</div>
```

Файл Menu.cshtml

```
@model Education.BLL.DTO.Pages.PagesDTO


<div class="Description">
    <div class="Name">
        @Html.ActionLink("Меню","Index","Home")
    </div>
    <div class="Edit">
        @if (Model.CanCreate)
        {
            @Html.ActionLink("Добавить страницу","Control","Page")
        }
    </div>
</div>
```

Файл Index.cshtml

```
@model Education.BLL.DTO.User.UserProfileDTO

@{
    ViewData["Title"] = "Личный кабинет";
}

<div class="Description">
    <div class="Name">
        Личный кабинет
    </div>
</div>

@Html.Partial("ProfilePanel", Model);
```

Файл ProfilePanel.cshtml

```
@model Education.BLL.DTO.User.UserProfileDTO

@{
    string GetBool(bool? b)
    {
        return b == true ? "true" : "false";
    }
}
```

```
@using  Education.BLL.Logic;
@using Education.DAL.Entities;
@using Education.BLL.Services.UserServices.Interfaces;

<script src="/lib/jquery/dist/jquery.js"></script>

<script>
    var Patterns = {};
    Patterns["email"] =
"@Html.Raw(regRegularExpressions.EmailRegex.Replace("\\","\\\\"))";
    Patterns["phone"] =
"@Html.Raw(regRegularExpressions.PhoneRegex.Replace("\\","\\\\"))";
    Patterns["password"] =
"@Html.Raw(regRegularExpressions.PasswordRegex.Replace("\\","\\\\"))";

    var sent = false;

    function AjaxF(to, info, succsessF, errorF) {
        if (sent) return;
        sent = true;
        $.ajax({
            url: to,
            method: 'POST',
            data: info,
            success: function (data) {
                succsessF(data);
                sent = false;
            },
            error: function (jqxhr, status, errorMsg) {
                errorF("Внутреняя ошибка сервера");
                sent = false;
            }

        });
    }

    function Contact(name, id, val, cnf, place, placeHolder, regExString,
btnValues, onClick, btnvls) {
        var Name = name;
        var Value = val;
        var Confirm = cnf;
        var Place = place;
        var RegEx = RegExp(regExString);
        var btnvls = btnValues;

        var Child;

        var html = "<input id=\"" + id + "\" type = \"text\" placeholder=\""
+ placeHolder + "\"disabled>";
        html += "<input id=\"" + id + "_ButtonSet\" type = \"button\"
value=\"" + btnvls[0] + "\">";
        html += "<input id=\"" + id + "_ButtonRemove\" type = \"button\"
value=\"" + btnvls[1] + "\">";
        html += "<input id=\"" + id + "_ButtonConfirm\" type = \"button\"
value=\"" + btnvls[2] + "\">";
        place.html(html);

        var SetButton = $("#" + id + "_ButtonSet");
        var RemoveButton = $("#" + id + "_ButtonRemove");
        var ConfirmButton = $("#" + id + "_ButtonConfirm");

        var Input = $("#" + id);
```

```
//---------------------------------
var redraw = function () {
    Input.val(Value);
    if (Value == null || Value == "") {
        SetButton.hide();
        RemoveButton.hide();
        ConfirmButton.hide();
        Input.removeAttr('disabled');
    } else if (Confirm) {
        SetButton.hide();
        RemoveButton.show();
        ConfirmButton.hide();
        Input.attr('disabled', 'disabled');
    } else {
        SetButton.hide();
        RemoveButton.show();
        ConfirmButton.show();
        Input.attr('disabled', 'disabled');
    }
    Input.css("border-color", "");
    if (Child) Child.redraw();
};

var onChange = function () {
    var val = Input.val();
    if (val == "") {
        Input.css("border-color", "");
        SetButton.hide();
    }
    else if (!RegEx.test(val)) {
        Input.css("border-color", "red");
        SetButton.hide();
    }
    else {
        SetButton.show();
        Input.css("border-color", "lightgreen");
    }
};

var reset = function () {
    Confirm = false;
    Value = "";
    redraw();
};

var set = function (val) {
    Confirm = false;
    Value = val;
    redraw();
};

var confirm = function () {
    Confirm = true;
    redraw();
};

var getInput = function () {
    return Input.val();
};

var IsConfirm = function () {
    return Confirm === true;
}
```

```
        var SetChild = function (child) {
            Child = child;
        }

        //---------------------------------
        var res = {
            getInput,
            redraw,
            onChange,
            reset,
            set,
            confirm,
            IsConfirm,
            SetChild,
            Name
        };
        //---------------------------------
        var ob = onClick.bind(res);
        //---------------------------------
        SetButton.click(function () { ob(0) });
        RemoveButton.click(function () { ob(1) });
        ConfirmButton.click(function () { ob(2) });
        //-------
        redraw();
        //-------
        Input.bind("input", onChange);
        return res;
    }

    function authTypeSelect(id, place, Contacts, selectedID, onClick) {
        var SelectedID = selectedID;
        var html = "<select id=\"" + id + "\">";
        html += "<option value=\"" + 0 + "\">Простой</option>";
        for (i = 0; i < Contacts.length; i++)
                html += "<option value=\"" + (i + 1) + "\">" +
Contacts[i].Name + "</option>";

        html += "<input id=\"" + id + "_ButtonSet\" type = \"button\"
value=\"" + btnvls[0] + "\">";
        html += "<input id=\"" + id + "_ButtonRemove\" type = \"button\"
value=\"" + btnvls[1] + "\">";
        //---------------
        place.html(html);
        //---------------
        var SetButton = $("#" + id + "_ButtonSet");
        var RemoveButton = $("#" + id + "_ButtonRemove");
        var Select = $("#" + id);
        //-------------------------------------------------
        var onchange = function () {
            if (Select.val() == SelectedID) SetButton.hide();
            else SetButton.show();
        }

        Select.change(onchange);
        //-------------------------------------------------
        var redraw = function () {
            if (SelectedID != 0 && !Contacts[SelectedID - 1].IsConfirm()) {
                SelectedID = 0;
            }
            Select.val(SelectedID);

            var options = Select.children('option');
```

```
            for (i = 1; i < options.length; i++) {
                if (Contacts[i - 1].IsConfirm()) {
                    $(options[i]).show();
                }
                else $(options[i]).hide();
            }

            if (SelectedID == 0) {
                onchange();
                RemoveButton.hide();
                Select.removeAttr('disabled');
                return;
            }
            SetButton.hide();
            RemoveButton.show();
            Select.attr('disabled', 'disabled');
        }

        var set = function (id) {
            SelectedID = id;
            redraw();
        };

        var selected = function () {
            return Select.val();
        };
        //----------------------
        var res = {
            redraw,
            selected,
            set
        };
        //-------------------------------------------------
        var ob = onClick.bind(res);

        SetButton.click(function () { ob(0) });
        RemoveButton.click(function () { ob(1) });

        redraw();
        onchange();
        return res;
    }

    function PasswordChange(id, place, ih, regExString, onClick) {
        var RegEx = RegExp(regExString);
        var html = "<input id=\"" + id + "_old\" type = \"password\"
placeholder=\"" + ih[0] + "\">";
        html += "<input id=\"" + id + "\" type = \"password\" placeholder=\""
+ ih[1] + "\">";
        html += "<input id=\"" + id + "_repeat\" type = \"password\"
placeholder=\"" + ih[2] + "\">";
        html += "<input id=\"" + id + "_set\" type = \"button\" value=\"" +
ih[3] + "\">";
        place.html(html);
        var SetButton = $("#" + id + "_set");
        var RepeatInput = $("#" + id + "_repeat");
        var Input = $("#" + id);
        var OldPasswordInput = $("#" + id + "_old");
        //--------------------------------
        var checkInput = function (inpt) {
            var val = inpt.val();
            if (val == "") {
```

```
            inpt.css("border-color", "");
            return false;
        }
        else if (!RegEx.test(val)) {
            inpt.css("border-color", "red");
            return false;
        }
        else {
            inpt.css("border-color", "lightgreen");
            return true;
        }

    }

    var clear = function () {
        OldPasswordInput.val("");
        Input.val("");
        RepeatInput.val("");
        onChange();
    }

    var onChange = function () {
        var res = checkInput(Input);
        if (Input.val() != RepeatInput.val()) {
            RepeatInput.css("border-color", "red");
            res = false;
        }
        else res == res && checkInput(RepeatInput);
        if (res && OldPasswordInput.val() != "") SetButton.show();
        else SetButton.hide();
    }
    //---------------------------------
    var res = {
        clear
    };
    //---------------------------------
    var ob = onClick.bind(res);

    Input.bind("input", onChange);
    RepeatInput.bind("input", onChange);
    SetButton.click(function () {
        ob(OldPasswordInput.val(), Input.val());
    });
    //----
    onChange();
    //----
    return res;
}



function Click(id) {
    alert(id);
}

var Email;
var Phone;
var Auth;
var Claims;
var Password;

var btnvls = ["Задать", "Сбросить", "Подтвердить"];
```

```
    var ih = ["Ваш старый пароль", "Новый пароль", "Повторите новый
пароль","Сменить"];

    var authTypes = [
        ["@AuthType.Simple", "Простой"],
        ["@AuthType.Email", "Email"],
        ["@AuthType.Phone", "Телефон"]
    ];


    //-----------------------------------------
    function ContactAction(contact, id, val, keyValue, setf, removef,
confirmf, errorf) {
        var action = "/api/";
        var data;
        var sf;
        if (id == 0) {
            action += "Set" + contact;
            data = { value: val, key: keyValue };
            sf = setf;
        }
        else if (id == 1) {
            action += "Remove" + contact;
            data = { key: keyValue };
            sf = removef;
        }
        else if (id == 2) {
            action += "Confirm" + contact;
            data = { key: keyValue };
            sf = confirmf;
        }
        AjaxF(action, data, sf, errorf);
    }
    //---------------------
    function AuthTypeAction(id, val, keyValue, setf, removef, errorf) {
        var action = "/api/";
        var data;
        var sf;
        if (id == 0) {
            action += "SetAuthType";
            data = { value: val, key: keyValue };
            sf = setf;
        }
        else if (id == 1) {
            action += "ResetAuthType";
            data = { key: keyValue };
            sf = removef;
        }
        AjaxF(action, data, sf, errorf);
    }
    //-----------------------------------------
    function ChangePasswordAction(op, np, key, setpf, errorf) {
        var action = "/api/SetPassword";
        var data = {
            oldPassword: op,
            newPassword: np,
            key: key
        };
        AjaxF(action, data, setpf, errorf);
    }
    //-----------------------------------------
    function SetPF(data, passwordChange) {
            var code = data.status;
```

```
            if (code == @((int)ConfirmCode.ContactNotFound)){
                ErrorMessage("Ошибка сервера (контакт не найден) обратитесь к
администратору!");
            }else if (code == @((int)ConfirmCode.Fail)){
                KeyMessage("Неправильный ключ!");
            }else if (code == @((int)ConfirmCode.KeySend)){
                KeyMessage("Ключ отправлен, срок действия: до " +
data.keyTime);
            }else if (code == @((int)ConfirmCode.NeedNewKey)){
                ErrorMessage("Срок действия ключа истек");
                passwordChange.clear();
            } else if (code == @((int)ConfirmCode.Success)){
                passwordChange.clear();
                Message("Данные успешно обновлены");
            }else if (code == @((int)ConfirmCode.UserNotFound)){
                ErrorMessage("Неправильный старый пароль");
            }
        };
    //----------------------------------------
    function RemoveF(data, contact) {
            var code = data.status;
            if (code == @((int)ConfirmCode.ContactNotFound)){
                ErrorMessage(contact.Name + " не найден!");
                contact.reset();
            }else if (code == @((int)ConfirmCode.Fail)){
                KeyMessage("Неправильный ключ!");
            }else if (code == @((int)ConfirmCode.KeySend)){
                KeyMessage("Ключ отправлен, срок действия: до " +
data.keyTime);
            }else if (code == @((int)ConfirmCode.NeedNewKey)){
                ErrorMessage("Срок действия ключа истек");
            } else if (code == @((int)ConfirmCode.Success)){
                contact.reset();
                Message("Данные успешно обновлены");
            }else if (code == @((int)ConfirmCode.UserNotFound)){
                ErrorMessage("Ошибка (пользователь не найден), обратитесь к
администратору");
            }
        };

        function ConfirmF(data, contact) {
            var code = data.status;
            if (code == @((int)ConfirmCode.AlreadyConfimed)){
                ErrorMessage(contact.Name + " уже потвержден!");
                contact.confirm();
            }else if (code == @((int)ConfirmCode.ContactNotFound)){
                ErrorMessage(contact.Name + " не найден!");
                contact.reset();
            }else if (code == @((int)ConfirmCode.Fail)){
                KeyMessage("Неправильный ключ!");
            }else if (code == @((int)ConfirmCode.KeySend)){
                KeyMessage("Ключ отправлен, срок действия: до " +
data.keyTime);
            }else if (code == @((int)ConfirmCode.NeedNewKey)){
                ErrorMessage("Срок действия ключа истек");
            }else if (code == @((int)ConfirmCode.Success)){
                contact.confirm();
                Message("Данные успешно обновлены");
            }else if (code == @((int)ConfirmCode.UserNotFound)){
                ErrorMessage("Ошибка (пользователь не найден), обратитесь к
администратору");
            }
        };
```

```
        function SetF(data, contact) {
            if (data == @((int)SetContactCode.AlreadySet)){
                ErrorMessage("Ошибка (Значение уже установленно),
перезагрузите страницу");
            } else if (data == @((int)SetContactCode.Succsess)){
                contact.set(contact.getInput());
                Message("Данные успешно обновлены");
            }else if (data == @((int)SetContactCode.UserNotFound)){
                ErrorMessage("Ошибка (пользователь не найден), обратитесь к
администратору");
            }else if (data == @((int)SetContactCode.WrongValue)){
                ErrorMessage("Ошибка (не подходяшее значение), обратитесь к
администратору");
            } else if (data == @((int)SetContactCode.AlreadyExists)){
                ErrorMessage(contact.Name + ": " + contact.getInput() + " уже
присутствует в системе, введите другое значение");
            }
        };
    //----------------------------------------
        function ResetAt(data, AuthTypeSelect) {
            var code = data.status;
            if (code == @((int)ConfirmCode.ContactNotFound)){
                ErrorMessage("Ошибка (контакт не найден), обратитесь к
администратору!");
            }else if (code == @((int)ConfirmCode.Fail)){
                KeyMessage("Неправильный ключ!");
            }else if (code == @((int)ConfirmCode.KeySend)){
                KeyMessage("Ключ отправлен, срок действия: до " +
data.keyTime);
            }else if (code == @((int)ConfirmCode.NeedNewKey)){
                ErrorMessage("Срок действия ключа истек");
            }else if (code == @((int)ConfirmCode.Success)){
                AuthTypeSelect.set(0);
                Message("Данные успешно обновлены");
            }else if (code == @((int)ConfirmCode.UserNotFound)){
                ErrorMessage("Ошибка (пользователь не найден), обратитесь к
администратору");
            }
        };

        function SetAt(data, AuthTypeSelect) {
            var code = data.status;
            if (code == @((int)ConfirmCode.ContactNotFound)){
                ErrorMessage("Необходимо сначала задать и потвердить данный
контакт!");
            }else if (code == @((int)ConfirmCode.Fail)){
                KeyMessage("Неправильный ключ!");
            }else if (code == @((int)ConfirmCode.KeySend)){
                KeyMessage("Ключ отправлен, срок действия: до " +
data.keyTime);
            }else if (code == @((int)ConfirmCode.NeedNewKey)){
                ErrorMessage("Срок действия ключа истек");
            } else if (code == @((int)ConfirmCode.Success)){
                AuthTypeSelect.set(AuthTypeSelect.selected());
                Message("Данные успешно обновлены");
            }else if (code == @((int)ConfirmCode.UserNotFound)){
                ErrorMessage("Ошибка (пользователь не найден), обратитесь к
администратору");
            }else if(code == @((int)ConfirmCode.NeedContact)){
                ErrorMessage("Необходимо сначала сбросить метод
авторизации.");
            }
```

```
    };
//----------------------------------------
var CurrAction;

function AuthTypeClickHelper(id, AuthTypeSelect, key) {
    CurrAction = function (keyv) {
        AuthTypeAction(id, AuthTypeSelect.selected(), keyv,
            function (data) {
                SetAt(data, AuthTypeSelect);
            },
            function (data) {
                ResetAt(data, AuthTypeSelect);
            },
            ErrorMessage
        );
    };
    CurrAction(key);
}

function ContactClickHelper(contactName, id, contact, key) {
    CurrAction = function (keyv) {
        ContactAction(contactName, id, contact.getInput(), keyv,
            function (data) {
                SetF(data, contact);
            },
            function (data) {
                RemoveF(data, contact);
            },
            function (data) {
                ConfirmF(data, contact);
            },
            ErrorMessage
        );
    };
    CurrAction(key);
}

function ChangePasswordHelper(op,np,key,changePassword) {
    CurrAction = function (keyv) {
        ChangePasswordAction(op, np, keyv,
            function (data) {
                SetPF(data, changePassword);
            },
            ErrorMessage
        )
    };
    CurrAction(key);
}
//-----------------------------
function Message(text) {
    HideDarkSpace();
    $("#Message").html("<font color=\"lightgreen\">" + text + "</font>");
}

function ErrorMessage(text) {
    HideDarkSpace();
    $("#Message").html("<font color=\"red\">" + text + "</font>");
}

function KeyMessage(text) {
    ShowDarkSpace();
    $('#keyInfo').text(text);
}
```

```
    //---------------------------


    function ClickEmail(id) {
        ContactClickHelper("Email", id, this, null);
    }

    function ClickPhone(id) {
        ContactClickHelper("Phone", id, this, null);
    }

    function ClickAuthType(id) {
        AuthTypeClickHelper(id, this, null);
    }

    function ClickPassword(oldPassword, newPassword) {
        ChangePasswordHelper(oldPassword, newPassword, null, this);
    }

    var ctext = [ "IP адресс","Дата входа","Браузер" ];



    $(document).ready(function () {

        Email = new Contact("Email", "eID",
"@Model.email",@GetBool(Model.emailConfirm), $("#EmailPanel"), "Ваш Email",
Patterns["email"], btnvls, ClickEmail, Auth);

        Phone = new Contact("Телефон", "pID",
"@Model.phone".replace("&#x2B;","+"),@GetBool(Model.phoneConfirm),
$("#PhonePanel"), "Ваш номер телефона", Patterns["phone"], btnvls,
ClickPhone, Auth);

        Auth = new authTypeSelect("aID", $("#AuthPanel"), [Email, Phone],
@((int)Model.authType), ClickAuthType);

        Email.SetChild(Auth);
        Phone.SetChild(Auth);

        Password = new PasswordChange("pwID", $("#PasswordPanel"), ih,
Patterns["password"], ClickPassword);
    });



    //-------------------------
    function KeyClick() {
        CurrAction($("#key").val());
        $("#key").val("");
    }
    //-------------------------
    function HideDarkSpace() {
        $(".DarkSpace").hide();
    }

    function ShowDarkSpace() {
        $(".DarkSpace").show();
    }
    //-------------------------
</script>

<div class="Panel">
```

```
    <h3>Аватар:</h3>
    <img class="Avatar" src="@Model.Avatar" />
    <form method="post" action="/Profile/SetAvatar" enctype="multipart/form-
data">
        <label class="SelectFile">
            <input type="file" name="file"/>
            Выбрать файл
        </label>
        <div class="DataFields">
            <input type="submit" value="Изменить аватар" />
        </div>
    </form>
    <div id="Message"></div>
    <h3>Email:</h3>
    <div class="DataFields" id="EmailPanel">

    </div>
    <hr />

    <h3>Телефон:</h3>
    <div class="DataFields" id='PhonePanel'>

    </div>
    <hr />

    <h3>Метод авторизации:</h3>
    <div class="DataFields" id='AuthPanel'>

    </div>
    <hr />

    <h3>Пароль:</h3>
    <div class="DataFields" id='PasswordPanel'>

    </div>
</div>
    <hr />
    <h3>Активные сессии:</h3>
    <div class="DataTable" id='ClaimsPanel'>
        <table>
            <tr><td>IP адрес</td><td>Время
авторизации</td><td>Браузер</td></tr>
            @foreach (var a in Model.Claims)
            {

<tr><td>@a.Ip</td><td>@a.LoginTime</td><td>@a.Browser</td></tr>

            }
        </table>
    </div>
    <br />
    <div class="DataFields">
        <div class="Panel">
            <input type="Button" value="Сбросить все сессии"
onclick="window.location='/api/ResetClaims';" />
        </div>
    </div>
    <div class="DarkSpace" id="DarkSpace" style="display: none">
        <div class="SecretKeyPanel">
            <div class="PanelLabel">
                <text>Подтверждение</text>
            </div>
            <div class="PanelBody">
```

```
                <text id="keyInfo"></text><br />
                <input id="key" type="text" />
                <div class="Buttons">
                    <input type="button" onclick="KeyClick()"
value="Потвердить" />
                    <input type="button" onclick="HideDarkSpace()"
value="Отмена" />
                </div>
            </div>
        </div>
    </div>
```

Файл Control.cshtml

```
@model Education.Models.SectionControl

@{
    string Desctiprion, Action, ButtonText;

    if (Model.SectionDTO == null)
    {
        Desctiprion = "Создать секцию";
        Action = "/Section/Create";
        ButtonText = "Создать";
    }
    else
    {
        Desctiprion = "Редактировать секцию " + Model.SectionDTO.Name;
        Action = "/Section/Update";
        ButtonText = "Сохранить";

    }
}

<!DOCTYPE html>

<html>
<head>
    <meta name="viewport" content="width=device-width" />
    <title>View</title>
</head>
<body>
    <h4>@Desctiprion</h4>

    <form action="@Action" method="post">

        <input type="hidden" name="Id" value="@(Model.SectionDTO != null ?
Model.SectionDTO.Id : -1)" />
        <input type="hidden" name="GroupId" value="@(Model.SectionDTO != null
? Model.SectionDTO.Route.GroupId : Model.GroupId)" />
        <input type="text" name="Name" placeholder="Название секции"
value="@(Model.SectionDTO != null ? Model.SectionDTO.Name : "")" />
        <label><input type="checkbox" name="Open" @(Model.SectionDTO != null
&& Model.SectionDTO.Open ? "checked" : "") value="true"/>
            Открытая секция
        </label>
        <input type="submit" value="@ButtonText" />
    </form>

    @if (Model.SectionDTO != null && Model.SectionDTO.Access.CanDelete)
    {
        <form action="/Section/Delete" method="post">
            <input type="hidden" name="Id" value="@(Model != null ?
Model.SectionDTO.Id : -1)" />
```

```
            <input type="submit" value="Удалить" />
        </form>
    }

</body>
</html>
```

Файл Index.cshtml

```
@model Education.BLL.DTO.Forum.SectionDTO

<div class="Navigation">
    @await Component.InvokeAsync("Navigation", Model)
</div>

@Html.Partial("View", Model)
```

Файл View.cshtml

```
@model Education.BLL.DTO.Forum.SectionDTO
@{
    var id = new { id = Model.Id };
}
<div class="Section">
    <div class="Description">
        <div class="Name">
            @Html.ActionLink(Model.Name, "Index", "Section", id)
        </div>
        <div class="Edit">
            <strong>
                @if (Model.Access.CanUpdate)
                {
                    @Html.ActionLink("Редактировать", "Control", "Section",
id)
                }
                @if (Model.Access.CanCreateElements)
                {
                    @Html.ActionLink("Создать тему", "Add", "Theme", id)
                }
            </strong>
        </div>
    </div>
    <div class="Body">
            @foreach (var theme in Model.Themes)
            {
                <div class="ThemePreview">
                    <a href="/Theme/Index/@theme.Id" class="ThemeInfo">
                        <div class="Name">@theme.Name</div>
                        <div class="Owner">Создал: @theme.Owner.Name</div>
                        <div class="Pages">Страниц: @theme.Pages</div>
                    </a>

                    <div class="LastMessage">
                        <strong><a
href="/Theme/Index/@theme.Id/@theme.Pages">Последнее сообщение:</a></strong>
                        <div
class="Owner">@theme.LastMessages.Owner.Name</div>
                        <div class="Time"> @theme.LastMessages.Time</div>
                    </div>
                </div>
            }
    </div>
</div>
```

Файл Error.cshtml

```
@model ErrorViewModel
@{
    ViewData["Title"] = "Error";
}

<h1 class="text-danger">Error.</h1>
<h2 class="text-danger">An error occurred while processing your request.</h2>

@if (Model.ShowRequestId)
{
    <p>
        <strong>Request ID:</strong> <code>@Model.RequestId</code>
    </p>
}

<h3>Development Mode</h3>
<p>
    Swapping to <strong>Development</strong> environment will display more
detailed information about the error that occurred.
</p>
<p>
    <strong>Development environment should not be enabled in deployed
applications</strong>, as it can result in sensitive information from
exceptions being displayed to end users. For local debugging, development
environment can be enabled by setting the
<strong>ASPNETCORE_ENVIRONMENT</strong> environment variable to
<strong>Development</strong>, and restarting the application.
</p>
```

Файл _Layout.cshtml

```
@inject Education.BLL.Services.ConfigService.Interfaces.IConfigService
ConfigService

<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>@ConfigService.Name - @ViewData["Title"]</title>
    <link rel="shortcut icon" href="@ConfigService.IconPath" type="image/x-
icon">

    <link rel="stylesheet" href="~/css/site.css" />
    <environment include="Development">
        <link rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.css"
/>
        <link rel="stylesheet" href="~/css/site.css" />
    </environment>
    <environment exclude="Development">
        <link rel="stylesheet"
href="https://ajax.aspnetcdn.com/ajax/bootstrap/3.3.7/css/bootstrap.min.css"
              asp-fallback-href="~/lib/bootstrap/dist/css/bootstrap.min.css"
              asp-fallback-test-class="sr-only" asp-fallback-test-
property="position" asp-fallback-test-value="absolute" />
        <link rel="stylesheet" href="~/css/site.min.css" asp-append-
version="true" />
    </environment>
</head>
<body>
    <div class="Header">
        <div class="Main">
            <div class="Name"><a href="/">@ConfigService.Name</a></div>
            @await Component.InvokeAsync("Profile")
```

```
        </div>
        <div class="Menu">
            @await Component.InvokeAsync("Menu")
        </div>
    </div>

    <div class="Content">
        @RenderBody()
    </div>

</body>
</html>
```

Файл _ValidationScriptsPartial.cshtml

```
<environment include="Development">
    <script src="~/lib/jquery-validation/dist/jquery.validate.js"></script>
    <script src="~/lib/jquery-validation-
unobtrusive/jquery.validate.unobtrusive.js"></script>
</environment>
<environment exclude="Development">
    <script
src="https://ajax.aspnetcdn.com/ajax/jquery.validate/1.14.0/jquery.validate.m
in.js"
            asp-fallback-src="~/lib/jquery-
validation/dist/jquery.validate.min.js"
            asp-fallback-test="window.jQuery && window.jQuery.validator"
            crossorigin="anonymous"
            integrity="sha384-
Fnqn3nxp3506LP/7Y3j/25BlWeA3PXTyT1l78LjECcPaKCV12TsZP7yyMxOe/G/k">
    </script>
    <script
src="https://ajax.aspnetcdn.com/ajax/jquery.validation.unobtrusive/3.2.6/jque
ry.validate.unobtrusive.min.js"
            asp-fallback-src="~/lib/jquery-validation-
unobtrusive/jquery.validate.unobtrusive.min.js"
            asp-fallback-test="window.jQuery && window.jQuery.validator &&
window.jQuery.validator.unobtrusive"
            crossorigin="anonymous"
            integrity="sha384-
JrXK+k53HACyavUKOsL+NkmSesD2P+73eDMrbTtTk0h4RmOF8hF8apPlkp26JlyH">
    </script>
</environment>
```

Файл Control.cshtml

```
@model Education.Models.ThemeControl

@{
    string Desctiprion, Action, ButtonText;
    string text = "";

    if (Model.ThemeDTO == null)
    {
        Desctiprion = "Создать тему";
        Action = "/Theme/Create";
        ButtonText = "Создать";
    }
    else
    {
        Desctiprion = "Редактировать тему " + Model.ThemeDTO.Name;
        Action = "/Theme/Update";
        ButtonText = "Сохранить";
        var fs = Model.ThemeDTO?.Messages?.FirstOrDefault();
        if (fs != null)
```

```
        {
            text = fs.Text;
        }
    }


}

<!DOCTYPE html>

<html>
<head>
    <meta name="viewport" content="width=device-width" />
    <title>View</title>
    <script type="text/javascript" src="~/editor/nicEdit.js"></script>
    <script type="text/javascript">
        bkLib.onDomLoaded(function () {
            new nicEditor({ iconsPath: '/nicEditorIcons.gif', fullPanel: true
}).panelInstance('Area');
        });
    </script>
</head>
<body>
    <h4>@Desctiprion</h4>
    <div>
        <form action="@Action" method="post">

            <input type="hidden" name="Id" value="@(Model.ThemeDTO != null ?
Model.ThemeDTO.Id : -1)" />
            <input type="hidden" name="SectionId" value="@(Model.ThemeDTO !=
null ? Model.ThemeDTO.Route.SectionId : Model.SectionId)" />
            <input type="text" name="Name" placeholder="Название темы"
value="@(Model.ThemeDTO != null ? Model.ThemeDTO.Name : "")" />
            <label>
                <input type="checkbox" name="Open" @(Model.ThemeDTO != null
&& Model.ThemeDTO.Open ? "checked" : "") value="true" />
                Открытая тема
            </label>
            <textarea cols="50" id="Area" name="Text" style="width:100%;
height:300px">@text</textarea>
            <input type="submit" value="@ButtonText" />
        </form>
    </div>
    @if (Model.ThemeDTO != null && Model.ThemeDTO.Access.CanDelete)
    {
        <form action="/Theme/Remove" method="post">
            <input type="hidden" name="Id" value="@(Model != null ?
Model.ThemeDTO.Id : -1)" />
            <input type="submit" value="Удалить" />
        </form>
    }

</body>
</html>
```

Файл Index.cshtml

```
@model Education.BLL.DTO.Forum.ThemeDTO

@{
    var paging = await Component.InvokeAsync("Paging", new { url =
"/Theme/Index/" + Model.Id + "/", page = Model.CurPage, pages = Model.Pages
});
}
```

```
<div class="Navigation">
    @await Component.InvokeAsync("Navigation", Model)
</div>
<div class="Theme">
    @foreach (var message in Model.Messages)
    {
        var id = new { id = message.Id };
        <div class="Message">
            <div class="Info">
                <div class="Name">@message.Owner.Name</div>
                <img class="Avatar" src="@message.Owner.AvatarPath" />
            </div>
            <div class="Body">
                <div class="MessageTime">
                    @message.Time
                </div>
                @if (message == Model.Messages.First())
                {
                    <div class="Text">@Html.Raw(message.Text)</div>
                    @if (Model.Access.CanUpdate)
                    {
                        <div class="Edit">
                            @Html.ActionLink("Редактировать", "Control",
"Theme", new { id = Model.Id })
                        </div>
                    }
                }
                else
                {
                    <div class="Text">@message.Text</div>
                    @if (message.LastEditor != null)
                    {
                        <div class="Edited">Изменено:
@message.LastEditor.Name, @message.LastEditTime</div>
                    }
                    @if (message.Access.CanUpdate)
                    {
                        <div class="Edit">
                            @Html.ActionLink("Редактировать", "Control",
"Message", id)
                        </div>
                    }

                }
            </div>
        </div>
    }

    @paging
    @if (Model.Access.CanCreateElements)
    {
        <div class="AddMessage">
            <form method="post" action="/Message/Create/">
                <input type="hidden" name="themeId" value="@Model.Id" />
                <textarea class="Text" name="Text"></textarea>
                <input type="submit" value="Отправить Сообщение" />
            </form>
        </div>
    }
</div>
```

Файл _ViewImports.cshtml

```
@using Education
@using Education.Models
@addTagHelper *, Microsoft.AspNetCore.Mvc.TagHelpers
```

Файл _ViewStart.cshtml

```
@{
    Layout = "_Layout";
}
```

Файл site.css

```
a, a:hover {
    color: #333;
    text-decoration: none;
}

.Menu {
    border: 0px;
    border-top: 1px;
    border-style: solid;
    border-color: white;
}

.Menu a{
    padding-right: 10px;
    padding-left: 10px;
    font-weight: 600;
    border: 0px;
    border-right: 1px;
    border-style: solid;
}

.CreateGroup {
    font-weight: 600;
}

.Page .ChildPages {
    padding-left: 20px;
    padding-top: 20px;
}

.Page .ChildPages ol {
    list-style-type: disc;
}

.Page .ChildPages a, .Page .ChildPages text {
    font-weight: 600;
}

.Page .Text {
    padding-left: 20px;
    padding-top: 10px;
}

.Paging {
    margin: 10px;
    text-align: right;
}

.Paging a{
```

```
    font-weight: 600;
    margin: 2px;
}

.Paging .Selected{
    color: blue;
}

.Header {
    padding-left: 10%;
    width: 100%;
    background-color: white;
}

.Header .Main{
    display: table;
    width: 100%;
}

.Header .Login{
    display: table-cell;
    text-align: right;
    padding-right: 30px;
    min-width: 400px;
}

.Header .Login text{
    padding-right: 10px;
    font-weight: 600;
}

.Menu a:hover, .AdminMenu a:hover {
    color: #e2ec54 !important;
}

.Header .Login a{
    padding-right: 10px;
}

.Header .Name{
    display: table-cell;
    font-size: 22px;
    margin-left: 10%;
}

.ThemeInfo .Name{
    font-weight: 600;
}

.ThemePreview:hover {
    background-color: #f0f6fb;
}

.Navigation {
    font-weight: 600;
}
```

```
.Header, .Description, .ThemePreview, .Navigation, .Message, .Message .Body
.Panel {
}

.Message .Body .Panel{
    width: 100%;
    text-align: right;
}

/* Wrapping element */
/* Set some basic padding to keep content from hitting the edges */
.Content {
    margin: 0 auto;
    width: 60%;
    margin-top: 20px;
}

.GroupLogo {
    width: 150px;
}

.DarkSpace .SelectImage {
    width: 90%;
    margin: 0 auto;
    background-color: white;
    padding: 10px;
    border-style: groove;
    margin-top: 100px;
}

.Message .Info{
    font-weight: 600;
}

.MessageEdit textarea{
    width: 100%;
    height: 150px;
    resize: none;
}

.Body .ThemePreview {
    margin-left: 25px;
    display: table;
    width: calc(100% - 25px);
    box-shadow: 1px 1px 6px 1px;
    margin-top: 5px;
    margin-bottom: 5px;
}

.ImageManager {
    height: 150px;
    margin: 0 auto;
    border-style: dashed;
    text-align: center;
    border-color: gray;
}

.ImageManager .Label{
```

```
    margin-top: 50px;
    font-size: 20px;
}

.ImageManager input[type="file"], .SelectFile input[type="file"] {
    display: none;
}

.ImageManager .SelectButton{
     border: 1px solid #ccc;
    display: inline-block;
    padding: 6px 12px;
    cursor: pointer;
}

.Images img{
    width: 150px;
    cursor: pointer;
}

.Message {
    display: table;
    width: 100%;
    box-shadow: 1px 1px 6px 1px;
}

.Message .Body{
    display: table-cell;
}

.Message .Info {
    width: 155px;
    text-align: center;
    display: table-cell;
    border-width: 0px;
    border-right-width: 1px;
    border-style: solid;
}

.Theme {
    margin-top: 5px;
}

.User{
    margin: 10px;
    text-align: center;
    width: 150px;
    display: inline-block;
}

.User .Info{
    font-weight: 600;
}

.User .Edit{
    margin-top: 3px;
}
```

```
.User input, .User select{
    width: 100%;
}



.SelectedImage {
    border-style: inset;
    border-width: 3px;
}

.ImagesControl {
    text-align: right;
    padding-top: 5px;
}

.ImagesControl input{
    background-color: white;
    border-style: solid;
}

.ImagesControl input :hover {
    background-color: gray;
    color: white;

}

.LoadCounter {
    text-align: center;
    font-weight: 600;
    color: white;
    background: linear-gradient(to left, #36ce5e, rgba(96, 243, 36, 0.47));
}

.Message .Info .Name {
    font-weight: 600;
    font-weight: 600;
    background-color: #4083ff;
    color: white;
}

.AddMessage .Text {
    width: 100%;
    height: 100px;
    resize: none;
}

.Message .Text {
    min-height: 160px;
    vertical-align: middle;
    margin: 10px;
}

.ThemePreview .ThemeInfo {
    display: table-cell;
    vertical-align: middle;
    padding-left: 10px;
}
```

```
.ThemePreview .LastMessage {
    text-align: right;
    margin-right: 10px;
}

.Description .Name a{
    color: white;
}

.Description .Name {
    color: white;
}

.Header {
    background-color: #4083ff;
    box-shadow: 1px 1px 6px 1px;
}

.Body .MessageTime {
    background-color: #4083ff;
    color: white;
    padding-right: 10px;
    text-align: right;
}

.Body .Edited {
    padding-left: 10px;
}

.Description .Edit {
    padding-right: 10px;
}

.Description .Edit a{
    color: white;
}

.Header a, .Header text{
    color: white !important;
}

.Description {
    display: table;
    width: 100%;
    background-color: #4083ff;
    box-shadow: 1px 1px 6px;
    padding-left: 10px;
    margin-top: 3px;
    margin-bottom: 3px;
}

.Description .Name {
    font-size: 20px;
    display: table-cell;
}

.Description .Edit a:not(:last-child){
```

```
    border-style: solid;
    border-width: 0px;
    border-color: lightgray;
    border-right-width: 1px;
    padding-right: 3px;
}



.Description .Edit {
    display: table-cell;
    text-align: right;
    font-weight: 600;

}

.Sections{
    padding-left: 25px;
}

.AddMessage {
    margin-top: 20px;
}

.Message .Body .Edit {
    float: right;
    font-weight: 600;
}

.Edited {
    text-align: right;
}

.Avatar {
    width: 150px;
    max-width: 150px;
    margin: 5px;
}

.Groups {
    margin-top: 15px;
}

.Groups .All{
    padding-left: 20px;
    padding-top: 10px;
}

.Group {
    position: relative;
    margin-bottom: 10px;
    display: inline-block;
    padding: 5px;
    vertical-align: top;
}

.Group img {
    height: 50px;
```

```
    position: absolute;
}

.Group .Info {
    display: inline-block;
    margin-left: 60px;
}

.Group .Info .Name {
    font-size: 26px;
}

.Panel .error{
    color: red;
}

.ErrorMessage{
    color: red;
    text-align: center;
}

.DataTable{
    width: 700px;
}

.DataTable table, .DataTable table tr, .DataTable table tr td {
    border-style: solid;
    text-align: center;
}

.DataTable table tr:first-child{
        font-weight: 600;
}

.DataFields    input[type="password"],    .DataFields    input[type="text"],
.DataFields input[type="email"], .DataFields select {
    background-color: #ffffff00;
    border-left: 0px;
    border-right: 0px;
    border-top: 0px;
    border-color: black;
    margin-bottom: 5px;
    outline: none;
    font-size: 18px;
    width: 100%;
}

.SelectFile{
    text-align: center;
    cursor: pointer;
}

.SelectFile,    .Button,    .DataFields    input[type="button"],    .DataFields
input[type="submit"], .DataFields input[type="file"] {
    width: 100%;
    font-size: 13px;
    padding: 6px 0 4px 10px;
    border: 1px solid #cecece;
```

```
    background: white;
    color: black;
    border-radius: 5px;
    margin-bottom: 10px;
}

.ErrorMessage{
    color: red;
}

.SecretKeyPanel {
    display: table;
    width: 300px;
    overflow: auto;
    margin: auto;
    position: absolute;
    top: 0;
    left: 0;
    bottom: 0;
    right: 0;
    background-color: white;
}

.SecretKeyPanel .PanelBody .Buttons input[type="button"]{
    position: absolute;
    width: 100px;
}

.SecretKeyPanel .PanelBody .Buttons input[type="button"]:nth-child(1){
    left: 10px;
}

.SecretKeyPanel .PanelBody input[type="text"]{
    width: 100%;
}

.SecretKeyPanel .PanelBody .Buttons input[type="button"]:nth-child(2){
    right: 10px;
}


.SecretKeyPanel .PanelLabel{
    width: 100%;
    background-color: #222222;
    color: white;
}

.SecretKeyPanel .PanelBody{
    margin: 10px;
    word-wrap: break-word;
    margin-bottom: 40px;
}

.PanelBody .Buttons {
    margin-top: 5px;
}

.PanelLabel text{
```

```
    margin-left: 5px;
}

.DarkSpace {
    position: fixed;
    width: 100%;
    height: 100%;
    background-color: #000000d4;
    z-index: 1;
    margin: auto;
    top: 0;
    bottom: 0;
    left: 0;
    right: 0;
}

.Panel{
    width: 300px;
    margin-left: 20px;
}

.AllPages .Page{
    font-weight: 600;
    margin-left: 20px;
    margin-bottom: 5px;
}

.AllPages .Page a {
    color: #4083ff;
}

.AdminMenu {
    margin-top: 10px;
    margin-left: 20px;
}

.AdminMenu a {
    font-weight: 600;
    color: white;
    background-color: #4083ff;
    padding: 10px;
}

/* Carousel */
.carousel-caption p {
    font-size: 20px;
    line-height: 1.4;
}

/* Make .svg files in the carousel display properly in older browsers */
.carousel-inner .item img[src$=".svg"] {
    width: 100%;
}

.Note {
    padding-left: 3%;
}
```

```
.Note .Prev{
    min-width: 100%;
    max-width: 100%;
}

/* QR code generator */
#qrCode {
    margin: 15px;
}

/* Hide/rearrange for smaller screens */
@media screen and (max-width: 767px) {
    /* Hide captions */
    .carousel-caption {
        display: none;
    }
}
```