Министерство науки и высшего образования 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 09.04.04 Программная инженерия

РАБОТА ПРОВЕРЕНА
Рецензент,

_____
«____»_____2019 г.

ДОПУСТИТЬ К ЗАЩИТЕ
Заведующий кафедрой, д.ф.-м.н., доцент
_____/ А.А. Замышляева
«____»_____2019 г.

Разработка системы электронного документооборота для сдачи отчетов по лабораторным и курсовым работам

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ
ЮУрГУ–09.04.04.2019.420.ПЗ ВКР

Руководитель работы, доцент
_____/А.К. Демидов
«____»_____2019 г.
Автор работы
Студент группы ЕТ-225
_____/ А.И. Подкопаев
«____»_____2019 г.
Нормоконтролер, ассистент
_____/Н.С. Мидоночева
«____»_____2019 г.

Челябинск
2019

АННОТАЦИЯ

Подкопаев А.И. Разработка системы электронного документооборота для сдачи отчетов по лабораторным и курсовым работам. – Челябинск: ЮУрГУ, ЕТ-225, 68 с., 24 ил., 1 табл., библиогр. список – 14 наим., 1 прил.

Цель данной работы – разработать web-приложение для сдачи курсовых и лабораторных работ. В рамках выпускной квалификационной работы был проведен обзор существующих систем электронного оборота и на основе полученной информации выбран метод реализации последовательного движения документа. Для описания движения документа предложена нотация шаблона маршрута на основе XML. Разработана база данных и спроектирован пользовательский интерфейс.

Первый раздел работы посвящен терминам и понятиям, используемым в системах электронного документооборота, анализу и сравнению существующих систем, постановке задачи и определению функциональных требований к web-приложению.

Второй раздел содержит выбор средств разработки, проектирование структуры базы данных и интерфейса web-приложения.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

В настоящее время учебные заведения для улучшения качества образования стали активно внедрять вычислительную технику и информационные технологии. Так как те или иные внедрения положительно сказываются на рабочем процессе, делая его наиболее эффективным. К этому относятся различные системы тестирования знаний и документооборота.

Стоит отметить, что в учебных учреждениях системы документооборота реализованы только на более высоком уровне, а именно для наиболее эффективного обмена различными документами: приказы, указания, докладные записки, служебные письма, акты, объяснительные записки, протоколы, должностные инструкции, характеристики, контракты, преподавательские отчеты. Все перечисленные документы относятся к сотрудникам учебного заведения и, следовательно, улучшается эффективность обмена только между ними. То есть в данный момент никак не автоматизируется процесс сдачи документов (курсовых и лабораторных работ) между преподавателем и обучающимися.

Актуальность данной работы заключается в том, чтобы используя принципы характерные для систем электронного документооборота автоматизировать процесс движения документов (курсовых и лабораторных работ) между преподавателем и обучающимися.

Целью данной выпускной квалификационной работы является разработка web-приложения для учета сдачи лабораторных и курсовых работ обучающихся, в основе которого лежит реализация последовательного движения документа, используемое в workflow-системах.

В соответствии с данной целью были выявлены следующие задачи:

1) провести анализ предметной области;

2) определить функциональные требования к информационной системе;

3) провести обзор и анализ существующих информационных систем, предназначенных для организации документооборота;

4) изучить соответствующую техническую литературу по языкам программирования: C#, JavaScript. Также изучить литературу по использованию фреймворка ASP.NET Core;

5) разработать собственную нотацию для описания последовательного движения документа;

6) осуществить проектирование и разработку web-приложения.

1 СИСТЕМА ЭЛЕКТРОННОГО ДОКУМЕНТООБОРОТА

1.1 Основные понятия и задачи СЭД

Электронный документооборот (далее ЭД) – это система реализующая автоматическое движение электронных документов внутри разработанного приложения на основе схемы бизнес-процесса. Благодаря этой системе, пропадает необходимость бумажного делопроизводства [1].

Системы электронного документооборота (далее СЭД) позволяют значительно снизить временные трудозатраты на обработку типовых документов, по этой причине такие системы активно развиваются и по сегодняшний день.

Главным элементом ЭД является документ. Стоит отметить, что в качестве документа выступают не только привычные текстовые файлы, но и любая информация, вводимая пользователем вручную в соответствующие поля для ввода.

Главными задачами СЭД является: организация хранения и обработка документов. Хранение документов должно выполнятся как во время выполнения бизнес-процесса, так и после его окончания (хранение документа в архиве). Также в СЭД должно быть реализовано отслеживание состояния бизнес-процесса. Под состоянием подразумевается следующее:

- срок исполнения документа;
- движение документа, на каком этапе находится исполнение процесса;
- контроль за версиями всех документов.

Полнофункциональная СЭД – это СЭД, которая охватывает весь цикл документооборота на предприятии с момента создания документа и до его отправления в архив системы. Для данной СЭД характерно следующее:

- централизованное хранение документов в системе;

• объединяет разрозненные потоки документов с разных предприятий в единую систему;

• возможность гибко управлять документами. Это достигается за счет возможности задавать маршрут двумя способами: с помощью жесткого определения движения документа или свободной маршрутизации;

• жесткое разграничение доступа пользователей к тем или иным документам в зависимости от выданных администратором им полномочий;

• интеграция с уже существующими в организации системами.

Основными пользователями СЭД являются крупные организации, чья деятельность связана с хранением и обработкой большого объема типовых документов.

Существует несколько типов СЭД:

1) СЭД, которые акцентируют свое внимание на автоматизации бизнес-процессов. Как правило, такие системы управляют слабоструктурированными данными (такое управление лежит в основе всех ECM-систем) и их часто используют для решения каких-либо узконаправленных задач в промышленной отрасли. Самая известная компания, разрабатывающая СЭД такого типа является Documentum;

2) следующим типом СЭД является корпоративная. Данная СЭД предназначена для организации одновременной работы над проектом множества пользователей в реальном времени, поэтому такие системы в своем составе имеют такой функционал, как: общий чат в режиме онлайн и возможность проведения видеоконференции. Данные СЭД, в отличие от предыдущего типа, предназначены для решения более общих задач, свойственных любым организациям, вне зависимости от отрасли;

3) workflow-системы, предназначенные для обеспечения движения электронного документа согласно маршруту, указанному в бизнес-процессе. Данная система является неотъемлемой частью любой СЭД, однако ее принято выделять, как отдельный тип систем. Workflow-систему включают в состав СЭД для повышения эффективности выполнения бизнес-процесса.

1.1.1 Основные свойства СЭД

1.1.1.1 Открытость

Это значит, что все СЭД в основе имеют архитектуру, построенную на модулях, и их API-интерфейсы являются открытыми [2]. Это значительно упрощает добавление нового функционала к имеющейся СЭД. По этой причине возникли компании, которые разрабатывают различные модули, интегрируемые с СЭД. Например, разработаны такие модули, как: ввод документов в систему со сканера, связь СЭД с электронной почтой и др.

1.1.1.2 Высокая степень интеграции с прикладным ПО

Также СЭД характеризуется высокой степенью интеграции с прикладными программами. Это достигается за счет использования таких технологий, как: DDF, OLE Automation, ActiveX и др. Таким образом, отпадает вовсе необходимость пользоваться утилитами СЭД [2], так как в момент установки клиентской части СЭД в прикладные программы добавляются новые функции с соответствующим меню. Например, пользователь, открывая Microsoft Word, видит все библиотеки и документы СЭД. А после окончания работы с документом, при нажатии на кнопку сохранения, он автоматически добавится в базу данных (далее БД) СЭД.

1.1.1.3 Особенности хранения документов

Чаще всего, в СЭД для хранений документов реализуется иерархическая система. При этом количество уровней вложенности является неограни-ченным.

Также стоит отметить, что один и тот же документ может входить в несколько папок одновременно. Это возможно за счет применения ссылок. Сам же документ всегда хранится в одном месте, определенном администратором.

Каждый документ в СЭД имеет определенный набор атрибутов, таких как: название, дата создания и т. д. При этом набор атрибутов у одного типа документа может отличаться от набора у другого. Все атрибуты документа хранятся в БД [2].

Серверная часть СЭД состоит из следующих частей:

- хранилища атрибутов документов [2];
- хранилища документов [2];
- сервисов полнотекстовой индексации [2].

Хранилище документов и атрибутов чаще всего объединены и именуются архивом документов. Для хранения атрибутов в большинстве случаев используют такие системы управления базами данных (далее СУБД), как: Oracle и MS SQL Server. Данные СУБД позволяют выполнять поиск документов по их атрибутам.

Для хранения самих документов используют следующие файл-серверы: MS Windows NT, UNIX и др. В таком случае можно организовать взаимодействие серверов с разными операционными системами (далее ОС). Например, приложение будет работать без проблем, сли БД с атрибутами документов под управлением ОС UNIX в сети TCP/IP, а сами документы хранятся под ОС Novell NetWare в сети IPX/SPX.

Еще одним вариантом реализации является хранение самих документов с их атрибутами в БД. Преимущество в том, что значительно повышается безопасность доступа к документам [3], а недостаток – это низкая эффективность работы с документами, если хранится большой объем информации.

1.1.1.4 Особенности маршрутизации документов

Задать маршрут документа можно с помощью либо свободной, либо жесткой маршрутизации [4].

При свободной маршрутизации пользователь, участвующий в бизнес-процессе при необходимости может изменить маршрут документа [4].

При жесткой маршрутизации маршрут задан строго, и никто из пользователей в бизнес-процессе не может его изменить [4]. В большинстве СЭД данный модуль, реализующий маршрутизацию документов, входит в ее состав по умолчанию.

1.1.1.5 Разграничение доступа

Разграничение полномочий пользователей и контроль за доступом к документам являются неотъемлемой частью любой СЭД [4].

СЭД содержат следующие виды доступа:

- полный контроль над документом;
- право редактировать, но не уничтожать документ;
- право создавать новые версии документа, но не редактировать его;
- право аннотировать документ, но не редактировать его и не создавать новые версии;
- право читать документ, но не редактировать его;
- полное отсутствие прав доступа к документу.

1.1.1.6 Отслеживание версий и подверсий документов

Отслеживание всех версий и подверсий документов необходимо для реализации параллельного выполнения задания. Например, если исполнитель создал документ и передал его на рассмотрение на следующий этап. Второй исполнитель решил изменить документ, создал на его основе документа новую версию и передал свою версию документа на следующий этап третьему пользователю. После, спустя некоторое время, ознакомившись с замечаниями и исправлениями, решает создать на основе исходной версии документа подверсию и дорабатывает его [4]. Возможность отслеживать все версии и подверсии документа является существенным достоинством СЭД.

1.1.1.7 Наличие утилит просмотра документов разных форматов

Большинство СЭД имеют встроенные модули для просмотра документов, понимающие многие десятки форматов файлов, в том числе, графические файлы [4]. Однако, если стандартного набора форматов недостаточно, то можно приобрести соответствующие интегрируемые с СЭД модули у третьих фирм.

1.1.1.8 Аннотирование документов

Важной функцией в СЭД является аннотирование [4]. Оно используется в случае, если пользователи не имеют прав на внесение изменений в документ, когда он находится на согласовании. Необходимость данной функции раскрывается в полной мере при организации групповой работы над документами.

1.2 Технологии электронного документооборота

В основе ЭД лежат технологии две технологии:

- docflow отвечает за движение документа по заданному маршруту, определяя условия его перемещения;
- workflow задает порядок работ по обработке документа и управляет правами доступа к нему.

Помимо вышеперечисленных, в СЭД применяются и другие технологии, которые связанны с обработкой и преобразованием документов.

1.2.1 Технологии хранения электронных документов

Любому документу в СЭД соответствует какой-либо набор атрибутов, который определяет его свойства, использующиеся для таких задач, как: поиск, построение отчётов, группировка и т. д. [5] Сам файл документа либо загружается пользователем, либо создается автоматически по заданному шаблону, в котором переменные заменяются значениями из атрибутов документа и из БД. Атрибуты документа хранятся в БД. Для реализации полнотекстового поиска необходимо поместить содержимое файла в

специальный индексный каталог СУБД. Сам файл документа сохраняется в папке, которая является хранилищем СЭД. Доступ к файлу документа имеют только те пользователи, которым выдали соответствующие полномочия через систему управления правами доступа СЭД. Для остальных по умолчанию доступ к файлу закрывается.

1.2.2 Технология поточного сканирования

Для реализации данной технологии необходимо специальное оборудование, а именно поточный сканер [5]. Суть данной технологии в сканировании множества документов с последующим добавлением их образов в БД. При этом возможна реализация нанесения штрих-кода на сканируемый документ.

1.2.3 Технология оптического распознавания текстов документов

Оптическое распознавание документа – это процесс преобразования документа из графического в текстовый формат [5]. Для реализации оптического распознавания документа используются либо самостоятельное приложение, либо уже входящее в состав СЭД. Например, СЭД ESCOM.BPM имеет встроенный модуль для оптического распознавания.

1.2.4 Технология штрих-кодирования документов

Штрих-кодирование используется для нанесения на графический документ уникального идентификатора. Данный штрих-код является абсолютно уникальным. Штрих-кодирование применяется в следующих случаях:

1) при регистрации входящих документов;

2) для осуществления поиска документа в БД;

3) для учёта места расположения оригинала документа.

1.2.5 Технология подписания документов ЭЦП

Электронно-цифровая подпись (далее ЭЦП) используется для определения неизменности документа в СЭД с момента его подписания. Само подписание осуществляется с помощью криптопровайдера и сертификата. Сертификат ЭЦП – это отдельный файл в специальном формате, который создаётся одним из удостоверяющих центров, обладающий соответствующей лицензией [3].

Для защиты сертификата ЭЦП от его хищения используют смарт-карты или ключи типа I-Token. Рекомендуется использовать именно эти средства по причине наличия у них защищенного хранилища и использования PIN-кода при обращении к сертификату. Если PIN-код будет неправильно введен несколько раз подряд, то сертификат заблокируется.

1.3 Workflow

Workflow – это бизнес-процесс, который содержит в себе описание движения документов в СЭД [6]. Примером такого бизнес-процесса может быть процесс сдачи лабораторной работы и последующее его согласование. Вначале документ утверждает проверяющий, а после, в случае правильного выполнения задания, преподаватель выставляет оценку.

Движок workflow – это модуль в системе, который производит выполнение бизнес-процесса [7]. Сам бизнес-процесс создается в какой-либо нотации либо в специальном приложении, либо вручную. Далее данная нотация загружается в движок workflow. Для работы с бизнес-процессом движок предоставляет специально разработанный интерфейс, который позволяет запускать, останавливать выполнение того или иного процесса, дает возможность просматривать его состояние и отслеживать ход выполнения.

Стоит отметить, что Workflow-системы вводятся в организации не для осуществления маршрутизации документов или автоматизации некоторых действий, а для описания бизнес-процесса. Маршрутизация же в свою очередь призвана повысить эффективность выполнения бизнес-процесса.

Каждый процесс Workflow-системы содержит в себе множество операций [7]. В свою очередь каждая операция имеет задание, которое предполагает ввод или обработку документов.

Операции содержат следующие параметры:

- адресат [8] – это пользователь или группа пользователей, которым выдается задание;
- экранная форма [8]. Данная форма содержит поля для ввода данных, которые после будут переданы другому пользователю согласно маршруту движения документа;
- предельный срок выполнения задания [8]. С помощью данного параметра определяют до какого времени операция должна быть выполнена;
- действия системы при инициализации и завершении операции [8].

При выполнении процесса Workflow вся введенная информация передается от одного пользователя к другому в виде упорядоченного множества данных. Каждая операция может использовать некоторую часть данного множества и отображать их в виде экранной формы. Любая workflow-система обязана предоставлять возможность динамически изменять состав и содержание экранной формы.

Все данные в экранной форме представлены в виде полей. Выделяют следующий виды этих полей:

- демонстрационные поля – эти поля только отображают какие-либо значения. Редактирование значений в этих полях не допускается;
- обязательные поля – это поля, которые необходимо заполнить перед тем, как завершить выполнение задания;
- необязательные поля – это поля, которые нет необходимости заполнять чтобы выполнить то или иное задание;
- вычисляемые поля – это поля, значения которых вычисляются по каким-либо заранее заданным правилам;
- невидимые поля – это поля, которые являются вычисляемыми, но при этом никак не отображаются на экранной форме.

Существует три метода для реализации workflow в СЭД:

- создание собственной нотации для описания бизнес-процесса. Недостаток такого метода в том, что ее нельзя будет применить в других СЭД;
- использование для описания бизнес-процесса таких языков, как: BPEL и XPDL. Данные языки считаются стандартом потому, что поддерживаются компаниями гигантами типа Microsoft, Oracle, собственно они же и занимаются их продвижениям, предоставляя соответствующие программы для работы с ними. Существенным недостатком данного подхода является сложность понимания данной нотации бизнес-процесса пользователями, не имеющими никаких знаний в программировании. Это означает, что организациям, чтобы создать новый бизнес-процесс, придется нанимать кодировщиков с соответствующей квалификацией, что повлечет за собой дополнительные расходы;
- реализация бизнес-процесса на уровне кода без использования каких-либо шаблонов маршрутов и движков для их загрузки. Существенными недостатками данной реализации являются: отсутствие гибкости системы и плохое восприятие бизнес-процесса как кодировщиками, так и заказчиками.

1.4 Примеры систем электронного документооборота

1.4.1 CHALEX

Система CHALEX представляет собой движок workflow, исполняющий бизнес-процессы, описание которых задано в формате XPDL [5].

К основным возможностям системы относятся:

- динамическая загрузка и выгрузка описаний бизнес-процессов в формате XPDL;
- динамическое определение и выбор шаблонов ресурсов, позволяющих назначать людей для выполнения задач бизнес-процесса;

- обеспечение полной интеграции с базой данных (БД) DAM фирмы CHALEX, что позволяет присоединять файлы к workflow, сохраняя их в БД;
- поддержка проектной организации workflow;
- поддержка организации присоединенных файлов в виде категорий, которые охватывают сразу несколько бизнес-процессов и проектов;
- поддержка XForm – следующего поколения HTML-форм, предоставление API для сохранения содержимого форм в базе данных;
- поддержка технологии JavaServer Faces (JSF) и JSF Forms, предоставление API для сохранения содержимого форм в базе данных;
- предоставление API для управления системой;
- предоставление web-ориентированного клиента для управления приложением.

Основными достоинствами данной СЭД является: возможность создания экранных форм с помощью встроенных средств системы и использование стандартного языка XPDL для описания бизнес-процесса.

1.4.2 DIRECTUM

Система DIRECTUM [5] – данная система охватывает весь цикл документооборота на предприятии с момента создания документа и до его отправления в архив системы. В основе архитектуры данной СЭД лежит модульный принцип.

Система предоставляет специальный инструмент IsBuilder, с помощью которого можно динамически изменять состав экранной формы для какого-либо этапа [5].

Для управления созданным бизнес-процессом система предоставляет специальный отдельный модуля с названием «Управление деловыми процессами».

Под маршрутом в данной системе подразумевается некоторая последовательность выполнения поставленных задач. В свою очередь каждая задача тоже имеет свой маршрут разделенный на этапы. На каждом этапе для

его исполнителя появляется уведомление с заданием. Маршруты могут быть заданы двумя способами: вручную или с помощью типового маршрута.

Типовым маршрутом является наиболее часто выполняемая задача или их набор. Удобство данной возможности заключается в автоматическом назначении исполнителей для всего маршрута, заполнении тех или иных полей, например, название и описание задачи. Настройка этих маршрутов осуществляется с помощью встроенного редактора схем.

Редактор схем располагает следующими элементами для создания маршрута.

1. Задание. Производит отправку уведомления с заданием назначенному исполнителю.

2. Сценарий. Это запрограммированный набор действий на встроенном в систему языке ISBL.

3. Условие. Данный элемент позволяет осуществлять ветвление процесса.

4. Ожидание. Приостанавливает выполнение процесса до тех пор, пока не истечет крайний срок.

5. Мониторинг.

Основными достоинствами данной СЭД является наличие встроенных механизмов для создания экранных форм и редактора схем, который предоставляет обильное количества функционала для создания маршрута любой сложности. При этом данный редактор схем имеет интуитивно понятный интерфейс, поэтому для создания маршрута пользователю нет необходимости иметь какие-либо знания о программировании.

1.4.3 Naumen DMS

Система Naumen DMS [5] – это корпоративная СЭД. В своей основе имеет сервисно-ориентированную архитектуру.

Все документы в системе хранятся централизованно, а все бизнес-процессы реализованы на языке BPEL. Для выполнения бизнес-процесса в системе предусмотрен так называемый «Диспетчер задач».

Основными достоинствами данной СЭД является: сервисно-ориентированная архитектура и использование языка BPEL, который позволяет создавать маршруты любой сложности.

Благодаря сервисно-ориентированной архитектуре становится возможным расширение функционала в СЭД путем интеграции других приложений. Причем все эти приложения могут быть созданы на любой платформе.

Недостатком данной СЭД является сложность создания бизнес-процесса на языке BPEL.

1.4.4 ЕВФРАТ-документооборот

Система ЕВФРАТ-документооборот [5] содержит редактор для экранных форм и специальный инструмент «Дизайнер маршрутов», использующийся для создания и редактирования маршрутов.

Данная СЭД имеет три типа маршрутов:

- последовательная работа с документом;
- параллельная работа с документом;
- параллельно-последовательная работа с документом.

При последовательной работе с документом переход от одной операции к единственной, следующей за ней, выполняется только после того, как первая операция завершена.

В случае параллельной работы осуществляется одновременный переход от какой-либо завершенной операции ко всем, следующим за ней операциям, которые начинают выполняться одновременно.

Параллельно-последовательная работа — это смешанный вид маршрута, получающийся в результате комбинирования первых двух типов.

Операции в системе делятся на поручения и согласования. В случае поручения операция завершается, только если контроллер отслеживающий ее

исполнения отметит ее как «выполненная». А при согласовании операция завершается, только после того, как исполнитель либо утвердит, что задание выполнено, либо отклонит, отправив на доработку [5].

Основными достоинством данной СЭД является то, что для создания маршрута документа не нужно иметь никаких навыков программирования. Существенным же недостатком является то, что в нотации отсутствуют элементы для описания циклов.

Результаты анализа реализации workflow в различных СЭД представлены в таблице 1.

Таблица 1 – Сравнение систем электронного документооборота

Название	Наличие встроенного конструктора бизнес-процессов	Поддержка стандартных языков исполнения бизнес-процессов	Достоинства	Недостатки
NAUMEN DMS	Нет	BPEL	Поддерживает взаимодействие со сторонними системами	Сложность разработки бизнес-процессов
CHALEX workflow	Нет	XPDL	Обеспечивает большую гибкость при создании действий бизнес-процессов и пользовательского интерфейса	Невозможно использовать систему в качестве движка исполне-ния XPDL
DIRECTUM	Да	Нет	Имеет весь необходимый инструментарий для создания сложных маршрутов документов	Использует собственную нотацию для разработки бизнес-процесса
Ефрат-документооборот	Да	Нет	Простота создания маршрутов документов	Нельзя создать в бизнес-процессе цикл

1.5 Возможность применения СЭД в образовательной деятельности

Совершенствование эффективности рабочего процесса является естественной потребностью любой организации. В настоящее время процесс работы в высшем учебном заведении не оптимизирован полностью, так как уже внедренная информационная система улучшает только внутреннее взаимодействие персонала. Следовательно, процесс сдачи и приема лабораторных и курсовых работ происходит в стандартном бумажном виде. Данный архаичный подход в свою очередь порождает множество проблем:

1) в первую очередь, самая очевидная – это потеря или уничтожение документов после прохождения определенного срока. То есть после приема они отправляются в архив и хранятся там несколько лет, после чего уничтожаются. Так как все высшие учебные заведения обязаны проходить аккредитацию, то возможны те или иные проверки со стороны государственной организации «Рособрнадзор», которая может потребовать предоставить утерянные или уничтоженные документы;

2) отсутствие наглядного представления, какие курсовые или лабораторные работы сданы были вовремя, а какие – нет.

Следовательно, разработка web-приложения, которое автоматизирует процесс приема или отказа документов (курсовых и лабораторных работ) между преподавателем и обучающимися, является разумным решением.

Для реализации данного web-приложения можно взять workflow-систему, которая является неотъемлемой частью любой СЭД.

Способ организации движения документа посредством использования заранее подготовленного шаблона маршрута с помощью определенной нотации, дает обширные возможности для создания заданий любой структуры.

Например, любой бизнес-процесс, который автоматизирует СЭД, имеет так называемые этапы, на которых происходит какая-либо обработка и ввод данных. И после выполнения задания для данного этапа, все загруженные

документы передаются дальше, в соответствии с маршрутом, следующему пользователю.

Данное разделение бизнес-процесса на этапы отлично подходит для сдачи курсовых и лабораторных работ. По причине того, что процесс сдачи можно представить в виде некоторой упорядоченной последовательности действий. Например, курсовая работа, как правило, состоит из двух частей: теоретическая и практическая. Тогда процесс сдачи данной работы можно представить следующим образом:

1) сдача теоретической части курсовой;

2) проверка теоретической части;

3) сдача практической части курсовой;

4) проверка практической части;

5) выставление оценки.

Как видно, маршрут для сдачи курсовой работы содержит в себе пять этапов потому, что после сдачи какой-либо части задания необходимо провести его проверку и в случае не удовлетворительного выполнения отправить на предыдущий этап на доработку. Последним, пятым, этапом является выставление оценки. Исполнителю на данном этапе передаются все части задания и комментарии пользователей, которые осуществляли проверку и, на основании всей полученной информации, он производит оценивание выполнения всего задания.

Лабораторная работа, в отличие от курсовой, имеет всего три этапа, так как подразумевается выполнение небольшого по объему задания. Вследствие чего, нет необходимости разделять его на части. Процесс сдачи для лабораторной работы в workflow-системе можно представить следующим образом:

1) сдача лабораторной работы;

2) проверка лабораторной работы;

3) выставление оценки.

Как видно, концепция движения документа в СЭД отлично подходит для образовательных целей, так как помимо разбиения одной полноценной работы на части, она также позволяет для каждой из них осуществить преждевременную проверку и, в случае не удовлетворительного выполнения, вернуть документ на предыдущий этап.

Также стандартный функционал workflow-систем предполагает:

- возможность задать крайний срок исполнения и, в случае его истечения, извещает пользователя (в данном случае преподавателя), который выдал задание;
- возможность просматривать состояние выполнения задания, а именно на каком этапе он находится на текущий момент времени.

Две данные функции являются крайне ценными в образовательной деятельности.

Стоит отметить, что workflow-система также не накладывает никаких ограничений на назначение исполнителя этапа. Таким образом, преподаватель может без проблем организовать выполнение какой-либо совместной работы для нескольких обучающихся.

1.6 Постановка задачи

Необходимо реализовать web-приложение для сдачи курсовых и лабораторных работ, используя workflow-систему, присущую СЭД, а именно реализация последовательного движения документов.

Для реализации данной workflow-системы наиболее оптимальной является разработка собственной нотации шаблона маршрута.

Сформулируем функциональные требования к разрабатываемой системе:

1) при работе с системой пользователь, имеющий доступ к функции «Выдать задание», должен иметь возможность осуществлять следующие функции:

- создание, редактирование, удаление задания;

- возможность просматривать состояние уже выданного задания. Состояние включает в себя следующие параметры: название, описание, дата создания задания, название этапа на котором находится выполнение задания, название шаблона маршрута, оценка, ссылки на все документы, которые были загружены во время выполнения задания;
- выставление оценки;

2) при работе с системой администратор должен иметь возможность:

- создавать, редактировать, удалять группы для пользователей;
- выдавать группе полномочие на использование функции «Выдать задание»;
- создавать, редактировать, удалять пользователей, а также привязывать их к определенной группе;
- загружать вручную сформированный шаблон маршрута в виде формата XML;

3) при работе с системой пользователь, который состоит в группе может просматривать все задания, для которых он был назначен исполнителем для конкретного этапа. Данное задание отображается только в случае, если этап, для которого он был назначен является активным.

1.7 Выводы по разделу

В данном разделе были определены термины и понятия, и проведено сравнение систем электронного документооборота.

В сети Интернет была найдена только одна распространяющаяся на бесплатной основе СЭД, называющаяся OpenDocman, но в ней отсутствовал функционал для организации движения документа. Все остальные системы документооборота, имеющие workflow, являются платными, а также имеют избыточный функционал в виде возможности организации параллельного выполнения задания. Для решения же поставленной задачи, а именно организация сдачи курсовых и лабораторных работ достаточно после-

довательного линейного движения документа. В связи с этим, было решено разработать собственное web-приложение, реализующее последовательное движение документа.

В качестве метода реализации workflow была выбрана разработка собственной нотации шаблона маршрута.

На основании всей полученной информации были сформулированы функциональные требования к разрабатываемой системе.

2 СРЕДСТВА РАЗРАБОТКИ WEB-ПРИЛОЖЕНИЯ

2.1 Языки программирования

Для серверной части будет использоваться C#. C# – объектно-ориентированный язык программирования. Данный язык имеет статическую типизацию, поддерживает полиморфизм, перегрузку операторов, делегаты, атрибуты, события, свойства, обобщённые типы и методы, итераторы, анонимные функции с поддержкой замыканий, LINQ, исключения, комментарии в формате XML [9].

Для клиентской части будут использоваться следующие языки:

1) HTML – стандартный язык разметки web-документов [9];

2) CSS – формальный язык описания внешнего вида документа, написанного с использованием языка разметки [9];

3) JavaScript – объектно-ориентированный скриптовый язык программирования. JavaScript поддерживается всеми существующими браузерами и является стандартом для современных интерактивных web-приложений [10].

2.2 Visual Studio

Для разработки web-приложения был выбран Visual Studio Community 2019. По причине того, что данная версия интегрированной среды разработки распространяется под лицензией «GNU General Public License» (распространяется на бесплатной основе) и обладает всеми необходимыми инструментами для реализации поставленной задачи:

- предоставляет возможность создавать web-приложение с помощью ASP.NET или ASP.NET Core. ASP.NET – платформа для разработки web-приложений, в состав которой входит: web-сервисы, программная инфра-

структура, модель программирования и т. д. ASP.NET Core является продолжением развития платформы ASP.NET [11];

- данная среда поддерживает множество языков при разработке web-приложений в том числе язык программирования C#;
- Visual Studio автоматически устанавливает ряд деталей при добавлении web-элемента управления, присоединении обработчиков событий;
- форматирование кода происходит по мере его ввода. Автоматически вставляются необходимые отступы и применяется цветовое кодирование для тех или иных элементов;
- поддерживает множество удобных функций, которые позволяют писать код быстрее. К таким функциям относится: IntelliSense, который перехватывает ошибки и предлагает правильные варианты, функции поиска и замены, которые позволяют отыскивать ключевые слова как в одном файле, так и во всем проекте и функции автоматического добавления и удаления комментариев, которая может временно скрывать блоки кода [11];
- имеет удобные инструменты отладки для отслеживания ошибок, а именно: выполнять код по строке за раз, устанавливать точки прерывания и сохранять их, а также в любое время просматривать текущую информацию из памяти [11];
- Visual Studio также позволяет создавать и подключать сторонние дополнения для расширения функциональности.

2.3 Базы данных

Существует множество СУБД.

СУБД – это комплекс языковых и программных средств, предназначенный для создания, ведения и совместного использования БД многими пользователями [12]. Обычно СУБД различают по используемой модели данных. Так, СУБД, основанные на использовании реляционной модели данных, называют реляционными СУБД.

Рассмотрим следующие СУБД:

− Microsoft SQL Server express;

− MySQL;

− SQlite.

Microsoft SQL Server express – система управления реляционными базами данных, разработанная корпорацией Microsoft. Основной используемый язык запросов – Transact-SQL, создан совместно Microsoft и Sybase. Transact–SQL является реализацией стандарта ANSI/ISO по структурированному языку запросов (SQL) с расширениями. Используется для работы с БД размером от персональных до крупных БД масштаба предприятия. Бесплатно предоставляется только хранилище объемом 10 ГБ [12].

MySQL – свободная реляционная СУБД. Разработку и поддержку MySQL осуществляет корпорация Oracle, получившая права на торговую марку вместе с поглощённой Sun Microsystems, которая ранее приобрела шведскую компанию MySQL AB [12].

Гибкость СУБД MySQL обеспечивается поддержкой большого количества типов таблиц: пользователи могут выбрать как таблицы типа MyISAM, поддерживающие полнотекстовый поиск, так и таблицы InnoDB, поддерживающие транзакции на уровне отдельных записей. Более того, СУБД MySQL поставляется со специальным типом таблиц EXAMPLE, демон-стрирующим принципы создания новых типов таблиц. Благодаря открытой архитектуре и GPL-лицензированию, в СУБД MySQL постоянно появляются новые типы таблиц [12].

SQlite – легко встраиваемая в приложения база данных. Так как эта система базируется на файлах, то она предоставляет довольно широкий набор инструментов для работы с ней, по сравнению с сетевыми СУБД [13]. При работе с этой СУБД обращения происходят напрямую к файлам, вместо портов и сокетов в сетевых СУБД. Именно поэтому SQLite очень быстрая, а также мощная благодаря технологиям обслуживающих библиотек. Распространяется данная СУБД по лицензии GPL.

На основании полученных данных была выбрана СУБД SQlite, так как она имеет весь необходимый функционал для реализации web-приложения, распространяется по лицензии GPL, а также не имеет каких-либо ограничений в размере БД.

2.4 Фреймворки

Интегрированная среда разработки Visual Studio предоставляет два на выбор фреймворка: ASP.NET Core и ASP.NET на платформе .NET Framework.

ASP.NET – web-фреймворк, который работает только на оперативной системе Windows. Для создания web-приложений предоставляет три следующие технологии [11]:

- ASP.NET WebForms дает возможность создавать динамические web-приложения путем перетаскивания на форму тех или иных компонентов, каждый из которых управляется посредством событий;
- ASP.NET MVC предлагает эффективный, основанный на шаблонах способ создания динамических web-приложений, который позволяет четко разделять проблемы и дает полный контроль над разметкой для гибких разработок. ASP.NET MVC содержит множество функций, позволяющих вести быструю TDD-совместимую разработку для создания сложных приложений, использующих новейшие web-стандарты;
- ASP.NET WebPages – это платформа, которая позволяет создавать динамические web-страницы. Обычные HTML-страницы являются статическими и их содержимое определяется разметкой на web-странице. Динамические же страницы позволяют создавать содержимое страницы на ходу, а также выполнять другие различные действия: можно запросить у пользователя входные данные с помощью формы и затем изменить содержимое самой страницы, можно получить информацию от пользователя, сохранить в базу данных и затем перечислить ее позже и т. д.;

ASP.NET Core – это web-фреймворк, который является кроссплатформенным, то есть разработанный проект будет работать на таких оперативных системах как: Windows, Linux и macOS. А для развертывания web-приложения можно использовать традиционный IIS, либо кроссплатформенный web-сервер Kestrel. ASP.NET Core является не просто продолжением развития платформы ASP.NET, а революцией всей платформы в целом. Также стоит отметить, что ASP.NET Core полностью является opensource-фреймворком.

Хоть ASP.NET Core и нацелен преимущественно на использование .NET Core, но фреймворк также может работать и с полной версией фреймворка .NET.

Благодаря модульности фреймворка все необходимые компоненты web-приложения могут загружаться как отдельные модули через пакетный менеджер Nuget.

Для создания web-приложений ASP.NET Core предоставляет следующие технологии: ASP.NET Core MVC и ASP.NET Core Razor Pages.

ASP.NET Core MVC в отличие от предыдущей версии объединяет функциональность MVC, Web API и Web Pages. Раньше данные технологии реализовались отдельно и поэтому содержали много дублирующей функциональности, по этой причине они были объединены в одну программную модель ASP.NET Core MVC, а технология Web Forms была удалена из-за своей архаичности.

Кроме объединения вышеупомянутых технологий в одну модель, в MVC был добавлен ряд дополнительных функций. Одной из таких функций являются tag helper, которые позволяют более органично соединять синтаксис html с кодом C# [14].

Также было упрощено управление зависимостями и конфигурирование проекта. Фреймворк теперь имеет свой легковесный контейнер для внедрения зависимостей, и больше нет необходимости применять сторонние контейнеры, такие как Autofac, Ninject [14].

В качестве инструментария разработки можно использовать последние выпуски Visual Studio, начиная с версии Visual Studio 2015. Кроме того, можно создавать приложения в среде Visual Studio Code, которая является кроссплатформенной и может работать как на Windows, так и на Mac OS X и Linux [14].

Для обработки запросов теперь используется новый конвейер HTTP, который основан на компонентах Katana и спецификации OWIN. А его модульность позволяет легко добавить свои собственные компоненты [14].

В результате главными отличиями ASP.NET Core от предыдущих версий ASP.NET являются:

- новый легковесный и модульный конвейер HTTP-запросов;
- возможность развертывать приложение как на IIS, так и в рамках своего собственного процесса;
- использование платформы .NET Core и ее функциональности;
- распространение пакетов платформы через NuGet;
- интегрированная поддержка для создания и использования пакетов NuGet;
- единый стек web-разработки, сочетающий Web UI и Web API;
- конфигурация для упрощенного использования в облаке;
- встроенная поддержка для внедрения зависимостей;
- расширяемость;
- кроссплатформенность: возможность разработки и развертывания приложений ASP.NET на Windows, Mac и Linux;
- развитие ASP.NET Core как open source.

На основании полученных данных в качестве web-фреймворка был выбран ASP.NET Core по причине кроссплатформенности и возможности использовать удобный встроенный загрузчик модулей Nuget.

2.5 Модели построения web-приложения

Фреймворк ASP.NET Core предоставляет на выбор две отличающиеся друг от друга модели построения web-приложения: ASP.NET Core MVC и ASP.NET Core Razor Pages.

ASP.NET Core MVC – является многофункциональной платформой для создания web-приложений и API-интерфейсов с помощью структуры проектирования Model-View-Controller.

Сам паттерн MVC не является какой-то новой идеей в архитектуре приложений, он появился еще в конце 1970-х годов в компании Xerox как способ организации компонентов в графическом приложении на языке Smalltalk [14].

Концепция паттерна MVC предполагает разделение приложения на три компонента:

- модель;
- представления;
- контроллер.

Отношения между компонентами паттерна можно описать следующей схемой, представленной на рисунке 2.1.

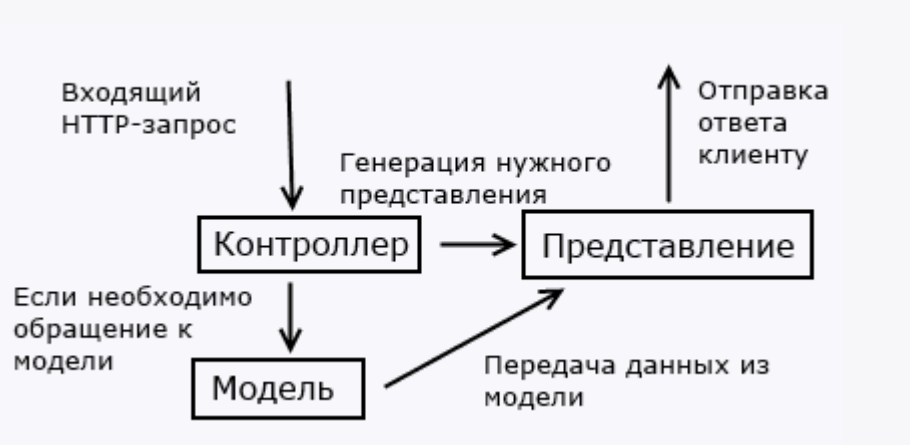

Рисунок 2.1 – Схема MVC

В данной схеме модель является независимым компонентом – любые изменения контроллера или представления никак не влияют на модель. Контроллер и представление являются относительно независимыми компонентами. Так, из представления можно обращаться к определенному

контроллеру, а из контроллера генерировать представления, но при этом нередко их можно изменять независимо друг от друга. Такое разграничение компонентов приложения позволяет реализовать концепцию разделение ответственности, при которой каждый компонент отвечает за свою строго очерченную сферу. В связи с чем, легче построить работу над отдельными компонентами. И благодаря этому приложение легче разрабатывать, поддерживать и тестировать отдельные его компоненты.

ASP.NET Core Razor Pages – это новый аспект платформы ASP.NET Core MVC и, по сути, работает поверх данной платформы, однако имеет существенные отличия при разработке web-приложения, по этой причине данную модель считают альтернативой стандартной MVC и при создании нового проекта эти две модели разделяют друг от друга.

Главное отличие – это наличие папки Pages в которой лежат динамические страницы. Данные страницы имеют две части: файл с расширением .cshtml, который содержит смесь кода html и конструкций C# и .cshtml.cs, который выступает в роли модели и участвует в обработке запроса для первого файла. В следствии этого, теперь нет необходимости создавать контроллеры в отдельной папке, что упрощает понимание того, где находится обработчик для той или иной web-страницы.

При создании проекта по умолчанию в папке Pages имеются следующие файлы:

- _Layout.cshtml – мастер-страница, в которую вставляются страницы Razor;
- _ViewStart.cshtml – задает мастер-страницу;
- _ViewImports.cshtml – определяет директивы Razor, которые добавляются на каждую страницу Razor;
- _ValidationScriptsPartial.cshtml – частичное представление, которое подключает js-скрипты валидации на стороне клиента;

• About.cshtml, Contact.cshtml, Error.cshtml и Index.cshtml – это страницы Razor, которые определяют визуальную часть страницы и логику обработки запроса.

Таким образом, уже на стадии создания проекта генерируются автоматически все необходимые файлы для создания шаблонов страниц, что упрощает и значительно ускоряет разработку web-приложения.

Существенным плюсом ASP.NET Core Razor Pages является марш-рутизация. В отличие от ASP.NET Core MVC, нет необходимости в ручную прописывать все маршруты. Строка запроса URL сопоставляется с определенной страницей Razor на основании ее расположения в проекте в папке Pages.

На основании полученной информации для разработки web-приложения была выбрана модель ASP.NET Core Razor Pages по следующим причинам:

1) нет необходимости вручную прописывать маршруты, что исключает возможность допущения ошибок и ускоряет процесс разработки;

2) автоматическая генерация представления и ее модели, которые отвечают за отображение данных и обработку запроса, располагаются в одном месте, что упрощает понимание работы web-приложения другим разра-ботчиком, если потребуется расширение функционала.

2.6 Обзор современных web-серверов

Немаловажным является выбор web-сервера, который будет об-служивать web-приложение.

Nginx – высокопроизводительный HTTP-сервер, предназначенный в основном для раздачи клиентам статического контента (изображений, javascript-файлов, css-стилей и т. д.). В основе лежит технология неб-локирующих соединений, что при большом количестве одновременных подключений существенно экономит ресурсы сервера. Разрабатывался специально для испытывающих большую нагрузку серверов [11].

Apache HTTP-сервер – свободный web-сервер. По статистике на февраль 2019 года используется в 45.8% всех сайтов в Интернете, что делает его самым популярным web-сервером. Версия 2.4.x обладает проверенной годами и миллионами пользователей стабильностью и надежностью. Сервер имеет большое количество модулей для работы со многими серверными технологиями. Apache также является кроссплатформенным ПО, поддерживает операционные системы Linux, Mac OS, Microsoft Windows, Novell NetWare, BeOS [11].

Microsoft IIS – еще один надежный сервер от компании Microsoft. Он жестко укрепился на втором месте с 14 % использования в сети. После установки программы, будут поддерживаться только два языка программирования (VBScript и JScript). Однако, можно открыть дополнительные возможности, установив для этого нужные расширения. С установкой таких модулей, функциональность данного сервера значительно повышается [11].

На основании данной информации в качестве web-сервера был выбран Microsoft IIS, так как встроен в интегрированную среду разработки Visual Studio и из нее же можно производить настройку данного сервера.

2.7 Вывод по разделу

В данном разделе был проведен обзор и обоснование выбора средств разработки.

Для разработки серверной части был выбран язык C#, который поддерживает объектно-ориентированный подход и является типобезопасным, по причине строгой типизации и не возможности выполнения неконтролируемого привидения типов. По этой причине в качестве средства разработки был выбран Visual Studio Community 2019, предоставляющий технологию ASP.NET для разработки web-приложения и встроенный web-сервер IIS.

Данная интегрированная среда разработки предоставляет различные фреймворки для разработки web-приложений. В качестве фреймворка был

выбран ASP.NET Core с уникальной моделью построения web-приложения Razor Pages, который значительно упрощает разработку.

3 ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА WEB-ПРИЛОЖЕНИЯ

3.1 Шаблон маршрута

Для реализации движения документа был выбран метод создания собственной нотации с последующей его загрузкой в базу данных. Стоит отметить, что реализовываться будет только последовательное движение документа.

Данный шаблон маршрута будет находится в формате XML, так как такой формат файла наглядно показывает структуру движения документа.

В шаблоне могут быть описаны следующие типы документов, причем с точки зрения документооборота под «документами» подразумеваются не только привычные всем текстовые или какие-либо другие файлы, но и поля в которые пользователь вводит информацию:

1) текст;

2) word-файл.

Каждый такой документ имеет следующие атрибуты: «Название», «Тип» и «Обязательный». Рассмотрим подробно каждый из этих атрибутов:

- атрибут «Название» необходим для вывода на web-форме, иначе пользователь не будет понимать для чего предназначено то или иное поле ввода;
- атрибут «Тип» у документа необходим для определения вывода того или иного html-тега;
- атрибут «Обязательный» является опциональным и если его не указывать, то по умолчанию документ будет обязательным, и форма на web-странице будет выдавать ошибку. Если же указать данный атрибут со значением false, то запрос пройдет успешно. Данный атрибут можно указывать у документов с любым типом.

Для организации более гибкой системы для каждого этапа указываются входные и выходные документы, а сам этап имеет следующие атрибуты: название, срок сдачи и возврат. Рассмотрим более подробно эти атрибуты:

- атрибут «Название» необходим для вывода названия этапа на web-странице;
- атрибут «Срок» определяет крайний срок выполнения этапа и в случае превышения лимита времени у пользователя, который выдал задание, данный этап будет помечен красным цветом;
- атрибут «Возврат» является опциональным, если его указать для этапа, то становится доступна функция возврата документа на предыдущий этап, а на web-форме отображается соответствующая кнопка.

Весь XML шаблон имеет четкую структуру, которую необходимо соблюдать. Если этого не делать, то web-приложение при загрузке данного файла будет показывать ошибку.

Шаблон начинается с общего блока «Шаблон». Данный блок, как и все остальные имеет закрывающийся тег для того, чтобы можно было определить конец нотации.

После него идет блок под названием «Документы», внутри которого описываются все документы под тегом «Документ». Количество описываемых документов может быть сколько угодно, однако каждый из них должен иметь уникальный идентификатор.

Далее начинается блок под названием «Этапы» и каждый этап обозначается тегом «Этап». Данных этапов, как и в случае с документами, может быть неограниченное количество.

В свою очередь каждый этап может иметь два блока «Входные документы» и «Выходные документы». Каждый из этих блоков может отсутствовать, если администратор считает, что в них нет необходимости, однако порядок написания строго регламентирован. Вначале описываются входные, а после выходные документы. Данные блоки необходимы для определения того, в каких полях выводить тот или иной документ на

экранной форме. Например, если документ находится в «Входные документы» и имеет тип «Файл», то на экранной форме отобразиться тег <a href> внутри которого будет находится ссылка для скачивания файла. Если же указан документ с типом «Текст», то отобразиться тег <p> с данными введенными ранее пользователем. В случае, если документ находится в «Выходные документы» и имеет тип «Файл», то отобразиться тег <input type=’file’>, использующийся для загрузки файлов. А если тип документа будет «Текст», то отобразиться тег <input type=’text’>, который предназначен для ручного ввода пользователем какой-либо информации.

Связь этапа и документа осуществляется посредством написания тега «Документ» внутри любого из блоков «Входные документы» и «Выходные документы» с указанием его уникального идентификатора.

Для сдачи лабораторной работы необходимо три этапа. Первый этап– это выполнение и загрузка лабораторной работы. На втором осуществляется проверка загруженного файла, а на третьем непосредственное выставление оценки. В следствии чего шаблон маршрута в формате XML будет выглядеть следующим образом, представленным на рисунке 3.1.

```
<?xml version="1.0" encoding="UTF-8"?>
<Шаблон>
  <Документы>
    <Документ Ид="1" Название="Лабораторная работа" Тип="Файл"></Документ>
    <Документ Ид="2" Название="Комментарий проверяющего" Тип="Текст"></Документ>
    <Документ Ид="3" Название="Комментарий студента" Тип="Текст" Обязательный="true"></Документ>
  </Документы>
  <Этапы>
    <Этап Срок="3" Название="Этап 1: загрузка лабораторной работы">
      <ВходныеДокументы>
        <Документ Ид="2"></Документ>
        <Документ Ид="1"></Документ>
      </ВходныеДокументы>
      <ВыходныеДокументы>
        <Документ Ид="3"></Документ>
        <Документ Ид="1"></Документ>
      </ВыходныеДокументы>
    </Этап>
    <Этап Срок="1" Название="Этап 2: проверка" Возврат="true">
      <ВходныеДокументы>
        <Документ Ид="3"></Документ>
        <Документ Ид="1"></Документ>
      </ВходныеДокументы>
      <ВыходныеДокументы>
        <Документ Ид="2"></Документ>
      </ВыходныеДокументы>
    </Этап>
    <Этап Срок="1" Название="Этап 3: оценка" Возврат="true">
      <ВходныеДокументы>
        <Документ Ид="2"></Документ>
        <Документ Ид="1"></Документ>
      </ВходныеДокументы>
    </Этап>
  </Этапы>
</Шаблон>
```

Рисунок 3.1 – Структура шаблона маршрута для лабораторной работы

Для сдачи курсовой же работы необходимо пять этапов, по причине того, что данное задание, как правило, имеет две части: теоретическая и практическая. Вначале сдается теоретическая часть и осуществляется проверка, а после практическая часть, которая также проверяется. На последнем пятом этапе ставится оценка за все выполненное задание заранее назначенным исполнителем. Таким образом шаблон маршрута для сдачи курсовой работы будет выглядеть следующим образом, представленным на рисунке 3.2.

```
    </Документы>
    <Этапы>
        <Этап Срок="3" Название="Этап 1: загрузка теоретической части">
            <ВходныеДокументы>
                <Документ Ид="2"></Документ>
                <Документ Ид="1"></Документ>
            </ВходныеДокументы>
            <ВыходныеДокументы>
                <Документ Ид="3"></Документ>
                <Документ Ид="1"></Документ>
            </ВыходныеДокументы>
        </Этап>
        <Этап Срок="1" Название="Этап 2: проверка" Возврат="true">
            <ВходныеДокументы>
                <Документ Ид="3"></Документ>
                <Документ Ид="1"></Документ>
            </ВходныеДокументы>
            <ВыходныеДокументы>
                <Документ Ид="2"></Документ>
            </ВыходныеДокументы>
        </Этап>
        <Этап Срок="3" Название="Этап 3: загрузка практической части">
            <ВходныеДокументы>
                <Документ Ид="1"></Документ>
                <Документ Ид="5"></Документ>
                <Документ Ид="4"></Документ>
            </ВходныеДокументы>
            <ВыходныеДокументы>
                <Документ Ид="6"></Документ>
                <Документ Ид="4"></Документ>
            </ВыходныеДокументы>
        </Этап>
        <Этап Срок="1" Название="Этап 4: проверка" Возврат="true">
            <ВходныеДокументы>
                <Документ Ид="6"></Документ>
                <Документ Ид="1"></Документ>
                <Документ Ид="4"></Документ>
            </ВходныеДокументы>
            <ВыходныеДокументы>
                <Документ Ид="5"></Документ>
            </ВыходныеДокументы>
        </Этап>
        <Этап Срок="1" Название="Этап 5: оценка" Возврат="true">
            <ВходныеДокументы>
                <Документ Ид="2"></Документ>
                <Документ Ид="1"></Документ>
                <Документ Ид="5"></Документ>
                <Документ Ид="4"></Документ>
            </ВходныеДокументы>
        </Этап>
    </Этапы>
</Шаблон>
```

Рисунок 3.2 – Шаблон маршрута для курсовой работы

В данной нотации никак не описываются исполнители по причине того, что реализуется только последовательное движение документа. Для данного движения у каждого этапа может быть только один исполнитель, поэтому нет необходимости их описывать по аналогии с документами.

Текст программы для десериализации шаблона маршрута в формате XML представлен в приложении 1.

3.2 Структура базы данных

В web-приложении была реализована следующая база данных, представленная на рисунке 3.3.

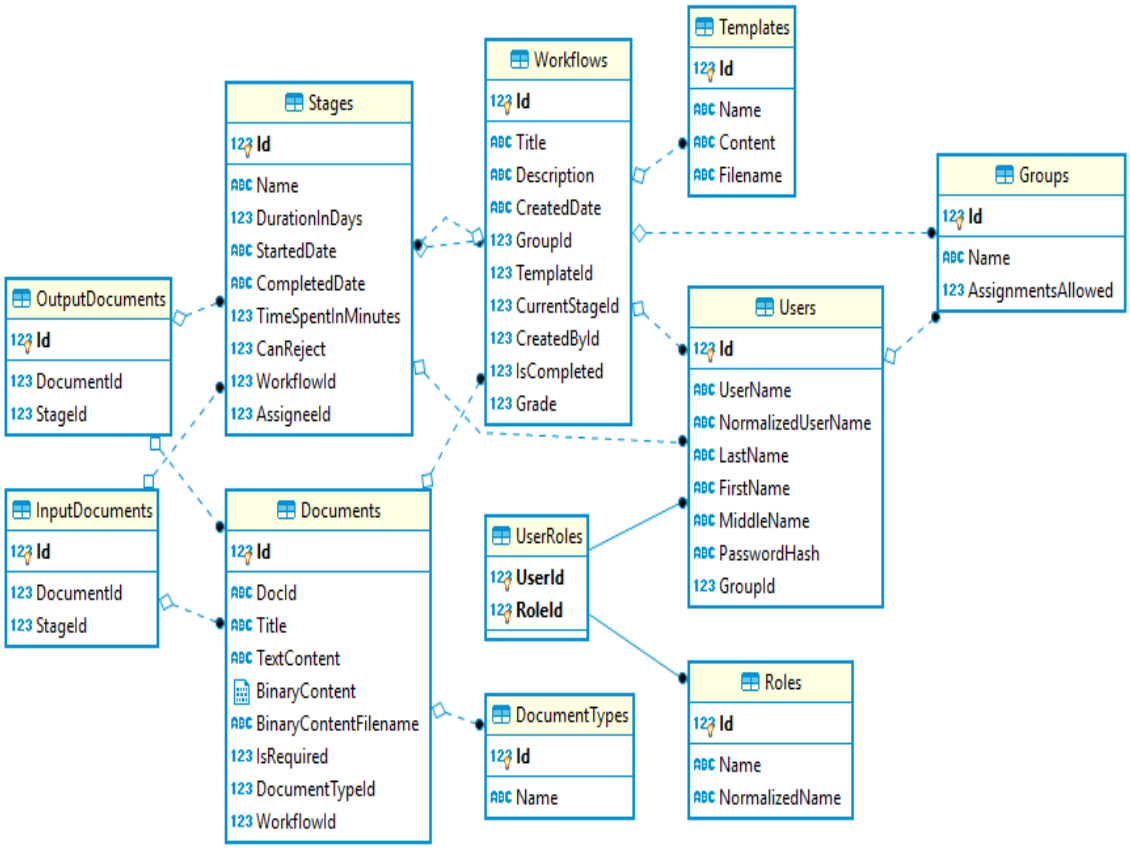

Рисунок 3.3 – Структура базы данных

Таблица «Roles» – список ролей пользователей. Содержит следующие поля:

- id – поле типа INTEGER. Идентификатор роли (первичный ключ);
- name – поле типа TEXT. Имя роли;

• normalizedName – поле типа TEXT. Имя роли в верхнем регистре. Служит для оптимизации поиска роли по его имени.

Таблица «Users» – пользователи системы. Содержит следующие поля:

• id – поле типа INTEGER. Идентификатор пользователя (является первичным ключом);

• userName – поле типа TEXT. Логин;

• normalizedUserName – поле типа TEXT. Логин в верхнем регистре. Служит для оптимизации поиска пользователя по его логину;

• lastName – поле типа TEXT. Фамилия;

• firstName – поле типа TEXT. Имя;

• middleName – поле типа TEXT. Отчество;

• passwordHash – поле типа TEXT. Хэш пароля;

• groupId – поле типа INTEGER. Идентификатор группы пользователя (внешний ключ).

Таблица «UserRoles» – таблица для организации связи между таблицами Users и Roles. Содержит следующие поля:

• userId – поле типа INTEGER. Идентификатор пользователя (внешний ключ, часть первичного ключа);

• roleId – поле типа INTEGER. Идентификатор роли (внешний ключ, часть первичного ключа).

Таблица «Groups» – группы пользователей. Содержит следующие поля:

• id – поле типа INTEGER. Идентификатор группы (первичный ключ);

• name – поле типа TEXT. Название группы;

• assignmentsAllowed – поле типа INTEGER. Признак наличия разрешения на выдачу заданий.

Таблица «Templates» – данные шаблонов маршрутов. Содержит следующие поля:

• id – поле типа INTEGER. Идентификатор шаблона (первичный ключ);

• name – поле типа TEXT. Имя шаблона;

- content – поле типа TEXT. XML-содержимое;
- filename – поле типа TEXT. Имя загруженного файла шаблона.

Таблица «OutputDocuments» – выходные документы этапа. Содержит следующие поля:

- id – поле типа INTEGER. Идентификатор (первичный ключ);
- documentId – поле типа INTEGER. Идентификатор документа (внешний ключ);
- stageId – поле типа INTEGER. Идентификатор этапа (внешний ключ).

Таблица «InputDocuments» – входные документы этапа. Содержит следующие поля:

- id – поле типа INTEGER. Идентификатор (первичный ключ);
- documentId – поле типа INTEGER. Идентификатор документа (внешний ключ);
- stageId – поле типа INTEGER. Идентификатор этапа (внешний ключ).

Таблица «Documents» – документы, определенные в маршруте. Содержит следующие поля:

- id – поле типа INTEGER. Идентификатор документа (является первичным ключом);
- docId – поле типа TEXT. Текстовый идентификатор документа в пределах шаблона;
- title – поле типа TEXT. Название документа;
- textContent – поле типа TEXT. Данные, введенные пользователем при выполнении задания (для документов с типом «Текст»);
- binaryContent – поле типа BLOB. Содержимое файла, загруженного пользователем при выполнении задания (для документов с типом «Файл»);
- binaryContentFilename – поле типа TEXT. Имя файла, загруженного пользователем;
- isRequired – поле типа INTEGER. Признак, указывающий на обязательность документа при выполнении задания;

• documentTypeId – поле типа INTEGER. Идентификатор типа документа (внешний ключ);

• workflowId – поле типа INTEGER. Идентификатор задания (является внешним ключом).

Таблица «Stages» – этапы маршрута. Содержит следующие поля:

• id – поле типа INTEGER. Идентификатор этапа (первичный ключ);

• name – поле типа TEXT. Имя этапа;

• durationInDays – поле типа INTEGER. Срок выполнения этапа (количество дней);

• startedDate – поле типа TEXT. Дата начала выполнения этапа;

• completedDate – поле типа TEXT. Дата завершения выполнения этапа;

• timeSpentInMinutes – поле типа INTEGER. Суммарное количество минут, затраченное на выполнение этапа;

• canReject – поле типа INTEGER. Признак, указывающий на возможность пользователя вернуть задание на предыдущий этап;

• workflowId – поле типа INTEGER. Идентификатор задания (является внешним ключом);

• assigneeId – поле типа INTEGER. Идентификатор пользователя-исполнителя этапа (внешний ключ).

Таблица «Workflows» – выданные задания. Содержит следующие поля:

• id – поле типа INTEGER. Идентификатор задания (первичный ключ);

• title – поле типа TEXT. Название задания;

• description – поле типа TEXT. Описание задания;

• createdDate – поле типа TEXT. Дата создания задания;

• groupId – поле типа INTEGER. Идентификатор группы (является внешним ключом);

• templateId – поле типа INTEGER. Идентификатор шаблона (является внешним ключом);

• currentStageId – поле типа INTEGER. Идентификатор текущего этапа (внешний ключ);

• createdById – поле типа INTEGER. Идентификатор пользователя, создавшего задание (внешний ключ);

• isCompleted – поле типа INTEGER. Признак завершенности задания;

• grade – поле типа INTEGER. Оценка, выставленная при завершении задания.

Для доступа к данной базе данных используется технология Entity Framework Core, суть которой в следующем: для каждой таблицы определяются классы модели с нужными полями, а также связи между ними, первичные ключи, ограничения и т. д., а после ASP.NET Core строит полную объектную модель и предоставляет удобные методы для работы с данными без необходимости написания SQL-запросов. В следствии этого данный метод работы с таблицами базы данных исключает возможность внедрения любых SQL-инъекций.

3.3 Разработка пользовательского интерфейса

В разработке интерфейса web-приложения использовалась библиотека «Bootstrap».

Bootstrap – свободный набор инструментов для создания сайтов и web-приложений. Включает в себя HTML и CSS-шаблоны оформления для типографики, web-форм, кнопок, меток, блоков навигации и прочих компонентов web-интерфейса, включая JavaScript-расширения [9].

Каждая страница web-приложения является динамической. Это достигается за счет инструментальных средств ASP.NET Core Razor Pages, который разделяет каждую страницу на две части.

Одна часть имеет расширение .cshtml.cs и выступает в роли модели, а также имеет два метода: OnGetasync и OnPostasync.

OnGetasync – данный метод вызывается при загрузке страницы и предназначен для наполнения классов модели значениями из базы данных.

OnPostasync – данный метод вызывается при отправке со страницы формы с введенными значениями. Предназначен для обработки полученных значений и сохранения их в базе данных.

Вторая часть имеет расширение .cshtml и является шаблоном web-страницы, отображающий данные из его модели.

Интерфейс пользователя приложения разрабатывался с учётом требований простоты, удобства и эргономичности. Начальная страница приложения представляет собой страницу авторизации, где пользователю необходимо ввести логин и пароль для входа в систему (рисунок 3.4).

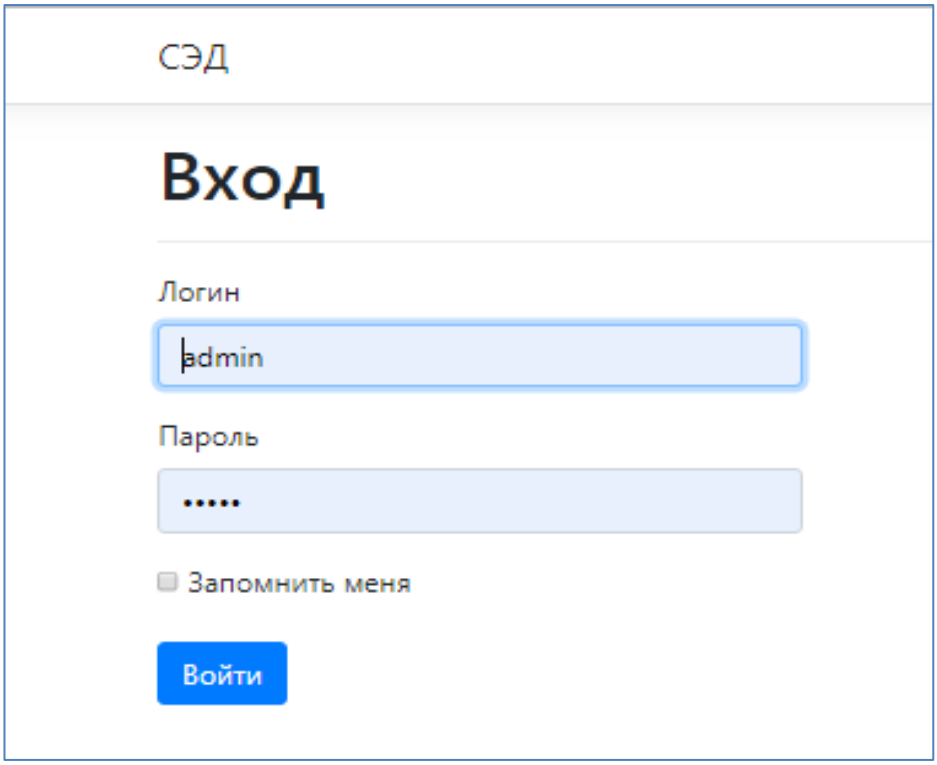

Рисунок 3.4 – Начальная страница, страница авторизации

После прохождения авторизации панель администратора выглядит следующим образом (рисунок 3.5).

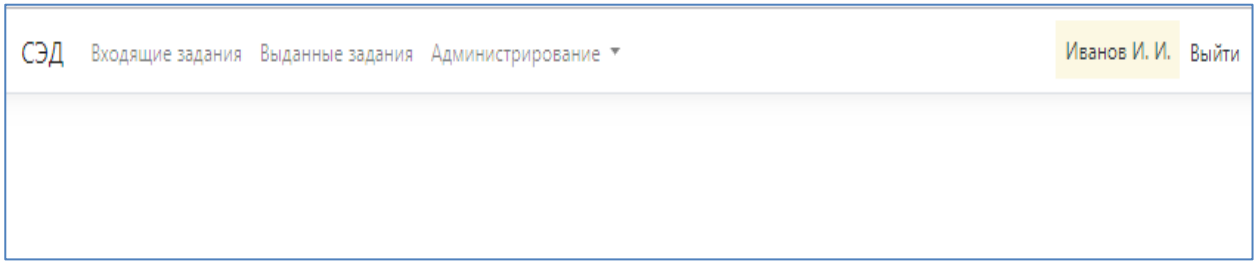

Рисунок 3.5 – Панель администратора

В верхней части экрана располагается панель управления с которой будет работать администратор. Данное меню является динамическим и если администратор одновременно имеет роль пользователя, то появляются две вкладки «Входящие задания» и «Выданные задания».

Рассмотрим вкладку «Администрирование». Она включает в себя страницу для создания групп, пользователей и загрузки шаблонов.

Web-страница для просмотра созданных групп представлена на рисунке 3.6. Все группы на данной странице представлены в виде списка в таблице.

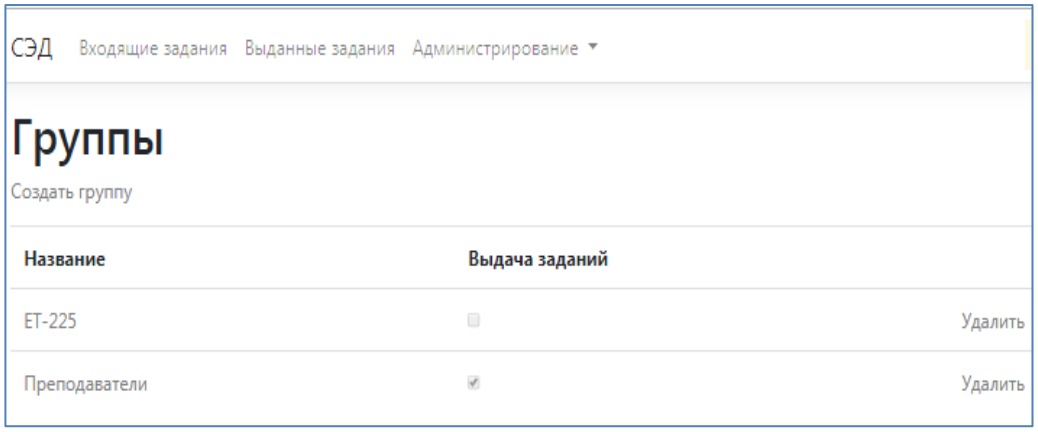

Рисунок 3.6 – Страница отображения всех групп

По нажатию на "Создать группу" отобразиться следующая форма (рисунок 3.7).

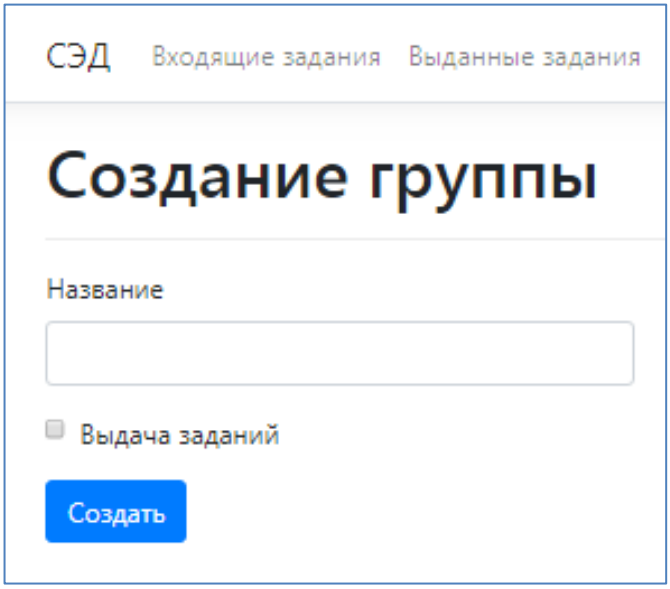

Рисунок 3.7 – Страница создания групп

Данная форма содержит одно поле для ввода названия группы и метку «Выдача заданий». Если создать группу с данной меткой, то всем пользователям, которые будут добавлены в группу станет доступна функция выдачи задания.

По нажатию «Удалить» отобразиться сообщение с подтверждением для данного действия, а также все ранее введенные параметры для данной группы, включая состав пользователей (рисунок 3.8).

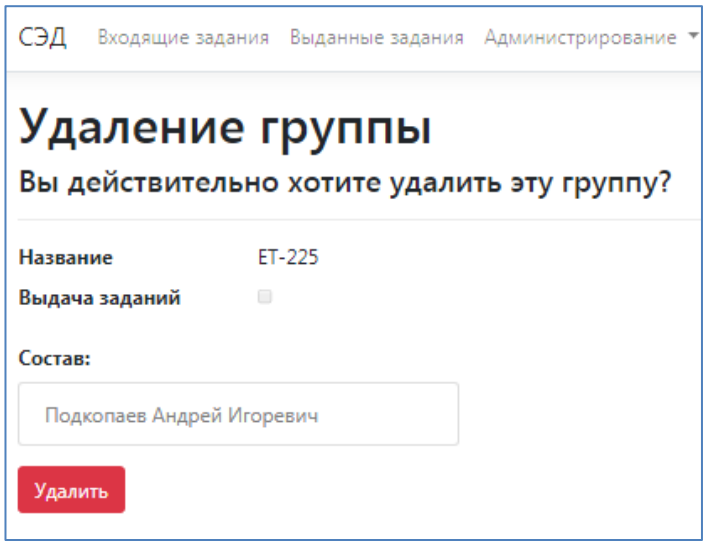

Рисунок 3.8 – Страница удаления группы

Для редактирования необходимо нажать на имя группы в таблице, тогда отобразятся все параметры группы с двумя функциями на выбор «Редактировать» и «Удалить» (рисунок 3.9).

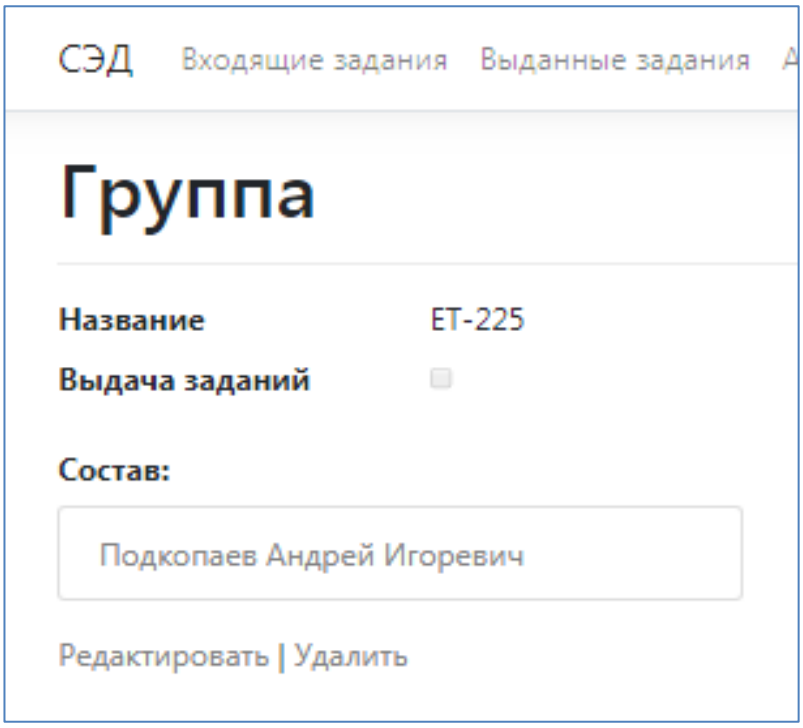

Рисунок 3.9 – Страница для просмотра группы

По нажатию на редактировать отображается web-страница с возможностью изменить название, доступ к функции «Выдача задания», а также изменить состав пользователей в группе (рисунок 3.10).

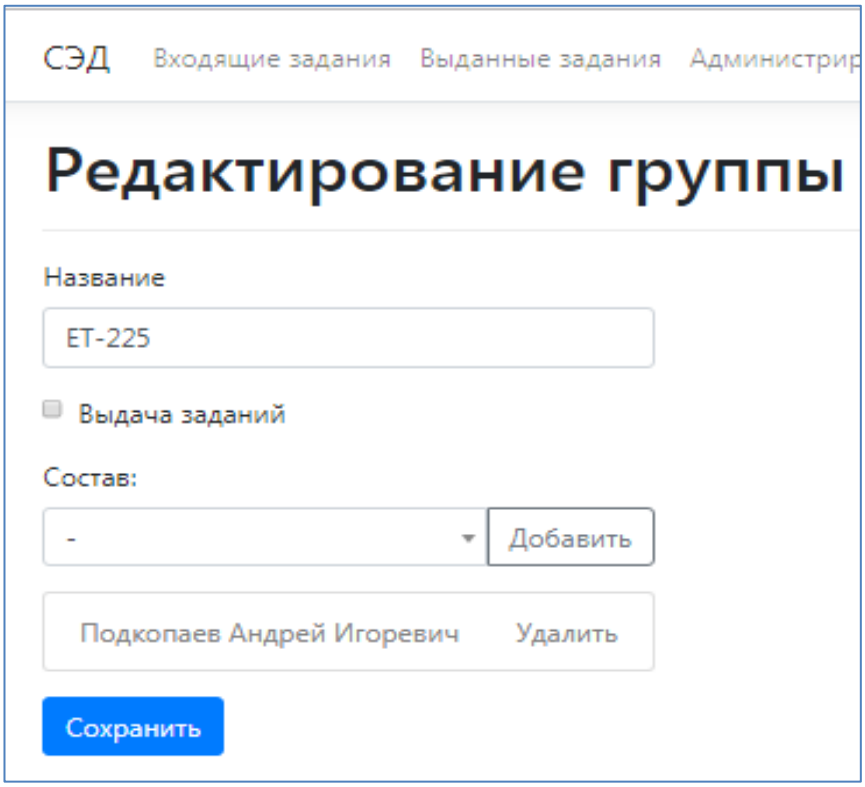

Рисунок 3.10 – Страница для редактирования группы

Далее рассмотрим пункт меню «Пользователи». Данный пункт отображает всех созданных пользователей в таблице и предоставляет возможность использовать те же функции, что и на странице «Группы», а именно: создание, редактирование и удаление пользователей (рисунок 3.11).

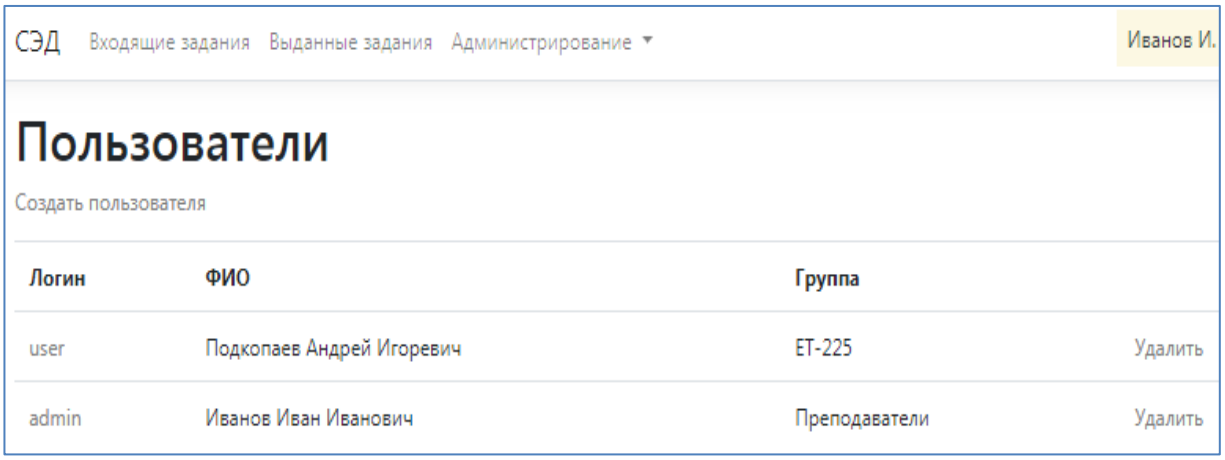

Рисунок 3.11 – Страница просмотра списка пользователей

Страница для создания нового пользователя содержит стандартные поля: логин, фамилия, имя, отчество, группа, пароль и подтверждение пароля (рисунок 3.12).

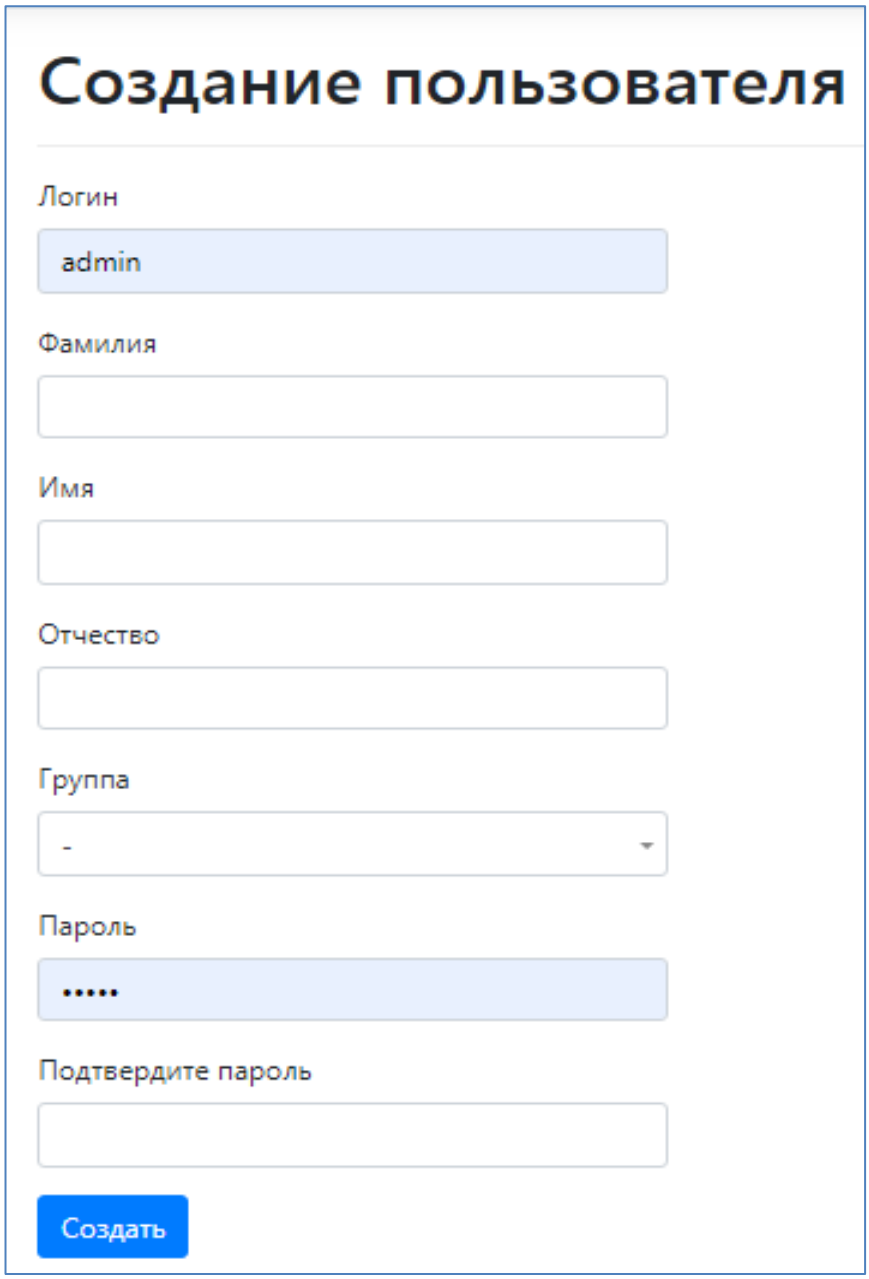

Рисунок 3.12 – Страница для добавления пользователя

Следующий немаловажный пункт — это «Шаблоны». Данная страница отображает список всех загруженных шаблонов маршрутов в формате XML, а также предоставляет по аналогии с предыдущими пунктами функции для добавления, редактирования и удаления шаблонов (рисунок 3.13).

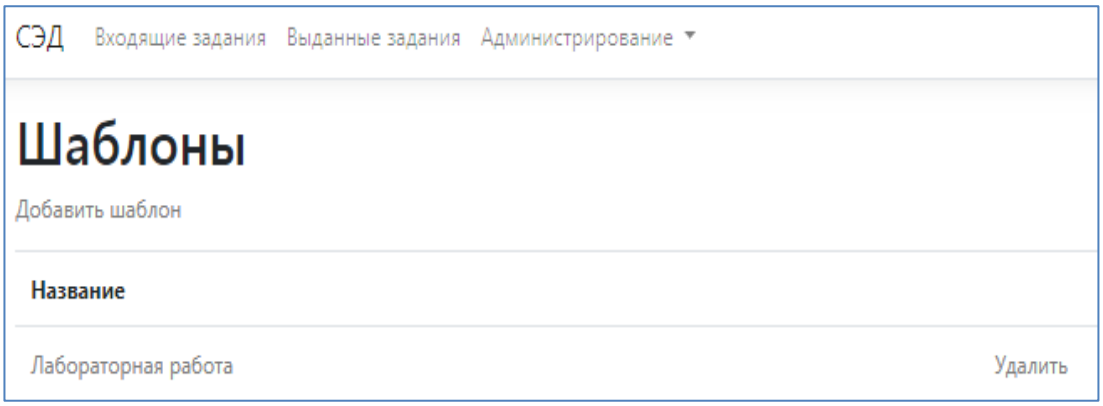

Рисунок 3.13 – Страница просмотра списка шаблонов маршрутов

Страница добавления шаблона представлена на рисунке 3.14.

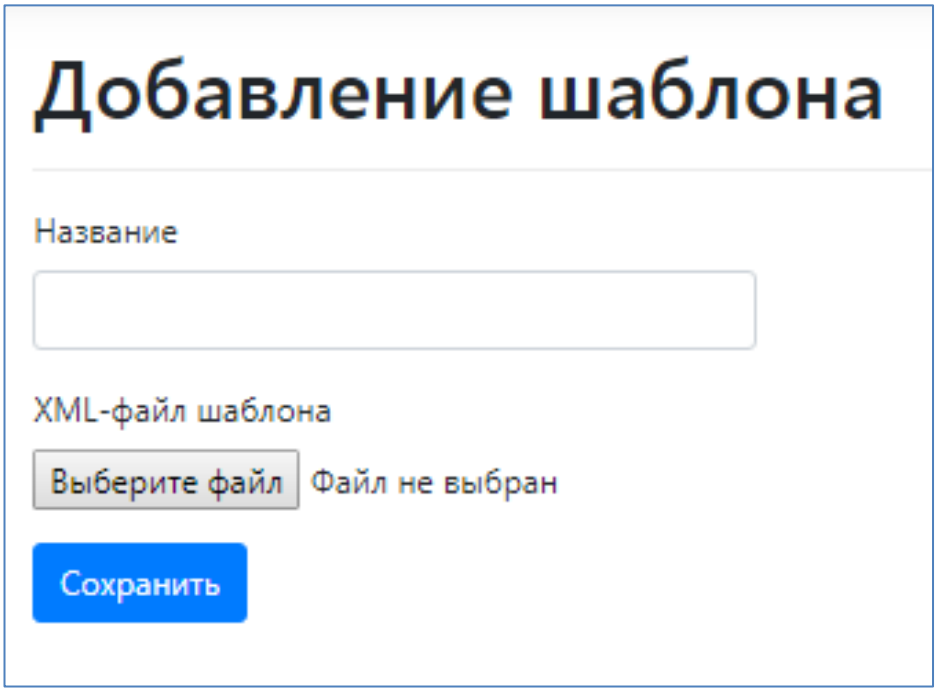

Рисунок 3.14 – Страница добавления шаблона маршрута

Два оставшихся пункта в меню «Входящие задания» и «Выданные задания» реализуют главный функционал всего web-приложения. Данные пункты отображаются у всех, кто имеет роль «User», однако если у группы в которой они состоят нет разрешения на выдачу задания, то web-страница «Выданные задания» не будет содержать в себе ссылку на страницу реализующий данный функционал.

«Выданные задания» – данная страница отображает список уже выданных данным пользователем заданий. Каждый элемент списка содержит в себе этапы внутри которого располагается ФИО исполнителя и треугольник обозначающий, что выполнения задания находится на данном этапе. Как

только задание будет завершено, то рядом с этапами будет находится оценка (рисунок 3.15).

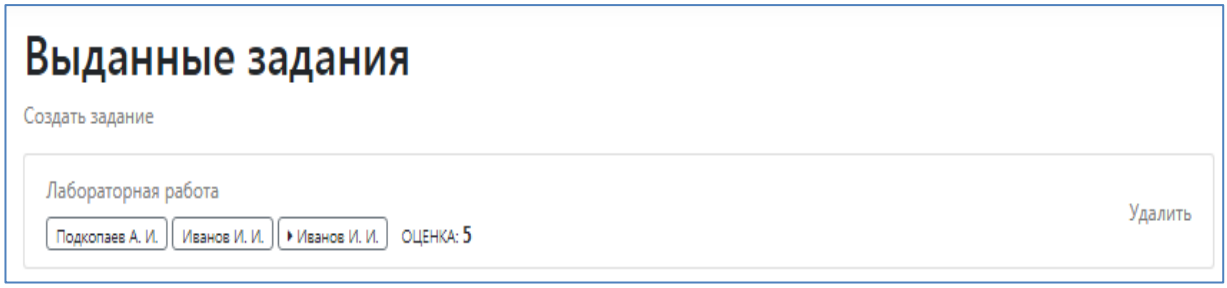

Рисунок 3.15 – Страница отображения списка назначенных заданий

В случае если крайний срок для этапа будет просрочен исполнителем, то у пользователя, который выдал задание данный этап будет помечен красным цветом. Также данная страница дает возможность получить более подробную информацию о задании, если нажать на его название (рисунок 3.16)

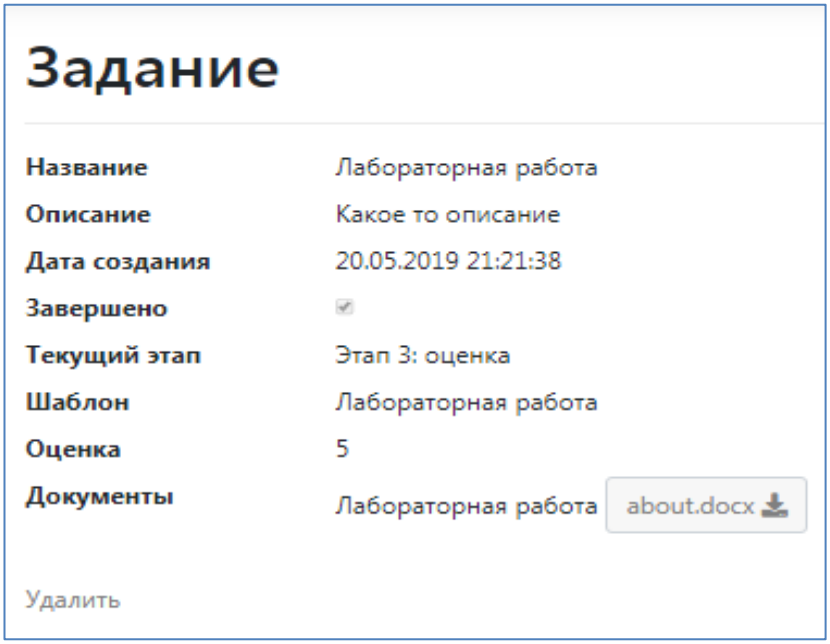

Рисунок 3.16 – Страница с подробной информацией задания

Как видно, данная страница содержит всю необходимую информацию, а именно: название, описание, дата создания, статус завершенности, название этапа, на котором находится выполнение задания, название используемого шаблона маршрута и поле для оценки. Однако после завершения задания

данная страница отображает также ссылки на документы, которые были загружены в ходе выполнения задания.

Для создания задания есть ссылка «Создать задание» по нажатию на которую отобразится список шаблонов маршрутов, загруженные администратором (рисунок 3.17).

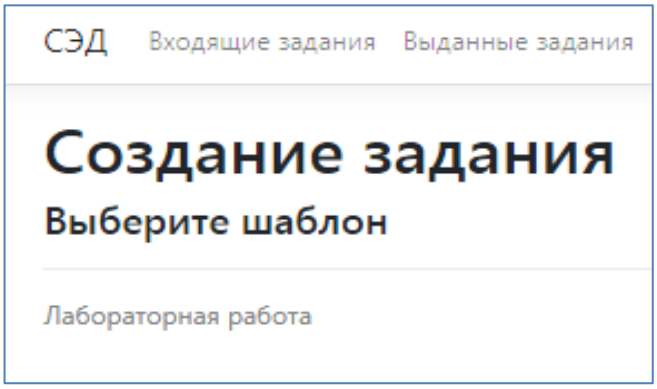

Рисунок 3.17 – Страница со списком шаблонов маршрута

По нажатию на название шаблона маршрута произойдет парсинг XML путем десериализации нотации шаблона маршрута в объекты классов модели. Десериализация выполняется методами из стандартной библиотеки для работы с XML файлами, предоставляемая .NET. Далее заполненные объекты классов используются для отображения на форме нужного количества этапов с их названиями и полями выбора исполнителя (рисунок 3.18).

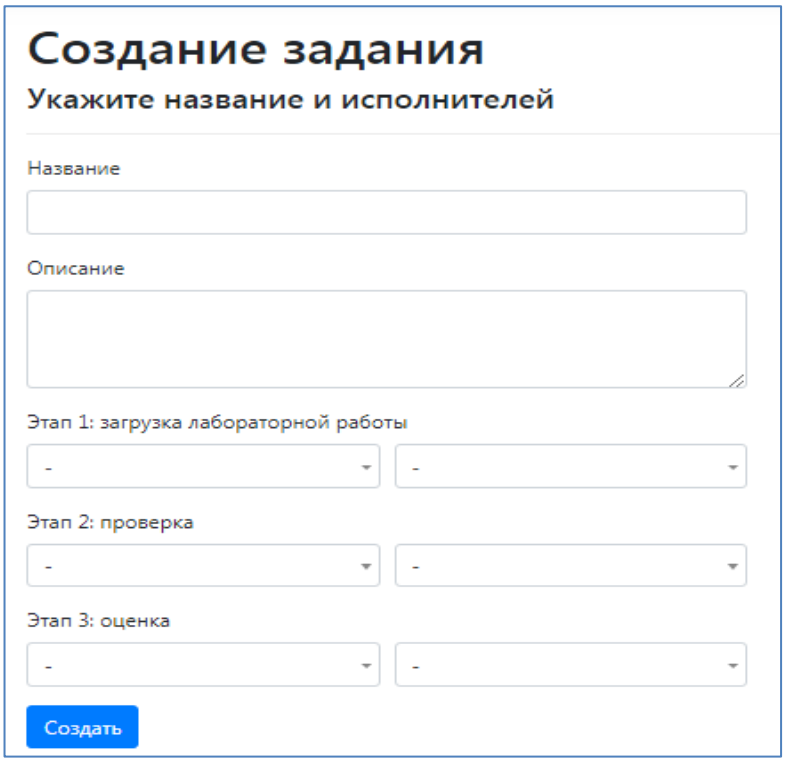

Рисунок 3.18 – Страница создания задания

Данная форма содержит поля для названия и описания задания независимо от шаблона маршрута по той причине, что они являются статическими, то есть будут отображаться для каждого этапа по ходу выполнения задания. Остальные же поля являются динамическими и соответствуют количеству этапов, которые были заданы в шаблоне «Лабораторная работа», а именно три этапа.

Сам же шаблон маршрута «Лабораторная работа» имеет структуру, представленную на рисунке 3.19.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<Шаблон>
  <Документы>
    <Документ Ид="1" Название="Лабораторная работа" Тип="Файл"></Документ>
    <Документ Ид="2" Название="Комментарий проверяющего" Тип="Текст"></Документ>
    <Документ Ид="3" Название="Комментарий студента" Тип="Текст" Обязательный="true"></Документ>
  </Документы>
  <Этапы>
    <Этап Срок="3" Название="Этап 1: загрузка лабораторной работы">
      <ВходныеДокументы>
        <Документ Ид="2"></Документ>
        <Документ Ид="1"></Документ>
      </ВходныеДокументы>
      <ВыходныеДокументы>
        <Документ Ид="3"></Документ>
        <Документ Ид="1"></Документ>
      </ВыходныеДокументы>
    </Этап>
    <Этап Срок="1" Название="Этап 2: проверка" Возврат="true">
      <ВходныеДокументы>
        <Документ Ид="3"></Документ>
        <Документ Ид="1"></Документ>
      </ВходныеДокументы>
      <ВыходныеДокументы>
        <Документ Ид="2"></Документ>
      </ВыходныеДокументы>
    </Этап>
    <Этап Срок="1" Название="Этап 3: оценка" Возврат="true">
      <ВходныеДокументы>
        <Документ Ид="2"></Документ>
        <Документ Ид="1"></Документ>
      </ВходныеДокументы>
    </Этап>
  </Этапы>
</Шаблон>
```

Рисунок 3.19 – Структура шаблона маршрута «Лабораторная работа»

«Входящие задания» – ничего не содержит до тех пор, пока пользователя не назначат исполнителем для какого-либо этапа (рисунок 3.20).

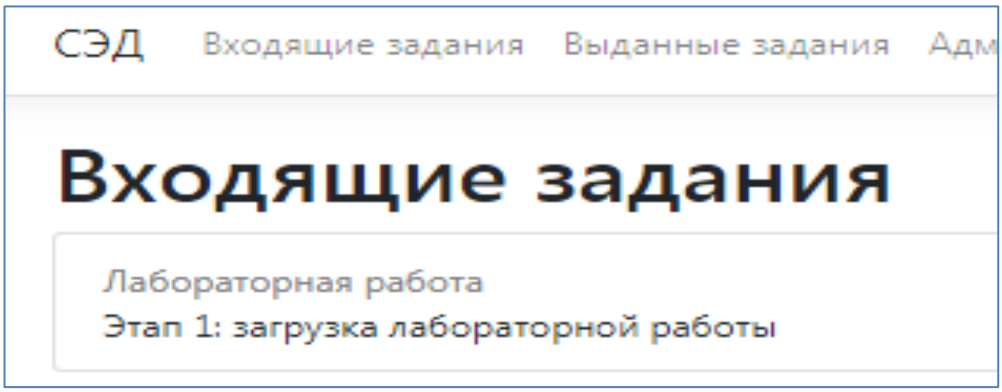

Рисунок 3.20 – Страница со списком назначенных заданий

По нажатию на название задания отобразиться форма, включающая в себя два статических поля (название и описание задания) и два динамических поля. Динамические поля в форме генерируются в зависимости от количества входных и выходных документов, описанных в шаблоне маршрута «Лабораторная работа». В случае если значение входного документа будет пустым, то оно никак не будет отображаться на web-странице (рисунок 3.21).

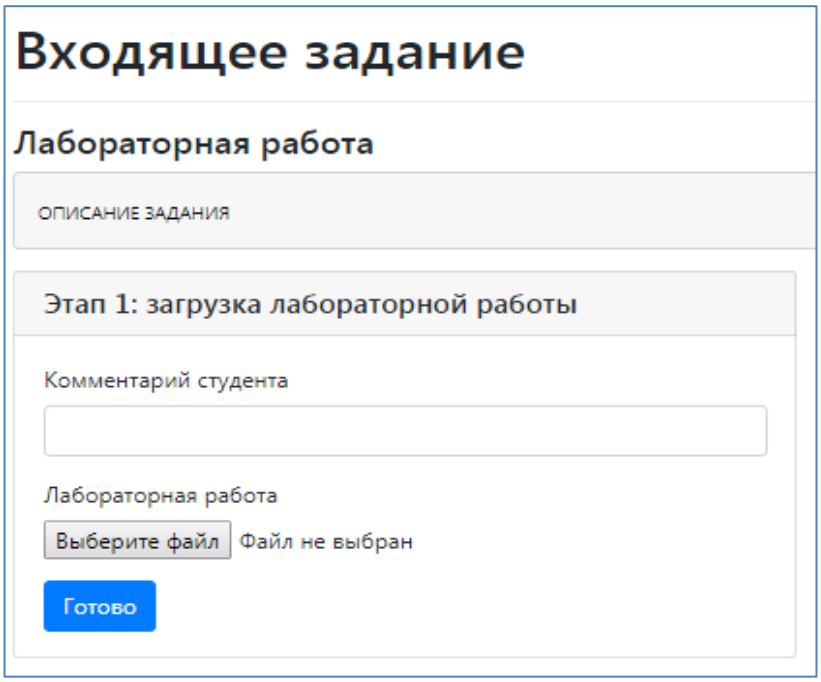

Рисунок 3.21 – Страница для отображения этапов

Стоит отметить, что в шаблоне маршрута для определенных этапов также описывается функция возврата на предыдущий этап. По этой причине для второго этапа маршрута «Лабораторная работа» в форме появиться кнопка «Возврат» (рисунок 3.22).

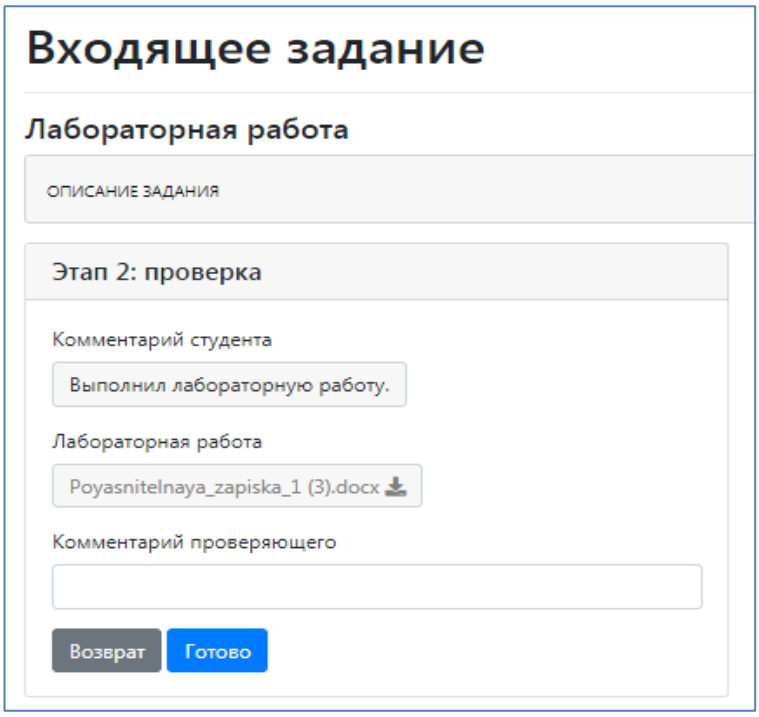

Рисунок 3.22 – Страница для проверки задания

На последнем этапе любого маршрута всему заданию выставляется оценка, по этой причине в форме всегда будет присутствовать статическое поле «Оценка» (рисунок 3.23).

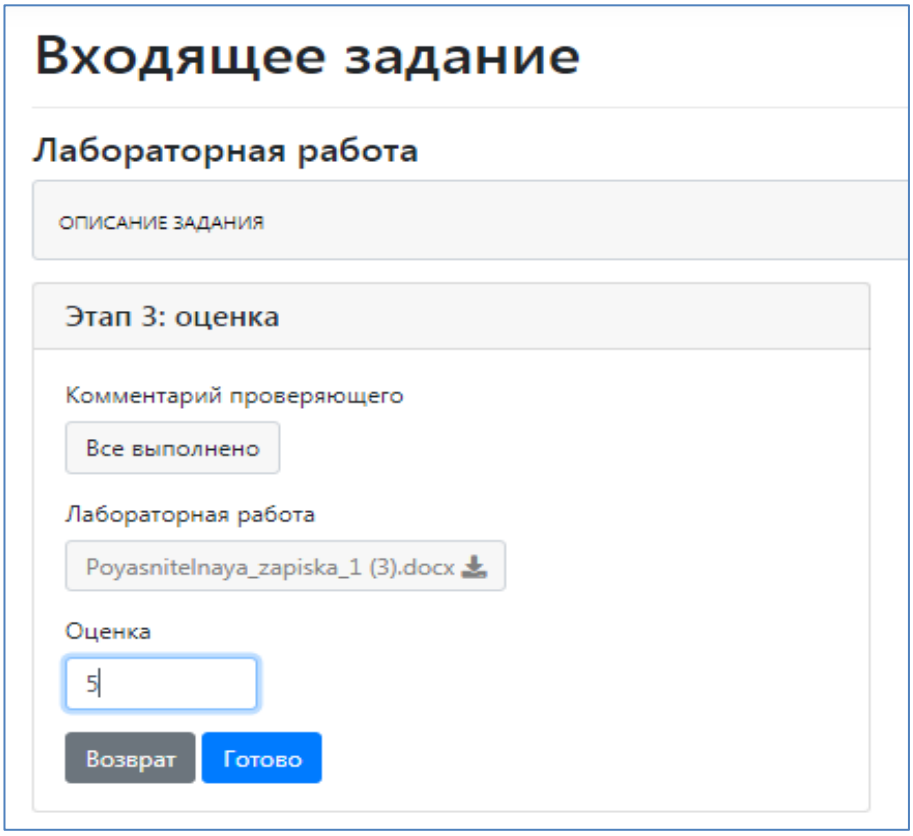

Рисунок 3.23 – Страница для последнего этапа маршрута

3.4 Безопасность web-приложения

Ниже представлены все уязвимости, которые могли возникнуть в разработанном web-приложении, а также инструментальные средства для их противодействия:

1) инъекции (Injection). Для доступа к данным в приложении используется object-relational mapping (ORM) решение, что является наиболее эффективным способом защиты от SQL-инъекций;

2) межсайтовый скриптинг – XSS (Cross-Site Scripting). Технология Razor Pages по умолчанию применяет HTML encoding при рендеринге значений любых переменных на странице, в т. ч. и при рендеринге пользовательского ввода, что исключает возможность произвести XSS-атаку. Подстановка пользовательского ввода в контексте JavaScript или при построении URL в приложении не происходит вообще;

3) небезопасные прямые ссылки на объекты (Insecure Direct Object References). В любой точке, где пользователь может изменить состояние объекта, проверяется, действительно ли пользователь имеет право на данное действие над запрошенным объектом. Таким образом пользователь не может путем подмены данных запроса изменить состояние объекта, к которому у него нет доступа. В web-приложении есть две такие точки: удаление и сдача задания;

4) незащищенность критичных данных (Sensitive Data Exposure). В качестве конфиденциальных данных, подлежащих защите, в приложении можно рассматривать только хэши паролей пользователей. В ASP.NET Core для создания хэшей паролей с солью используется стандарт PBKDF2. В Российской Федерации использование функции PBKDF2 регламентируется рекомендациями по стандартизации Р 50.1.111-2016 "Парольная защита ключевой информации";

5) отсутствие функций контроля доступа (Missing Function Level Access Control). Каждый запрос пользователя, который должен быть авторизован,

перед обработкой проходит автоматическую авторизацию на основе роли пользователя. При этом функционал авторизации не полагается на данные, переданные пользователем. Таким образом, пользователь не может получить доступ к непредназначенной для него информации путем подмены данных запроса или перехода по URL, не представленном в интерфейсе приложения;

6) межсайтовая подделка запроса – CSRF (Cross-Site Request Forgery). ASP.NET Core предоставляет автоматическую защиту от CSRF. Для этого на любой странице с формой приложение добавляет скрытое поле со значением antiforgery token. Это зашифрованное уникальное случайное значение, ассоциированное с учетной записью текущего пользователя. При отправке формы на сервере происходит валидация данного значения, и если оно отсутствует, или его не удается дешифровать (т.е. значение было модифицировано), или данные пользователя не совпадают, то запрос отклоняется.

3.5 Выводы по разделу

В данном разделе была разработана собственная нотация шаблона маршрута в формате XML для организации последовательного движения документов. Разработана структура базы данных, хранящая экземпляры маршрута и информацию о пользователях, а также задействована технология Entity Framework Core, которая предоставляет удобные методы для работы с данными без необходимости написания SQL-запросов. Разработан пользовательский интерфейс приложения. Визуальное оформление элементов пользовательского интерфейса было взято из библиотеки «Bootstrap». Страницы web-приложения динамически обновляются при помощи встроенных инструментальных средств ASP.NET Core Razor Pages,

ЗАКЛЮЧЕНИЕ

В данной выпускной квалификационной работе исследовались различные аспекты разработки электронного документооборота для сдачи лабораторных и курсовых работ обучающихся.

Во время выполнения выпускной квалификационной работы было сделано следующее:

1) была изучена теория по системам электронного документооборота, а именно: определения, термины, свойства и компоненты характерные для СЭД;

2) проведено исследование существующих аналогов разработок и их сравнение;

3) выбраны языки программирования для разработки web-приложения: для серверной части – C#, а для клиентской – JavaScript, CSS, HTML;

4) выбрана интегрированная среда разработки Visual Studio по причине предоставления множество возможностей для реализации, а также встроенного web-сервера;

5) проведен обзор средств разработки базы данных по итогу которого был выбран SQlite;

6) проведен обзор фреймворков предоставляемый интегрированной средой разработки Visual Studio. В следствии которого был выбран ASP.NET Core с моделью построения web-приложения Razor Pages;

7) разработана нотация для шаблона маршрута в формате XML;

8) разработана структура базы данных для хранения экземпляров маршрута, а также для взаимодействия с ней была выбрана технология Entity Framework Core;

9) разработан пользовательский интерфейс для web-приложения.

В результате проделанной работы было разработано web-приложение для сдачи лабораторных и курсовых работ обучающихся.

Таким образом, все поставленные задачи были успешно выполнены и цель работы достигнута.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Электронный документооборот. – Дата обновления: 07.03.2017. URL: https://cyberleninka.ru/article/v/elektronnyy-dokumentooborot (Дата обращения: 31.03.2018)

2. Система электронного документооборота. – Дата обновления: 12.11.2014. URL: https://cyberleninka.ru/article/v/sistema-elektronnogo-dokumentooborota (Дата обращения: 13.04.2018)

3. Особенности защиты информации в системах электронного документооборота. – Дата обновления: 13.10.2013. URL: https://cyberleninka.ru /article/v/osobennosti-zaschity-informatsii-v-sistemah-elektronnogo-dokumentooborota (Дата обращения: 13.04.2018)

4. Природа бизнес-процесса: в поисках утраченного workflow. – Дата обновления: 02.09.2016. URL: https://cyberleninka.ru/article/v/priroda-biznes-protsessa-v-poiskah-utrachennogo-workflow (Дата обращения: 18.07.2018)

5. Анализ систем электронного документооборота в органах муниципального управления. – Дата обновления: 02.06.2012. URL: https://cyberleninka.ru/article/v/ analiz-sistem-elektronnogo-dokumentooborota-v-organah-munitsipalnogo-upravleniya (Дата обращения: 17.05.2018)

6. Реализация workflow в отечественных и зарубежных системах электронного документооборота. – Дата обновления: 18.04.2008. URL: https:// cyberleninka.ru/article/v/realizatsiya-workflow-v-otechestvennyh-i-zarubezhnyh-sistemah-elektronnogo-dokumentooborota. (Дата обращения: 22.02.2018)

7. Электронный документооборот: технология внедрения и способ оптимизации бизнес-процедур. – Дата обновления: 02.09.2012. URL: https:// cyberleninka.ru/article/v/elektronnyy-dokumentooborot-tehnologiya-vnedreniya-i-sposob-optimizatsii-biznes-protsedur (Дата обращения: 23.06.2018)

8. 5 шагов к процессному управлению. – Дата обновления: 20.05.2002. URL: https://pqm-online.com/assets/files/books/intalev.pdf (Дата обращения: 22.02.2018).

9. Жаринов, К.В. Основы веб-мастеринга: учебник / К.В. Жаринов. – СПб.: БХВ-Петербург, 2003. – 352 с.

10. Мэрдок, К. JavaScript. Наглядный курс создания динамических Web-страниц: учебник / К. Мэрдок. – М.: Вильямс, 2001. – 288 с.

11. Эффективность применения технологии ASP.NET Core для разработки веб-приложений. – Дата обновления: 02.09.2018. URL: https://cyberleninka.ru /article/v/effektivnost-primeneniya-tehnologii-asp-net-core-dlya-razrabotki-veb-prilozheniy (Дата обращения: 23.02.2019)

12. Хомоненко, А.Д. Базы данных: учебник / А.Д. Хомоненко, В.М. Цыганков, М.Г. Мальцев. – СПб.: КОРОНА-Век, 2009. – 736 с.

13. Ржеуцкая, С.Ю. Базы данных. Язык SQL: учеб.пособие / С.Ю. Ржеуцкая – Вологда: ВоГТУ, 2010. – 159с.

14. Архитектура технологии разработки веб-приложений ASP.NET Core MVC. – Дата обновления: 02.03.2018. URL: https://cyberleninka.ru/article/v/ arhitektura-tehnologii-razrabotki-veb-prilozheniy-asp-net-core-mvc (Дата обращения: 23.02.2019)

ПРИЛОЖЕНИЯ

ПРИЛОЖЕНИЕ 1

Текст программы

Модель ApplicationContext.cs, предоставляющая доступ ко всем остальным моделям.

```
using Microsoft.EntityFrameworkCore;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace StudentAssignments.Web.Model
{
    public class ApplicationContext : DbContext
    {
        public
ApplicationContext(DbContextOptions<ApplicationContext> options)
            : base(options)
        { }

        public DbSet<User> Users { get; set; }
        public DbSet<Role> Roles { get; set; }
        public DbSet<UserRole> UserRoles { get; set; }
        public DbSet<Group> Groups { get; set; }
        public DbSet<Template> Templates { get; set; }
        public DbSet<Workflow> Workflows { get; set; }
        public DbSet<Document> Documents { get; set; }
        public DbSet<DocumentType> DocumentTypes { get; set; }
        public DbSet<Stage> Stages { get; set; }
        public DbSet<InputDocument> InputDocuments { get; set;
}
        public DbSet<OutputDocument> OutputDocuments { get;
set; }

        protected override void OnModelCreating(ModelBuilder
modelBuilder)
        {
            modelBuilder.Entity<UserRole>()
                .HasKey(x => new { x.UserId, x.RoleId });
        }
    }
}
```

Модель Model.cs использующаяся для десереализации XML документа.

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

namespace StudentAssignments.Web.Xml
{
    [XmlRoot("Шаблон")]
    public class WorkflowTemplate
    {
        [XmlArray("Документы")]
        [XmlArrayItem("Документ")]
        public Document[] Documents { get; set; }
        [XmlArray("Этапы")]
        [XmlArrayItem("Этап")]
        public Stage[] Stages { get; set; }
    }

    public class Document
    {
        [XmlAttribute(AttributeName = "Ид")]
        public string Id { get; set; }
        [XmlAttribute(AttributeName = "Название")]
        public string Title { get; set; }
        [XmlAttribute(AttributeName = "Тип")]
        public string Type { get; set; }
        [XmlAttribute(AttributeName = "Обязательный")]
        public bool IsRequired { get; set; } = true;
    }

    public class Stage
    {
        [XmlAttribute(AttributeName = "Название")]
        public string Title { get; set; }
        [XmlAttribute(AttributeName = "Срок")]
        public int Duration { get; set; }
        [XmlAttribute(AttributeName = "Возврат")]
        public bool CanReject { get; set; }
        [XmlArray("ВходныеДокументы")]
        [XmlArrayItem("Документ")]
        public InputDocument[] InputDocuments { get; set; }
        [XmlArray("ВыходныеДокументы")]
        [XmlArrayItem("Документ")]
        public OutputDocument[] OutputDocuments { get; set; }
    }

    public class InputDocument
    {
        [XmlAttribute(AttributeName = "Ид")]
        public string Id { get; set; }
    }

    public class OutputDocument
    {
```

```
            [XmlAttribute(AttributeName = "Ид")]
            public string Id { get; set; }
        }
    }
```

Сервис, выполняющий десереализацию XML файла.

```
using Microsoft.EntityFrameworkCore;
using StudentAssignments.Web.Model;
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Threading.Tasks;
using System.Xml.Serialization;
using StudentAssignments.Web.Xml;

namespace StudentAssignments.Web.Services
{
    public class WorkflowService
    {
        private readonly ApplicationContext _context;

        public WorkflowService(ApplicationContext context)
        {
            _context = context;
        }

        public WorkflowTemplate DeserializeXmlTemplate(string
templateXml)
        {
            WorkflowTemplate template;
            XmlSerializer serializer = new
XmlSerializer(typeof(WorkflowTemplate));
            using (var reader = new StringReader(templateXml))
            {
                template =
(WorkflowTemplate)serializer.Deserialize(reader);
            }
            return template;
        }
    }
}
```

Модель представления, использующаяся для парсинга XML и создания нового маршрута в БД.

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Authorization;
```

```
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.RazorPages;
using Microsoft.AspNetCore.Mvc.Rendering;
using Microsoft.EntityFrameworkCore;
using StudentAssignments.Web.Model;
using StudentAssignments.Web.Services;

namespace StudentAssignments.Web.Pages.Assignment
{
    public class CreateModel : PageModel
    {
        private readonly
StudentAssignments.Web.Model.ApplicationContext _context;
        private readonly WorkflowService _workflowService;
        private readonly UserManager<User> _userManager;

        public CreateModel(
            StudentAssignments.Web.Model.ApplicationContext
context,
            WorkflowService workflowService,
            UserManager<User> userManager)
        {
            _context = context;
            _workflowService = workflowService;
            _userManager = userManager;
        }

        public Template[] Templates { get; set; }

        [BindProperty]
        public InputModel Input { get; set; }

        public class InputModel
        {
            public int TemplateId { get; set; }
            [Required(ErrorMessage = "Поле обязательно для
заполнения.")]
            [Display(Name = "Название")]
            public string Title { get; set; }
            [Display(Name = "Описание")]
            public string Desctiption { get; set; }
            [Required]
            [Display(Name = "Исполнители этапов")]
            public StageModel[] Stages { get; set; }
        }

        public class StageModel
        {
            [Required]
            public string StageName { get; set; }
```

```
                [Required(ErrorMessage = "Выберите шаблон для
заполнения.")]
                public int? UserId { get; set; }
            }

            public async Task<IActionResult> OnGetAsync(int?
templateId)
            {
                if (templateId == null)
                {
                    Templates = await
_context.Templates.ToArrayAsync();
                    return Page();
                }
                else
                {
                    var template = await
_context.Templates.FirstOrDefaultAsync(t => t.Id == templateId);
                    if (template == null)
                        return NotFound();
                    Xml.WorkflowTemplate tpl =
_workflowService.DeserializeXmlTemplate(template.Content);
                    Input = new InputModel();
                    Input.TemplateId = template.Id;
                    Input.Stages = tpl.Stages.Select(s => new
StageModel
                        { StageName = s.Title }).ToArray();
                }
                return Page();
            }

            public async Task<IActionResult> OnPostAsync()
            {
                if (!ModelState.IsValid)
                {
                    return Page();
                }

                var template = await
_context.Templates.FirstOrDefaultAsync(t => t.Id ==
Input.TemplateId);
                if (template == null)
                    return NotFound();

                var user = await _userManager.GetUserAsync(User);
                if (user.Group == null ||
!user.Group.AssignmentsAllowed)
                    return RedirectToPage("./Index");

                Xml.WorkflowTemplate xmlTemplate =
_workflowService.DeserializeXmlTemplate(template.Content);
                var now = DateTime.Now;
```

```
                var workflow = new Workflow { CreatedById =
user.Id, CreatedDate = now, Group = user.Group, Template = template,
Title = Input.Title, Description = Input.Desctiption };
                using (var tran =
_context.Database.BeginTransaction())
                {
                    _context.Workflows.Add(workflow);
                    await _context.SaveChangesAsync();
                    var docTypes = await
_context.DocumentTypes.ToArrayAsync();
                    var docs = new List<Document>();
                    foreach (var doc in xmlTemplate.Documents)
                    {
                        var docType = docTypes.First(t => t.Name ==
doc.Type);
                        var document = new Document { Workflow =
workflow, DocId = doc.Id, Title = doc.Title, DocumentType = docType,
IsRequired = doc.IsRequired };
                        docs.Add(document);
                    }
                    _context.Documents.AddRange(docs);
                    Stage firstStage = null;
                    for (int i = 0; i < xmlTemplate.Stages.Length;
i++)
                    {
                        var st = xmlTemplate.Stages[i];
                        var assignedStage = Input.Stages.First(s =>
s.StageName == st.Title);
                        var assignee = await
_context.Users.FirstAsync(u => u.Id == assignedStage.UserId);
                        var stage = new Stage { Workflow =
workflow, Name = st.Title, DurationInDays = st.Duration, CanReject =
st.CanReject, Assignee = assignee };
                        _context.Stages.Add(stage);
                        if (i == 0)
                        {
                            stage.StartedDate = now;
                            firstStage = stage;
                        }
                        if (st.InputDocuments != null)
                        {
                            foreach (var inDoc in
st.InputDocuments)
                            {
                                var refDoc = docs.First(d =>
d.DocId == inDoc.Id);
                                var doc = new InputDocument {
Document = refDoc, Stage = stage };
                                _context.InputDocuments.Add(doc);
                            }
                        }
                        if (st.OutputDocuments != null)
```

```
                        {
                            foreach (var outDoc in
st.OutputDocuments)
                            {
                                var refDoc = docs.First(d =>
d.DocId == outDoc.Id);
                                var doc = new OutputDocument {
Document = refDoc, Stage = stage };
                                _context.OutputDocuments.Add(doc);
                            }
                        }
                    }
                    try
                    {
                        await _context.SaveChangesAsync();
                        workflow.CurrentStage = firstStage;
                        await _context.SaveChangesAsync();
                        tran.Commit();
                    }
                    catch (Exception)
                    {
                        tran.Rollback();
                        throw;
                    }
                }

                return RedirectToPage("./Index");
            }
        }
    }
```