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

РАБОТА ПРОВЕРЕНА
Рецензент,
_____
«____»_____20___г.

ДОПУСТИТЬ К ЗАЩИТЕ
Заведующий кафедрой, д.ф.-м.н.,
доцент
_____/А.А.Замышляева
«____»_____2017 г.

Разработка информационной системы «Выпускники кафедры»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ
ЮУрГУ–09.03.04.2018.104.ПЗ ВКР

Консультант, (должность)
_____(И.О.Ф.)
«____»_____2018 г.

Руководитель работы, доцент
_____/М.Ю.Сартасова
«____»_____2018 г.
Автор работы
Студент группы ММиКН-414
_____/ Ю.В. Соколова
«____»_____2018 г.
Нормоконтролер, доцент
_____/Т.Ю.Оленчикова
«____»_____2018 г.

Челябинск 2018

АННОТАЦИЯ

Соколова Ю.В. Разработка информационной системы «Выпускники кафедры ПМиП».– Челябинск: ЮУрГУ, ЕТ-414, 67 с., 61 ил., 12 табл., библиогр. список – 26 наим., 2 прил.

Данная работа посвящена разработке информационной системы для выпускников кафедры прикладной математики и программирования(ПМиП). Информационная система реализована как веб-ресурс.

В работе выполнен обзор существующих альтернативных ресурсов, анализ потребностей пользователей, а также определен необходимый функционал сайта. Спроектирована и разработана архитектура системы, включающая в себя диаграмму размещения и диаграмму компонентов. Создан макет сайта, спроектирована база данных для хранения информации о выпускниках.

Визуальная структура сайта была реализована с помощью HTML, CSS и JavaScript. Серверная часть была написана на PHP с использованием фреймворка Laravel. Используемая СУБД - MySQL. В приложениях приведены руководство пользователя и текст программы.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

В современном мире, в эпоху инноваций и стремительного прогресса, высшее образование приобретает для людей всё большую значимость. Более того, и работодатели и студенты понимают, что важен не сам факт наличия диплома, а качество образования: багаж полученных знаний и навыков и умение их применять на практике. Так как же судить о качестве образования, предоставляемого тем или другим университетом?

В первую очередь, лицом высшего учебного заведения являются его выпускники. Любые их достижения, особенно в профессиональной деятельности, в немалой степени являются заслугой университета. Ведь главная цель высшего учебного заведения – подготовить будущих профессионалов своего дела.

Конечно, в ряду выпускников ЮУрГУ в целом и нашей кафедры в частности множество успешных в различных сферах людей: политики, директора предприятий, профессора, топ-менеджеры, ведущие специалисты. Информацию о некоторых из них можно найти на сайте нашего университета (www.susu.ru) или сайте кафедры (prm.susu.ru).

Наша кафедра каждый год выпускает около 50-ти бакалавров и магистров, но информация о дальнейшей их деятельности известна лишь в единичных случаях. Большинство выпускников навсегда теряют связь с университетом и кафедрой.

Интересно ли кому-то знать, как сложилась профессиональная карьера у бывших студентов? Ответ однозначен: да! Абитуриентам, студентам, преподавателям эта информация нужна и важна. Она позволит судить об успешности тех или иных специальностей, о качестве образования, об области применения получаемых на кафедре знаний. Не говоря уже о самих выпускниках, которые смогли бы узнать о судьбе своих товарищей по учёбе, восстановить утерянные с ними контакты.

На фоне всего изложенного выше возникает необходимость в некой единой системе, способной хранить информацию о всех выпускниках (во всяком случае о тех из них, кто был согласен предоставить информацию о себе) и доступной каждому заинтересованному. Такая система смогла бы обеспечить постоянную связь между кафедрой и выпускниками.

Разработка именно такой системы «Выпускники кафедры ПМиП» и стала моей задачей в рамках данного дипломного проекта. Естественно, очень важно было выбрать подходящие средства реализации (подробнее выбор платформ описан в разделе 1), учесть все потребности будущих пользователей и сделать систему по-настоящему удобной и приятной в использовании. Приступая к работе, я всерьёз рассматривала перспективу дальнейшей доработки и расширения системы до масштабов всего университета. Однако, такая перспектива станет возможной только в случае успешного использования моего проекта на практике. Именно поэтому основной целью, которую я преследовала на всех этапах проектирования и разработки стала практическая применимость информационной системы.

1 АНАЛИЗ ТРЕБОВАНИЙ. ОБЗОР СУЩЕСТВУЮЩИХ МЕТОДОВ РЕШЕНИЙ

1.1 Анализ потребностей пользователя

Целью дипломного проекта является разработка информационной системы «Выпускники кафедры ПМиП». Целевой аудиторией нашей системы в первую очередь являются выпускники, а также студенты, абитуриенты и преподаватели кафедры.

Система должна обеспечивать следующие потребности пользователей:

- создание единого информационного пространства. Такое пространство позволит бывшим студентам не терять связь с кафедрой и университетом.
- поддержание связей между выпускниками. При помощи нашей системы любой выпускник сможет найти своего одногруппника или товарища с потока, узнать что-то о его карьере, связаться с ним по оставленным в системе контактам.
- лёгкий и быстрый сбор информации о выпускниках. Процесс накопления информации в нашей системе должен быть предельно простым и удобным для пользователя, чтобы выпускник, пожелав присоединиться к такой системе, не столкнулся с какими-либо трудностями.
- отслеживание статистики. Исходя из данных о выпускниках, собранных в единой системе, станет возможным отслеживать некоторые интересные факты. Например, какой процент выпускников работают по профессии, сколько выпускников работают в другом городе или даже в другой стране, многие ли продолжили обучение и получили научные степени.
- возможность судить о качестве образования, получаемого на кафедре. Судить о качестве образования можно в первую очередь по успешности выпускников тех или иных специальностей. Тут-то нам и пригодится статистика, отслеживание которой так же является потребностью пользователя нашей системы.

Ещё одной важной потребностью пользователей становится безопасность, ведь информационной системе пользователи доверяют свои личные данные. Размещение данных пользователей в этой системе должно происходить только при получении от них согласия.

1.2 Обзор существующих решений

Alumni.spbu.ru – портал ассоциации выпускников Санкт-Петербургского государственного университета (см. рисунок 1.1).

На главную страницу сайта вынесены некоторые новости, информация о проектах, мероприятиях. Есть раздел «Знаменитые выпускники».

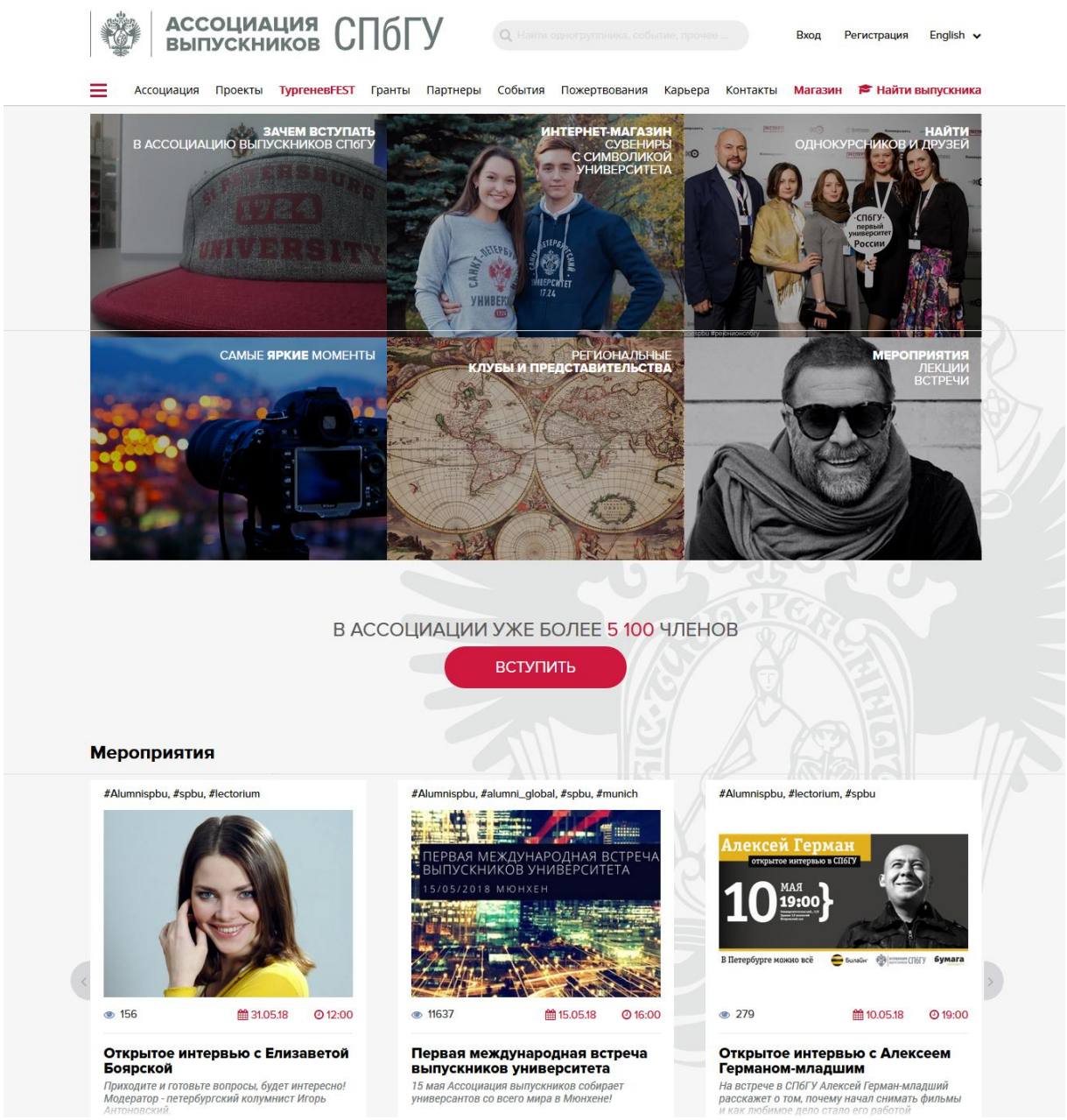

Рисунок 1.1 – Главная страница сайта alumni.spbu.ru

При нажатии на кнопку «Вступить», также расположенную на главной странице, открывается форма регистрации (см. рисунок 1.2). После ввода своего почтового адреса, пароля и проставления отметки о согласии на использование своих личных данных, предлагается выбрать тип регистрации (см. рисунок 1.3). Поддерживаются несколько типов регистрации: гость, студент, выпускник, партнёр, член ассоциации.

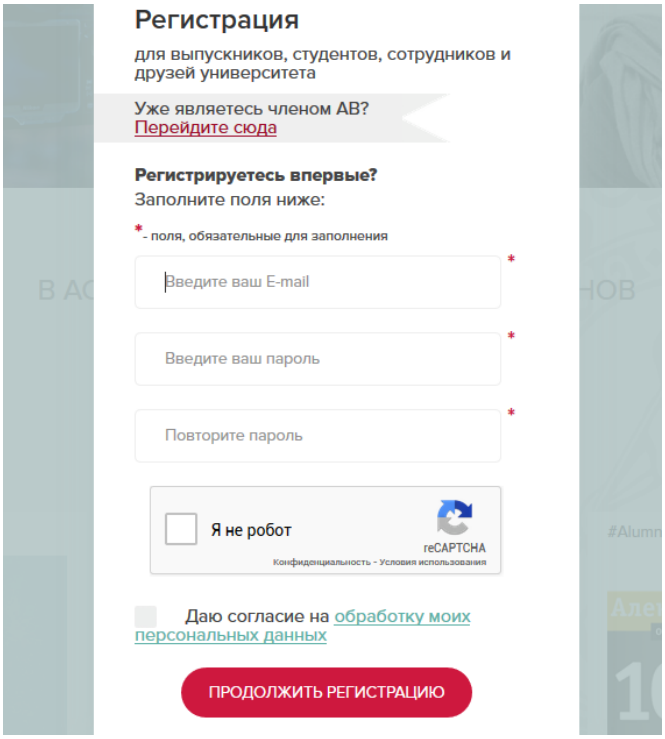

Рисунок 1.2 – Форма регистрации сайта alumni.spbu.ru

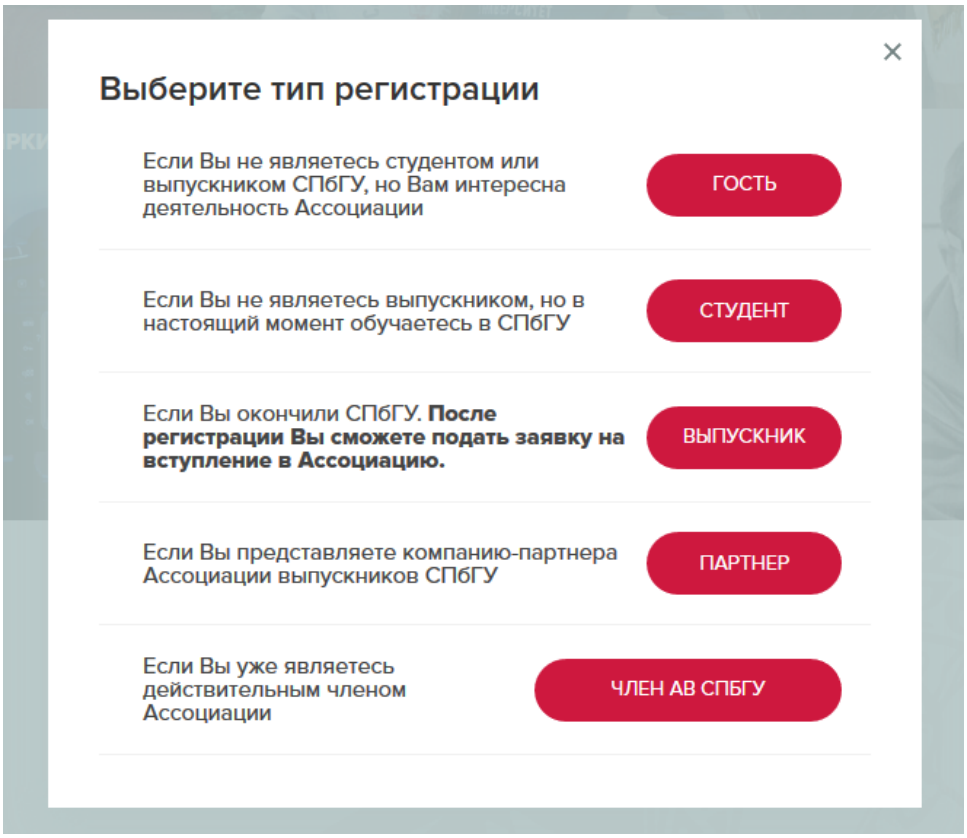

Рисунок 1.3 – Выбор типа регистрации

При выборе типа «выпускник» открывается анкета выпускника (см. рисунок 1.4). В анкете необходимо указать фамилию (причём как действующую, так и ту,

которая была у выпускника на момент обучения в университете, на случай если фамилия была изменена), имя, отчество, номер телефона. Обязательными для заполнения являются поля «факультет», «учёная степень», «специальность», «год выпуска». По желанию можно загрузить фотографию для своего профиля и указать номер группы.

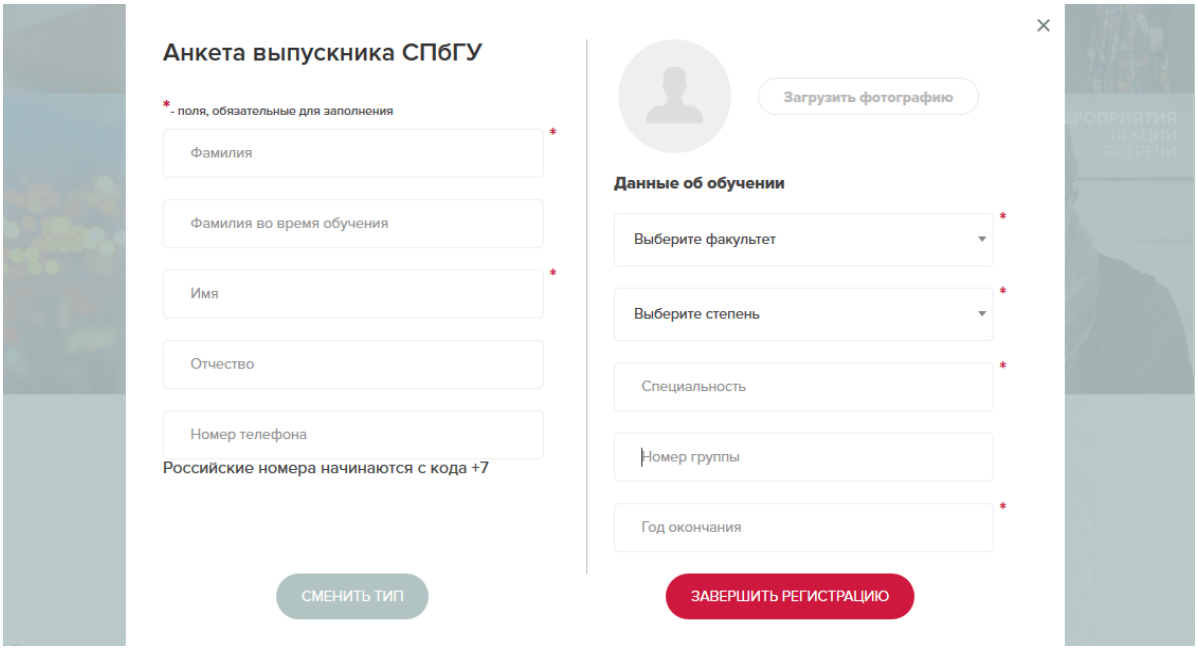

Рисунок 1.4 – Анкета выпускника

При выборе других типов регистрации открываются соответствующие этим типам анкеты. Например, выбрав «студент», появляется анкета студента, в которой необходимо указать фамилию, имя, отчество, номер телефона, факультет и год окончания.

После заполнения анкеты и нажатия кнопки «завершить регистрацию» на сайте создаётся личный профиль (по умолчанию тип профиля «гость»). Во вкладке «редактировать профиль» можно всегда сменить текущий тип регистрации, а также указать дополнительную информацию о себе, например, место работы, дополнительное образование, ссылки на свои профили в социальных сетях и т.п. (см. рисунок 1.5).

Во вкладке «Ассоциация» располагается вся информация об ассоциации: направление деятельности, органы управления, клубы и представительства, отчётность и реквизиты (см. рисунок 1.6).

На сайте поддерживается функция поиска выпускников. При нажатии на пункт меню «Найти выпускника» открывается форма поиска (см. рисунок 1.7). Искать выпускников можно по имени, фамилии; по году поступления, году выпуска; по месту проживанию; по факультету; по наименованию организации, в которой выпускник работает. После нажатия кнопки «Найти» под формой отображаются удовлетворяющие введённым данным профили (причём профили всех типов, в том числе студентов, гостей, партнёров).

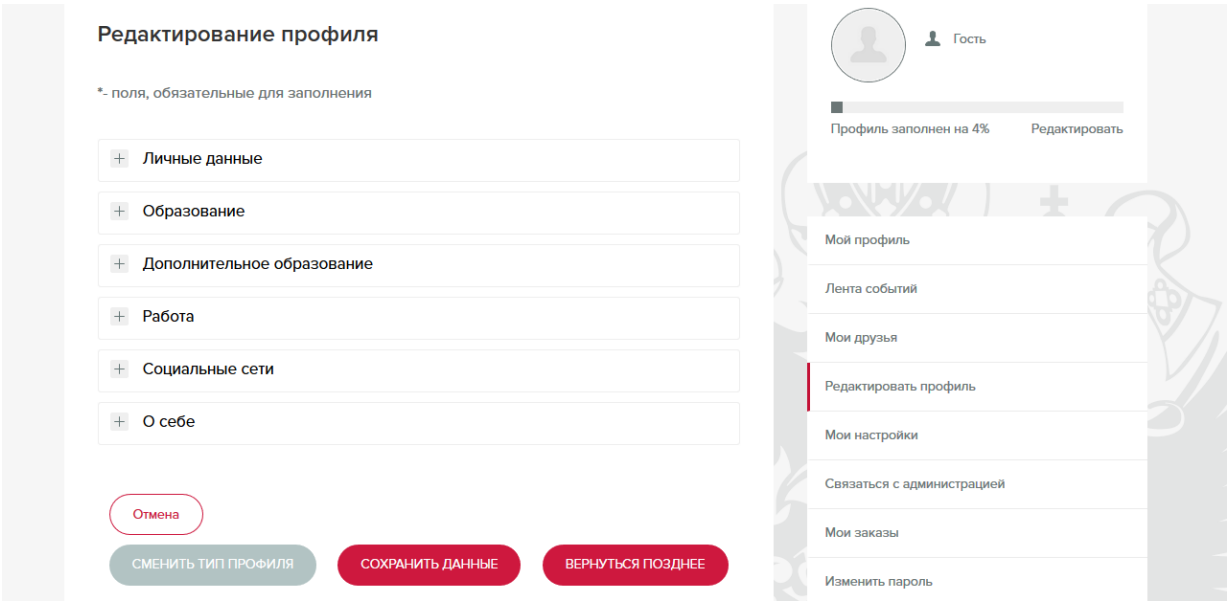

Рисунок 1.5 – Редактирование профиля

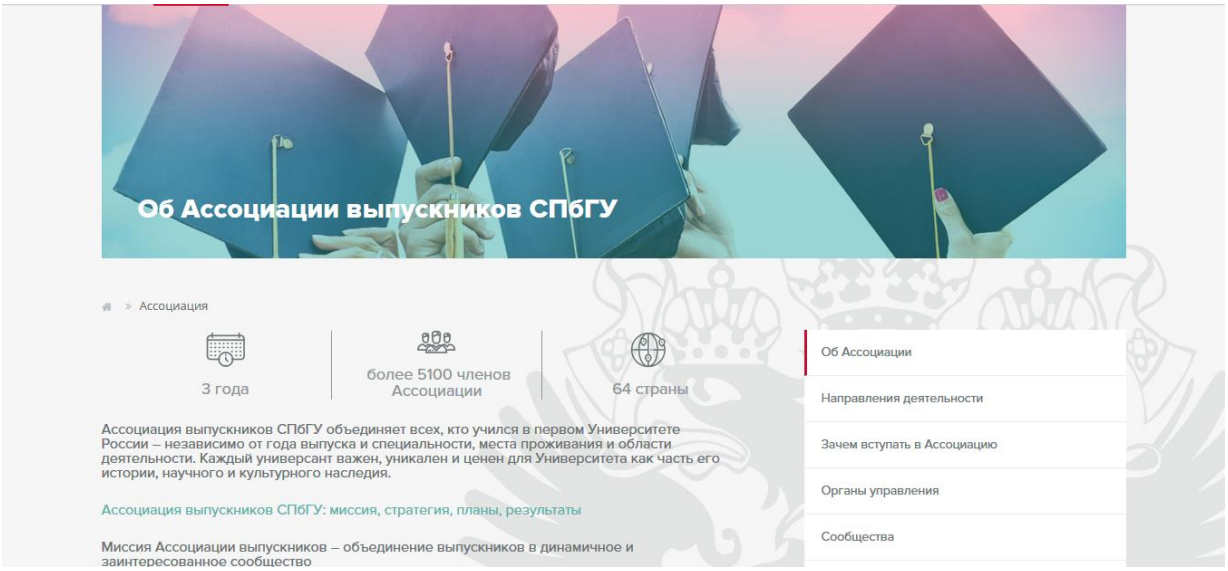

Рисунок 1.6 – Вкладка «Ассоциация»

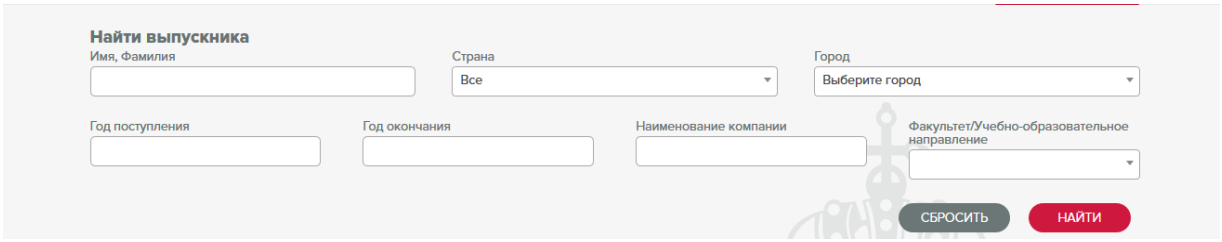

Рисунок 1.7 – Форма поиска выпускников

В целом сайт удобен в использовании и является отличным ресурсом для сбора информации о выпускниках, для поиска бывших одногруппников и товарищей по учёбе, для восстановления утерянных контактов. Сайт помогает

взаимодействовать студентам, выпускникам и друзьям университета. Этот ресурс может служить примером для системы, которую предстоит создать нам. Однако, недостатком портала является отсутствие какой-либо статистики о выпускниках. Невозможно понять, например, многие ли из них работают по специальности, в каких странах и городах проживают и т.д.

Graduates.unn.ru – портал ассоциации выпускников университета Лобачевского (см. рисунок 1.9).

На главной странице располагается основная информация об ассоциации, некоторые новости, информация о событиях, мероприятиях.

Также есть раздел с выдающимися выпускниками университета (см. рисунок 1.8). При наведении на фото выпускника можно увидеть его фамилию, имя и отчество. При клике на фото, невозможно узнать какую-либо более подробную информацию. Наверняка, это связвно с тем, что сайт функционирует совсем недавно и пока находится в стадии разработки.

На главной странице находится интерактивная карта (см. рисунок 1.10). Согласно пояснению, на карте отмечены страны и города, в которых живут и работают выпускники, но никакой более подробной информации при наведении на отметку не отображается, что очевидно является недоработкой.

На сайте не поддерживается регистрация пользователей, поэтому информация о выпускниках может появляться там только, если её размещает администратор или владелец сайта. Оставить информацию о себе самостоятельно, просто войдя на сайт - невозможно, поэтому данный ресурс не удовлетворяет потребность в лёгком и удобном сборе информации о выпускниках.

В целом ресурс явно находится на стадии разработки, есть некоторые проблемы с пользовательским интерфейсом (например, меню в шапке сайта почти сливается с фоном, а огромные сменяющиеся фото в самом верху главной страницы выглядят не эстетично и мешают ориентироваться), не работают некоторые кнопки и ссылки.

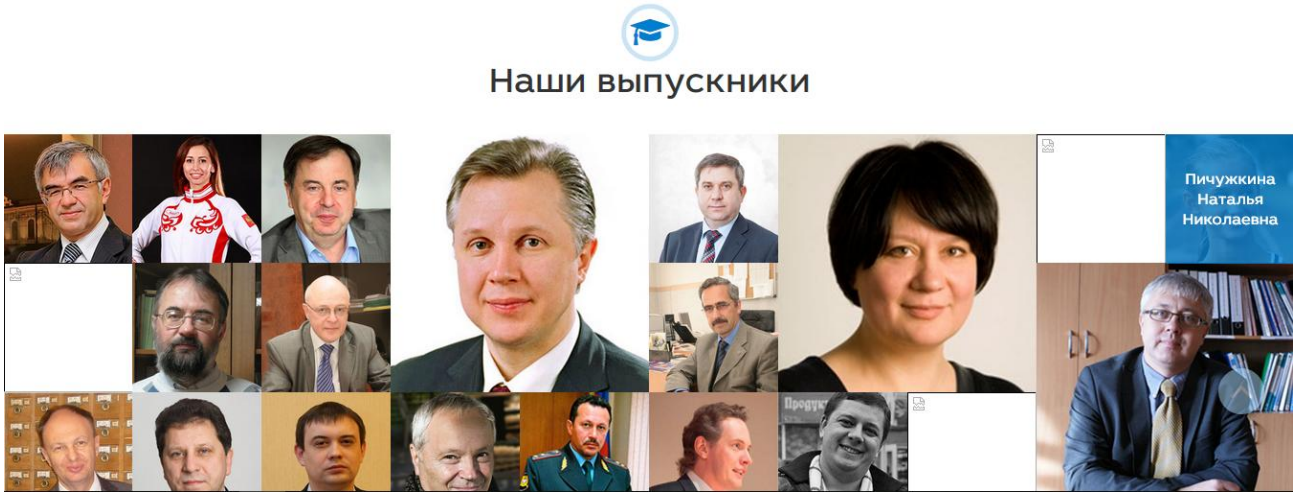

Рисунок 1.8 – Выдающиеся выпускники

Рисунок 1.9 – Главная страница сайта graduates.unn.ru

Преимуществом данного сайта является наличие интерактивной карты. После доработки её функционала, она может стать отличным наглядным отображением статистики.

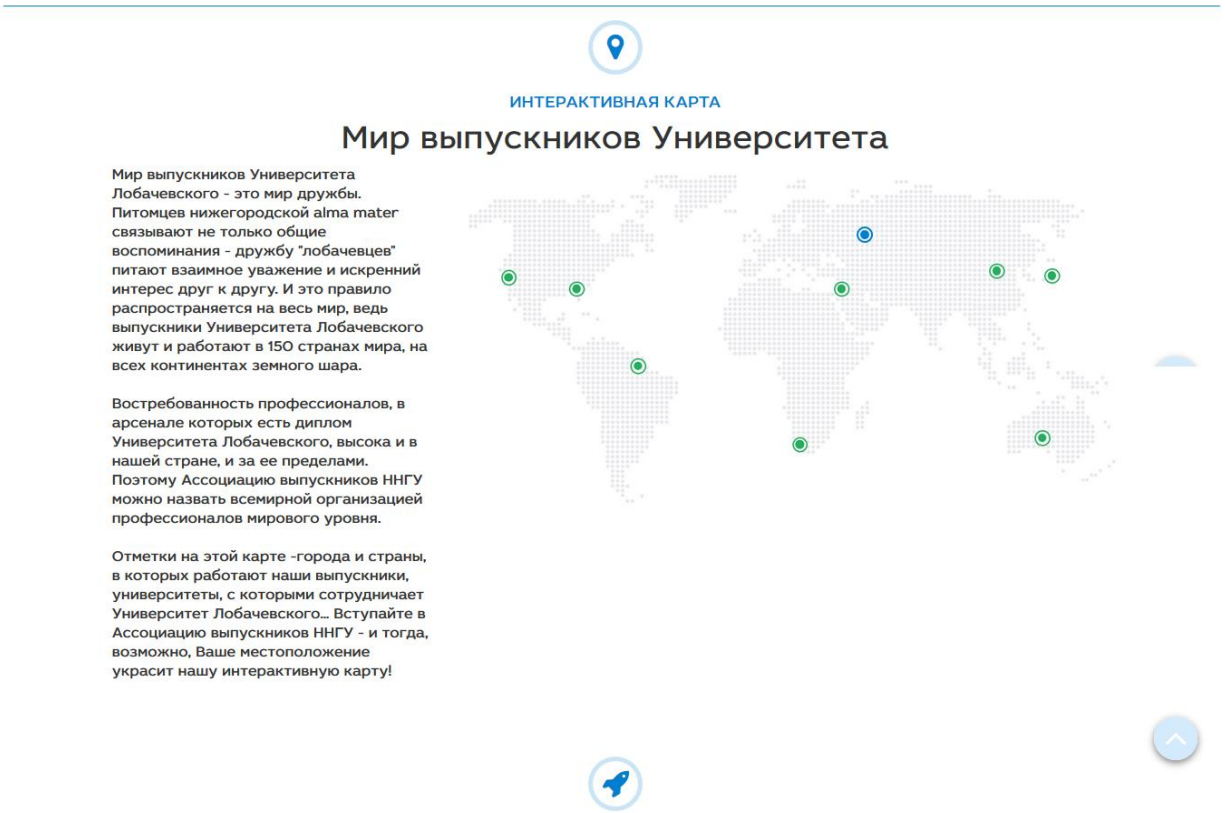

Рисунок 1.8 – Интерактивная карта

В основном, университеты не имеют отдельного ресурса, посвящённого выпускникам, но на сайтах высших учебных заведений часто встречается раздел «Выпускники» или «Выпускникам». Эти разделы также можно рассматривать, как альтернативы нашей будущей информационной системы.

Www.msu.ru/alumni – вкладка «Выпускники» сайта Московского Государственного Университета имени Ломоносова (см. рисунок 1.11). На этой странице расположены ссылки на клубы и сообщества выпускников по факультетам. Кроме того, есть ссылка «списки выпускников МГУ, давших согласие на обработку персональных данных». Перейдя по этой ссылке, выбрав год и факультет, можно увидеть список выпускников в формате PDF. Например, выбрав 2016 год и «Факультет вычислительной математики и кибернетики» откроется документ, изображённый на рисунке 1.12.

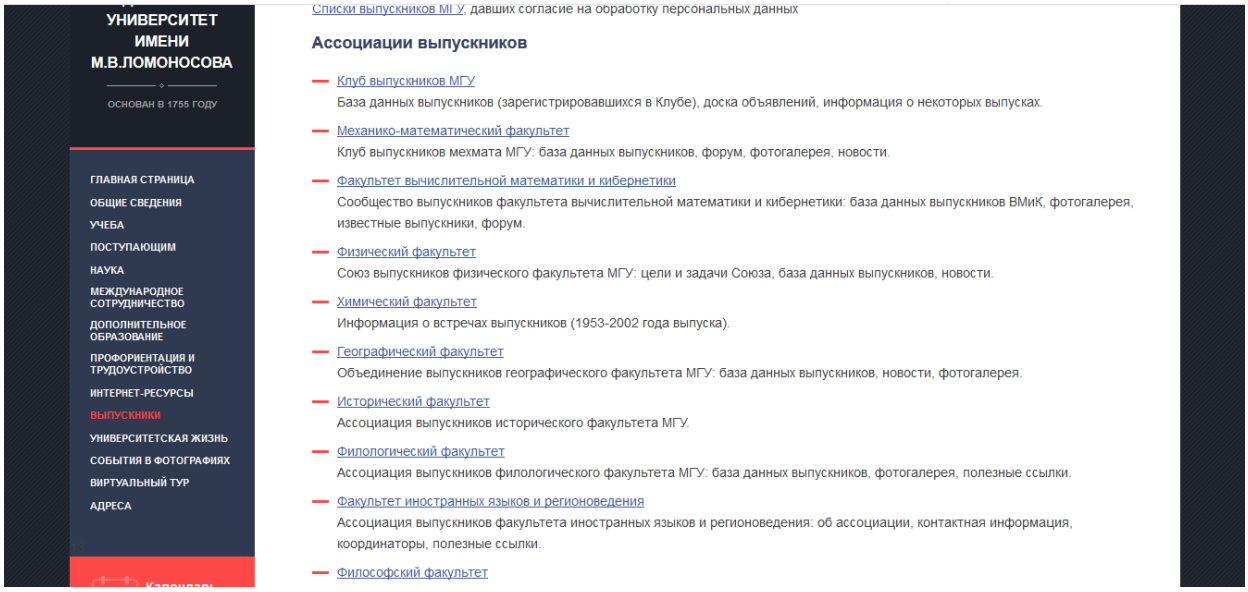

Рисунок 1.9 – Вкладка «Выпускники» сайта www.msu.ru

Список выпускников, давших согласие на обработку персональных данных

Уровень образования	Направление подготовки/специальность (код)	Фамилия	Имя	Отчество
Бакалавриат	Прикладная математика и информатика	Абдрашитова	Айгуль	Рашитовна
Бакалавриат	Фундаментальные информатика и информационные технологии	Абрамов	Денис	Михайлович
Бакалавриат	Прикладная математика и информатика	Агапов	Павел	Андреевич
Бакалавриат	Прикладная математика и информатика	Агуленко	Александр	Евгеньевич
Бакалавриат	Прикладная математика и информатика	Аксёнов	Кирилл	Александрович
Бакалавриат	Прикладная математика и информатика	Анисимов	Иван	Александрович
Бакалавриат	Прикладная математика и информатика	Анциферова	Анастасия	Всеволодовна
Бакалавриат	Прикладная математика и информатика	Ардаширов	Денис	Тимурович
Бакалавриат	Прикладная математика и информатика	Армяков	Иван	Дмитриевич
Бакалавриат	Прикладная математика и информатика	Архипенко	Константин	Владимирович
Бакалавриат	Прикладная математика и информатика	Асирян	Александр	Камоевич
Бакалавриат	Прикладная математика и информатика	Афанасьев	Никита	Александрович
Бакалавриат	Прикладная математика и информатика	Ахметзянова	Лилия	Руслановна
Бакалавриат	Прикладная математика и информатика	Бабуева	Александра	Алексеевна
Бакалавриат	Прикладная математика и информатика	Баринова	Ольга	Александровна
Бакалавриат	Прикладная математика и информатика	Беззубиков	Александр	Александрович
Бакалавриат	Прикладная математика и информатика	Белова	Вера	Андреевна
Бакалавриат	Прикладная математика и информатика	Беляев	Андрей	Юрьевич
Бакалавриат	Прикладная математика и информатика	Беляев	Михаил	Владимирович
Бакалавриат	Прикладная математика и информатика	Бикмуратов	Фархад	Мансурович
Бакалавриат	Прикладная математика и информатика	Борисов	Артем	Ильшатович
Бакалавриат	Прикладная математика и информатика	Брагин	Юрий	Олегович
Бакалавриат	Прикладная математика и информатика	Булгакова	Наталья	Алексеевна

Рисунок 1.10 – Список выпускников

Но, кроме имени и специальности, никакой информации получить невозможно. Похожая ситуация наблюдается и на сайтах многих других высших учебных заведений.

Практически все сайты университетов (либо факультетов, кафедр) оснащены разделом «Выпускники», в котором есть информация о нескольких самых выдающихся выпускниках. Этой информации крайне мало, чтобы составить какое-то мнение об успешности выпускников в целом и о качестве образования

университета в частности. К тому же, пользователь не сможет оставить информацию о себе самостоятельно, если пожелает.

Итак, из рассмотренных нами web-ресурсов наиболее полным и функциональным, способным удовлетворить потребности пользователя, схожие с выявленными нами ранее в подразделе 1.1, является сайт Ассоциации выпускников СПбГУ (alumni.spbu.ru).

1.3 Необходимые функции и другие требования к приложению

Исходя из потребностей пользователей и анализа существующих аналогичных систем, были сформулированы функции, поддержка которых должна быть обеспечена в нашем приложении:

- авторизация;
- создание, редактирование, удаление профилей пользователей;
- интерактивная карта.

1.4 Определение типа информационной системы

Необходимо разобраться с понятием «информационная система». Информационная система (ИС) – «система, организующая обработку информации о предметной области и её хранение» [1]. Также ИС должна предоставлять функцию поиска информации. Согласно ГОСТу №33707-2016 поиском информации называют «процесс нахождения и выбора(выдачи) требуемой (т.е. определённой заранее заданными признаками) информации» [1]. Понятие ИС включает в себя организационные ресурсы (человеческие, технические, финансовые и т. д.), которые обеспечивают и распространяют информацию. ИС предназначена для своевременного предоставления определённым людям определённой информации. «Подавляющее большинство информационных систем работает в режиме диалога с пользователем» [2].

По масштабу ИС делятся на одиночные, групповые и корпоративные [2] (см. рисунок 1.13).

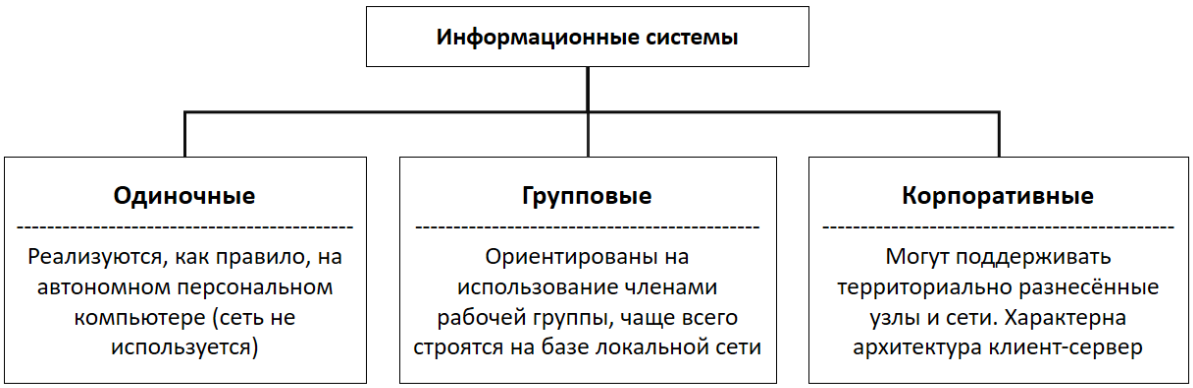

Рисунок 1.11 – Классификация ИС по масштабу

По характеру обработки данных ИС делятся на:

- информационно-справочные (информационно-поисковые) ИС. В таких системах нет сложных алгоритмов обработки данных, их целью является поиск и выдача информации в удобном виде;
- ИС обработки данных (решающие ИС). Данные в них обрабатываются по сложным алгоритмам.

Ещё один критерий классификации – используемая в основе архитектура (см. рисунок 1.14):

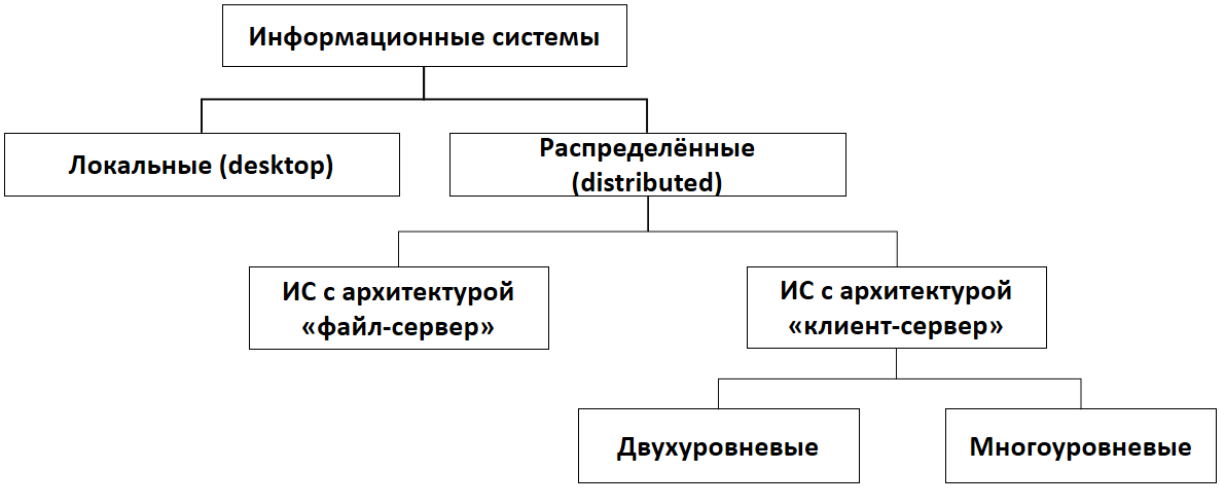

Рисунок 1.12 – Классификация ИС по архитектуре

Локальные ИС – все компоненты находятся на одном компьютере.

Файл-серверные – на сервере данные лишь извлекаются из файлов, вся система управления базами данных(СУБД) и клиентские приложения располагаются на стороне пользователя.

Клиент-серверные – база данных(БД) и СУБД располагаются на сервере, на стороне пользователя лишь клиентские приложения. Изначально клиент-серверные системы были двухуровневыми: «клиент нёс ответственность за отображение пользовательского интерфейса и выполнения кода приложения, а роль сервера обычно поручалась СУБД. Проблемы возникли с усложнением логики предметной области – бизнес-правил, алгоритмов вычислений, условий проверок и т.д. Прежде все эти обязанности возлагались на код клиента и находили отражение в содержимом интерфейсных экранов. Чем сложнее становилась логика, тем более неуклюжим и трудным для восприятия делался код» [3]. Всё это привело к появлению многоуровневой системной архитектуры, в рамках которой пользовательские клиентские приложения не обращаются к СУБД напрямую, они взаимодействуют с промежуточными уровнями.

Учитывая теоретические сведения, изложенные выше, информационную систему «Выпускники кафедры ПМиП» будем создавать как корпоративную,

информационно-справочную систему с многоуровневой клиент-серверной архитектурой.

1.5 Выбор шаблона проектирования

Основным требованием к системе является лёгкость её использования и доступность. Наиболее удобным решением, удовлетворяющим типу нашей системы, является Web-приложение. В таком приложении используется трёхуровневая (трёхслойная) архитектура [3]:

Таблица 1.1

Слой	Функции
Представление	Предоставление услуг, отображение данных, обработка событий пользовательского интерфейса, обслуживание запросов HTTP, поддержка функций командной строки и API пакетного выполнения
Домен	Бизнес-логика приложения
Источник данных	Обращение к базе данных, обмен сообщениями, управление транзакциями т.д.

Создавать сайт решено было с использованием паттерна трёхуровневой архитектуры MVC [4]. Концепция паттерна(шаблона) MVC (model – view - controller) подразумевает разделение данных приложения, пользовательского интерфейса и управляющей логики на три отдельных компонента (см. рисунок 1.15):

- модель;
- представление;
- контроллер.

Модель (Model) представляет данные предметной области представлению и реагирует на команды контроллера, изменяя своё состояние.

Представление (View) отвечает за отображение данных предметной области (модели) пользователю, реагируя на изменения модели. В случае Web-приложения, представление – это та HTML страница, которую видит пользователь.

Контроллер (Controller) интерпретирует действия пользователя, оповещая модель о необходимости изменений [5, 6].

Реализовывать эти компоненты удобно, используя классы ООП (Объектно-ориентированного программирования).

Основное преимущество этого паттерна – независимость компонентов. Они могут модифицироваться, удаляться, заменяться, не затрагивая друг друга. Вне зависимости от изменений система в целом будет оставаться рабочей.

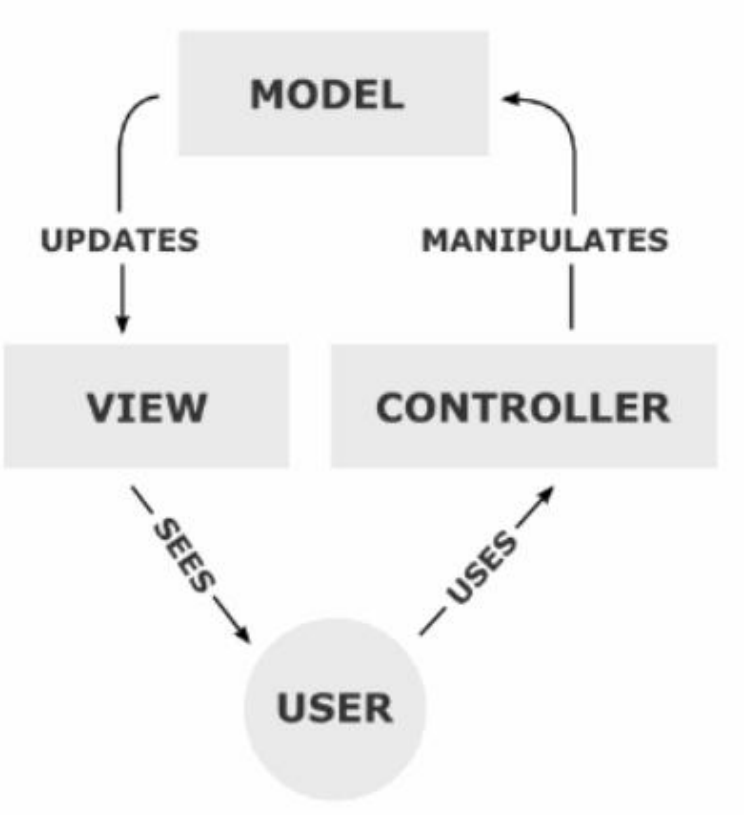

Рисунок 1.13 – Схема взаимодействия компонентов MVC

Благодаря этому, приложение приобретает важнейшие в условиях стремительно развивающихся технологий свойства – масштабируемость и гибкость. Например, появляется необходимость в новых, ранее не используемых, данных предметной области – тогда мы просто добавляем новую модель, описывающую их, контроллер, передающий запросы на отображение этих данных и, возможно, добавляем в вид элемент, эти данные предоставляющий пользователю. И вот новые данные уже в нашем приложении, при чём вся прежняя структура и функции не нарушены, они работают, как и раньше.

Большинство программистов специализируются в определённой узкой области: либо занимаются разработкой графических пользовательских интерфейсов, либо разрабатывают бизнес-логику программы. Используя подход MVC можно добиться разделения труда и тем самым увеличить эффективность и скорость разработки.

1.6 Выбор языков программирования

1.6.1 Язык серверных сценариев

Языком программирования серверной части приложения был выбран PHP 7.2 [7]. В первую очередь из-за колоссальной популярности в сообществе программистов и открытости исходных кодов.

Для характеристики языков программирования существует индекс TIOBE (TIOBE programming community index) [8]. Он является индикатором популярности языка, обновляется раз в месяц. Рейтинги основаны на количестве квалифицированных инженеров по всему миру, курсов и сторонних поставщиков. Для расчета рейтингов используются популярные поисковые системы, такие как Google, Bing, Yahoo !, Wikipedia, Amazon, YouTube и Baidu.

Значения индекса на май 2018 года приведены в таблице (см. рисунок 1.16). PHP является одним из лидеров среди языков, применяющихся для создания динамических веб-сайтов.

May 2018	May 2017	Change	Programming Language	Ratings	Change
1	1		Java	16.380%	+1.74%
2	2		C	14.000%	+7.00%
3	3		C++	7.668%	+2.92%
4	4		Python	5.192%	+1.64%
5	5		C#	4.402%	+0.95%
6	6		Visual Basic .NET	4.124%	+0.73%
7	9	^	PHP	3.321%	+0.63%
8	7	v	JavaScript	2.923%	-0.15%
9	-	^^	SQL	1.987%	+1.99%
10	11	^	Ruby	1.182%	-1.25%

Рисунок 1. 14 – Рейтинг TIOBE на май 2018 года

PHP («PHP: Hypertext Preprocessor») – широко используемый язык сценариев общего назначения с открытым исходным кодом, который особенно подходит для веб-разработки и может быть встроен в HTML. Его синтаксис опирается на C, Java и Perl и прост в освоении. Язык поддерживает концепцию объектно-ориентированного подхода (ООП). Основная цель языка – позволить веб-разработчикам быстро создавать динамические веб-страницы [9].

На данный момент 83.5% сайтов, язык платформы которых известен, используют PHP как основной язык (см. рисунок 1.17) [11].

Новая версия PHP 7 основывается на экспериментальной ветви PHP, которая изначально называлась phpng (PHP Next Generation – следующее поколение), и разрабатывалась с упором на увеличение производительности и уменьшение потребления памяти [6]. В новой версии добавлена возможность указывать тип возвращаемых из функции данных, добавлен контроль передаваемых типов для скалярных данных [10], а также новые операторы.

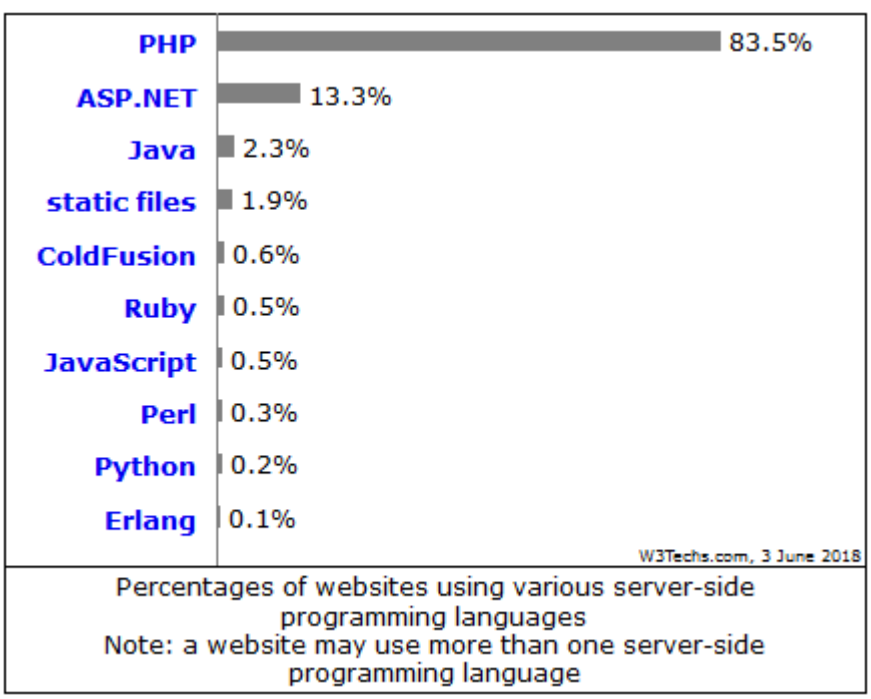

Рисунок 1.15 – Диаграмма использования языков программирования для веб-сайтов

PHP поддерживает доступ почти к любым системам управлениями базами данных, таким как: MySQL, SQLite, PostgreSQL, Oracle, Microsoft SQL и др. Именно с базой данных MySQL [12], нам и предстоит работать.

1.6.2 Язык пользовательского интерфейса

Для представления структуры страницы использовался HTML5 [13] – теговый язык разметки документов, представляющий собой набор элементов разметки. Отличие HTML5 от предыдущих версий языка заключается в введении новых тегов, определяющих более точную спецификацию элементов страницы. А для описания свойств элементов разметки использовался CSS3[14,15] – формальный язык описания внешнего вида документа, в частности CSS применяется для задания цвета заливки элементов, размеров, расположения на сайте и т.д. Основной целью разработки CSS является разделение описания структуры веб-страницы (HTML) от описания внешнего вида этой веб-страницы. Такое разделение увеличивает доступность документа, предоставляет большую гибкость и возможность управления его представлением, а также уменьшить сложность и повторяемость кода [16]. В версии CSS3 стало возможным создание анимированных элементов, введение теней, сглаживания и других эффектов элементов [14].

1.7 Выбор фреймворка

1.7.1 Зачем использовать фреймворк?

Несмотря на то, что PHP в чистом виде можно использовать для создания практически любого приложения, всё растущая функциональность сайтов требует структурирования и организации самого процесса разработки. Наиболее естественным решением этой задачи стали PHP-фреймворки.

Зачем использовать фреймворк вместо чистого PHP для разработки приложения? Несколько преимуществ использования фреймворков изложены ниже.

- PHP фреймворк существенно сокращает сроки разработки. Не нужно прописывать вручную подключения к базам данных, ajax-запросы и т.п.
- Изначально проект строго структурирован, поэтому вероятность допустить ошибки в архитектурных решениях крайне низка.
- Код хорошо документирован и пригоден для повторного использования.
- Фреймворки позволяют легко масштабировать, расширять проекты.
- Код становится лаконичнее и понятнее, что упрощает работу. Особенно актуально, если разработка ведётся командой программистов.
- Веб-приложения более безопасны, если написаны с использованием фреймворка. Ведь разработчику не приходится беспокоиться о низкоуровневой безопасности сайта.
- Обеспечивает следование шаблону. В нашем случае шаблону MVC (Model-View-Controller).
- Способствует применению парадигмы ООП (Объектно-ориентированного программирования).

1.7.2 Обзор популярных фреймворков: Laravel , Yii, Symfony.

Laravel.

Laravel (см. рисунок 1.18) [17] относительно молодой фреймворк, первый релиз появился только в 2011 году. Но по результатам опроса портала SitePoint [18] он является самым популярным среди веб-разработчиков.

Преимущества фреймворка.

- Огромная экосистема, выросшая за последние годы в 2 раза.
- Наличие простой и подробной документации, обучающих ресурсов, форумов и т.п.
- Наличие русскоязычного сообщества, где вся документация представлена на русском языке [18].
- Элегантный синтаксис, упрощающий такие задачи, как авторизация, управление кешированием, очередями, сессиями.
- Поддерживает Composer для управления пакетами.
- Поддержка собственного шаблонизатора Blade. В отличие от других шаблонных систем, Blade позволяет использовать PHP код в

представлениях(view), т.е. прямо внутри HTML-разметки. Код внутри HTML-файла превращается в чистый PHP на этапе обработки.

- Поддержка REST-full маршрутизации.
- Хорошо поддерживается модульное тестирование.
- Есть множество пакетов для расширения функциональности фреймворка.
- Многие хостинг компании предоставляют поддержку Laravel и предлагают хостинг решения для приложений на Laravel.

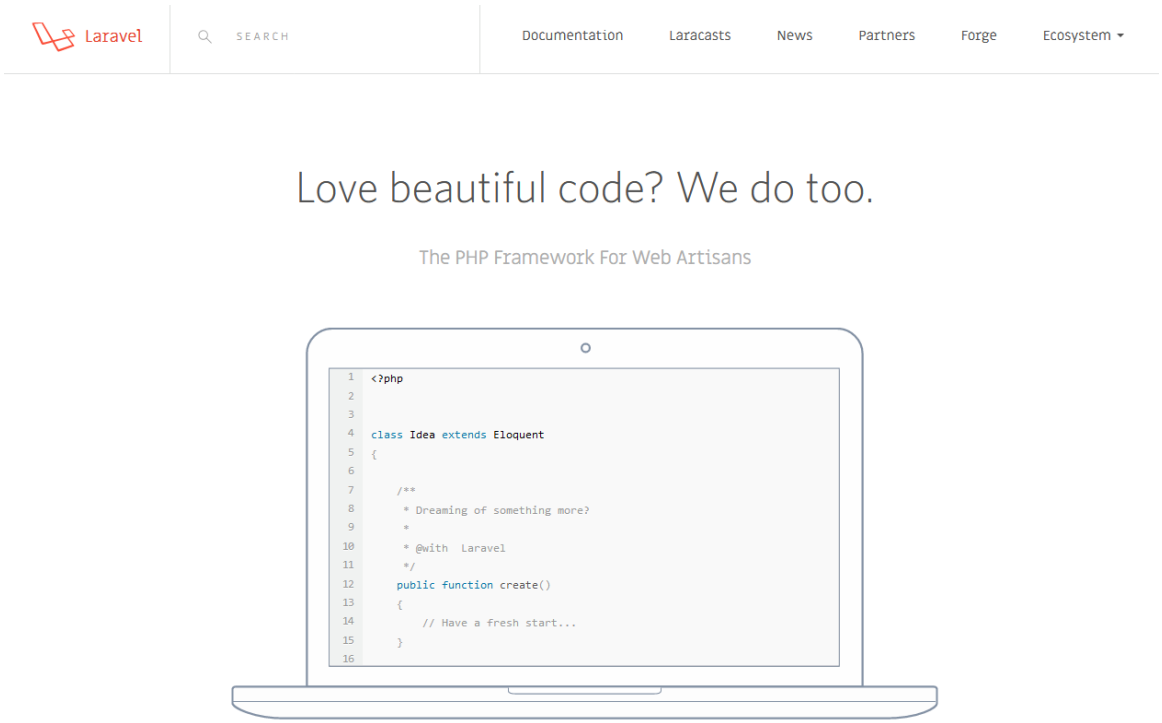

Рисунок 1.16 – Официальный сайт Laravel

Недостатки фреймворка.

- Laravel проигрывает другим фреймворкам в производительности (количество запросов, обрабатываемых фреймворком, в секунду). Сравнение производительности фреймворков для версии языка PHP 7.1 представлено на рисунке 1.19.
- Занимает много памяти по сравнению с другими фреймворками. (см. рисунок 1.20).

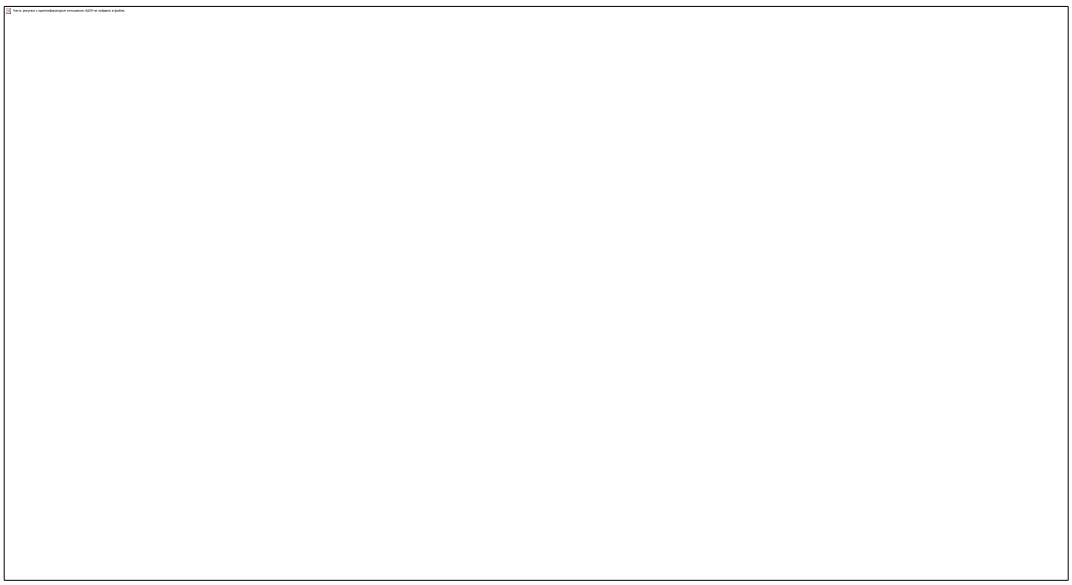

Рисунок 1.17 – Производительность фреймворков (запросов в секунду)

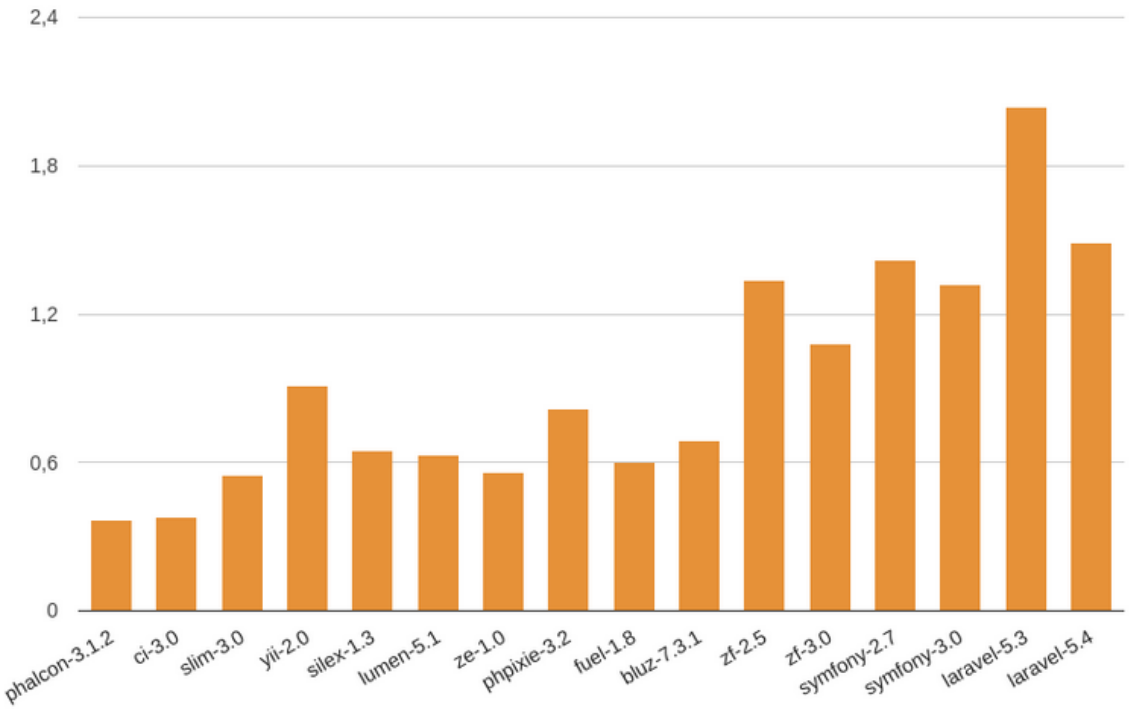

Рисунок 1.18 – Занимаемая фреймворками память (Мб)

Symfony.

Несмотря на то, что релиз третьей версии состоялся еще в 2015 году, именно вторая версия Symfony (см. рисунок 1.21) единолично удерживает 3-е место по популярности среди фреймворков. Причина здесь очевидна – скорость работы и общая простота. Но чтобы это не шло в разрез с функциональностью, пользователю предлагается выбрать одну из 3 версий для профильной работы.

- Standard Edition – для знакомства и выполнения общих задач. На ней основан дистрибутив Hello World Edition, который содержит ровно один скрипт оптимизации для дальнейшего использования в бенчмарках.
- Symfony CMF – адаптация для разработчиков, работающих с CMS-системами.
- REST Edition – оптимизация для работы с REST-архитектурой (интернет-магазины, поисковые системы и т.д.).

Стереотипно считается, что Symfony – это фреймворк для любителей командной строки. Действительно, встроенный интерфейс SensioGeneratorBundle поможет вам из одной строки текста получить целый скелет для вашего кода.

Преимущества.

- Высокая производительность благодаря кэшированию байт-кода.
- Надежность. В настоящее время является наиболее стабильным.
- Масса функций для расширения.
- Большое сообщество, с документацией, учебными ресурсами.
- Хорошая поддержка; полностью сформированный фреймворк.
- Система шаблонов Symfony Twig.

Рисунок 1.19 – Логотип фреймворка Symfony

Недостатки.

- Предоставляет неполную реализацию модели MVC, а только лишь модель и контроллер.
- Достаточно сложен в освоении.

Yii.

Yii (см. рисунок 1.22) является высокопроизводительным, безопасным и быстрым фреймворком для разработки приложений и web-сайтов. Yii использует менеджер зависимостей PHP для обработки различных зависимостей и установок.

Рисунок 1.20 – Логотип фреймворка Yii

Преимущества.

- Отлично подходит для разработки приложений в режиме реального времени
- Занимает минимальное количество памяти.
- Быстро справляется с оптимизацией для маршрутизации URL-адресов, кэш-моделями, кодом.
- Хорошо подходит для многоязычных приложений.
- Прост в установке.
- Легко настраивается для необходимых задач. Почти все компоненты фреймворка расширяемые.
- Система шаблонов Yii Default.
- Имеет большое сообщество с большим количеством учебных ресурсов.

Недостатки.

- Допускает повторения кода, в отличие от других MVC-фреймворков, в которых данная проблема решена.
- Сильная связанность классов. Все в системе наследуется от CComponent.
- Интеграция шаблонизатора (Twig, Smarty) довольно слабая, по сравнению с нативными шаблонами.

1.7.3 Выводы по обзору фреймворков

Сходства.

- Все три фреймворка являются full-stack PHP фреймворками, что позволяет создавать полностью готовые веб-приложения, начиная от интерфейса (front-end) и заканчивая серверными сценариями (back-end).
- Есть доступ к проектам с открытым исходным кодом. Их исходный код можно найти на GitHub.
- Фреймворки хорошо документированы и поддерживаются большими сообществами.
- Каждый из них поддерживает ORM (Object Relationship Mapping). ORM – это «технология программирования, которая связывает базы данных с концепциями объектно-ориентированных языков программирования, создавая виртуальную объектную базу данных» [20]. Например, в Laravel поддержку ORM обеспечивают «миграции».

- Все три фреймворка используют шаблонны для улучшения front-end разработки и технического обслуживания.
- Они являются надежными и безопасными для создания веб-приложений 2.0 [15].

Поддержка баз данных.

Каждый фреймворк оснащён поддержкой различных баз данных. Symfony 2 может работать с массивом баз данных, таким как NoSQL и DynamoDB. Поэтому в данном аспекте является более универсальным
Yii и Laravel также полезны, но обеспечивают меньшую совместимость.

Приведённая на рис. 1.23 таблица показывает какие базы данных поддерживаются тем или иным фреймворком.

Framework	Laravel	Yii	Symfony 2
Database	SQLite MySQL PostgreSQL Redis Microsoft BI MongoDB	MySQL SQLite Microsoft BI Oracle PostgreSQL MongoDB	Microsoft BI MongoDB MySQL NoSQL PostgreSQL CouchDB DynamoDB GemFire GraphDB MemBase MemCacheDB Oracle Apache Jackrabbit

Рисунок 1.21 – Совместимость фреймворков с базами данных

Быстрая разработка.

Для любой компании или клиента первостепенно важно как можно быстрее вывести приложение на рынок для удовлетворения потребительского спроса. Laravel быстро набирает обороты, закрепляясь в тройке основных PHP фреймворков. Кроме того, Laravel является идеальным для начинающих разработчиков. Он прост для понимания – легко найти документацию, обучающие уроки и видео, готовые программы для примера в интернете.

В свою очередь, Yii поднимает производительность на новый уровень.

Эффективность.

Производительность любого приложения важна, когда оно использует критически важные данные в режиме реального времени. Множество приложений обладают высокой производительностью и предъявляют высокие требования к скорости работы. Во многих проектах фреймворки играют решающую роль.

Несмотря на то, что Laravel является изначально самым медленным фреймворком, существует масса способов и ресурсов для ускорения Laravel приложений.

Это делает Laravel наиболее гибким, и позволяет использовать его в приложениях с динамической нагрузкой.

1.7.4 Фреймворк Laravel 5

Для реализации приложения был выбран фреймворк Laravel, т.к. он абсолютно бесплатный, имеет открытый исходный код, и изначально предназначенности для разработки с использованием архитектурной модели Model View Controller.

Также это один из наиболее популярных фреймворков для разработки на PHP (см. рисунок 1.24).

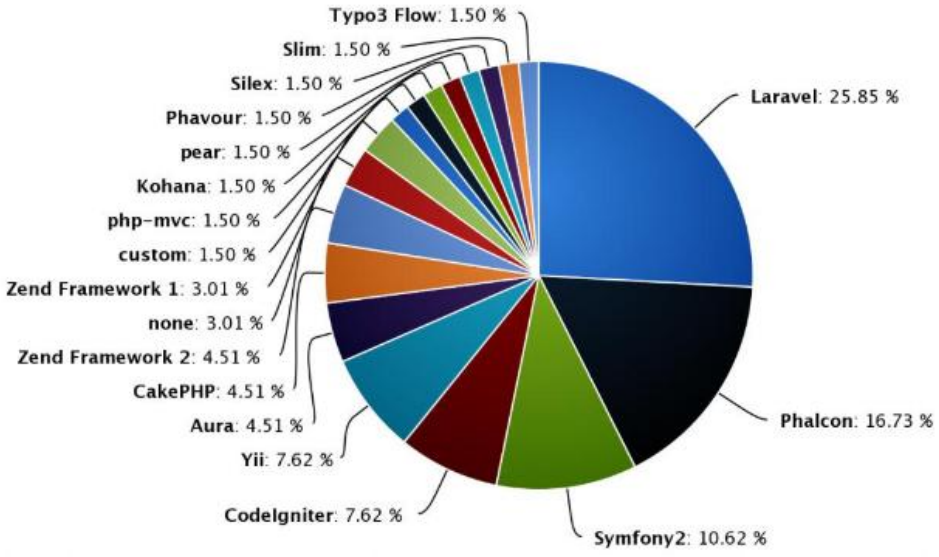

Рисунок 1.22 – Диаграмма популярности фреймворков для PHP

Разработка на Laravel 5 прекрасно подходит для создания:

- CRM и ERP систем;
- корпоративных порталов;
- интернет-магазинов;
- систем онлайн-бронирования.

Laravel является достаточно гибким фреймворком и позволяет решать нестандартные задачи, структурировать веб-сайт в соответствии с существующей логикой и поставленными целями.

Дополнительные преимущества Laravel.

- Обширный функционал.

Создание сайтов любого уровня возможно благодаря огромной функциональности. Используя этот фреймворк, можно реализовать проекты, предоставляющие возможность интеграции необходимого функционала в соответствии с индивидуальными требованиями и особенностями конкретного бизнеса.

- Возможность создать гибкую админпанель.

Можно реализовать наиболее удобный вариант управления ресурсом, создавая индивидуальную панель администратора под задачи конкретного веб-проекта.

- Безопасность баз данных.

Возможность получить несанкционированный доступ к базе данных, созданной с использованием Laravel, крайне сложно. «Высокий уровень безопасности гарантирует надежную защиту от SQL-injection, атак типа XSS, CSRF» [19].

- Регулярные релизы.

Исходный код изменяется с учетом нововведений в PHP и потребностей программистов. Свежие обновления помогают устранить ранее существовавшие проблемы и сделать фреймворк еще более удобным.

- Популярность и активное сообщество.

Наличие большого сообщества открывает простор для динамичной коммуникации, обмена личным опытом и мнениями, решения всевозможных вопросов связанных с проектированием и поддержкой интернет-ресурсов.

- Масштабируемость.

Разработка на Laravel 5 предполагает возможность расширения функционала (интеграции дополнительных модулей) без существенных затрат на изменение текущей системы, а также риска возникновения нежелательных потерь для веб-ресурса.

1.8 Выводы по разделу

В подразделе 1.1 были определены потребности пользователей, выявлены главные приоритеты разработки, а именно: приложение должно обеспечивать лёгкость доступа для пользователя, масштабируемость и безопасность данных. В подразделе 1.2 было рассмотрено три сайта для выпускников высших учебных заведений. В ходе рассмотрения был оценён функционал и выявлены положительные и отрицательные стороны каждого из ресурсов. Удачные решения в реализации этих ресурсов (например, интерактивная карта) можно применить и к нашей будущей информационной системе. В подразделе 1.3 были сформулированы требования к системе. В разделах 1.5 – 1.7 определены архитектура и средства разработки приложения. Принято решение разрабатывать информационную систему «Выпускники кафедры ПМиП» как клиент-серверное приложение (web-сайт) с трёхуровневой архитектурой на основе модели MVC и использовать PHP-фреймворк Laravel. Использование фреймворка обеспечит

строгое следование модели MVC, а значит гибкость и масштабируемость приложения, а также низкоуровневую безопасность приложения.

2 РАЗРАБОТКА WEB-ПРИЛОЖЕНИЯ

2.1 Разработка базы данных

2.1.1 Концептуальное проектирование

Для проектирования базы данных необходимо определить какие именно данные о выпускниках нам необходимо хранить. В профиле пользователя должны быть размещены:

- фамилия, имя, отчество;
- год выпуска;
- специальность кафедры, которую оканчивал выпускник;
- квалификация (бакалавр или магистр);
- учёная степень, если выпускник продолжил научную карьеру;
- информация о втором образовании, если есть;
- место проживания выпускника (страна, город);
- место работы (название компании, предприятия);
- должность;
- фотография (аватар);
- ссылка на личный сайт;
- информация о пользователе, которую он пожелает оставить.

Таким образом, выявляются 9 сущностей: роль, пользователь, профиль, город, страна, специальность, квалификация, учёная степень, научная область. Сущности и связи между ними изображены на ER-диаграмме [21] (см. рис. 2.1).

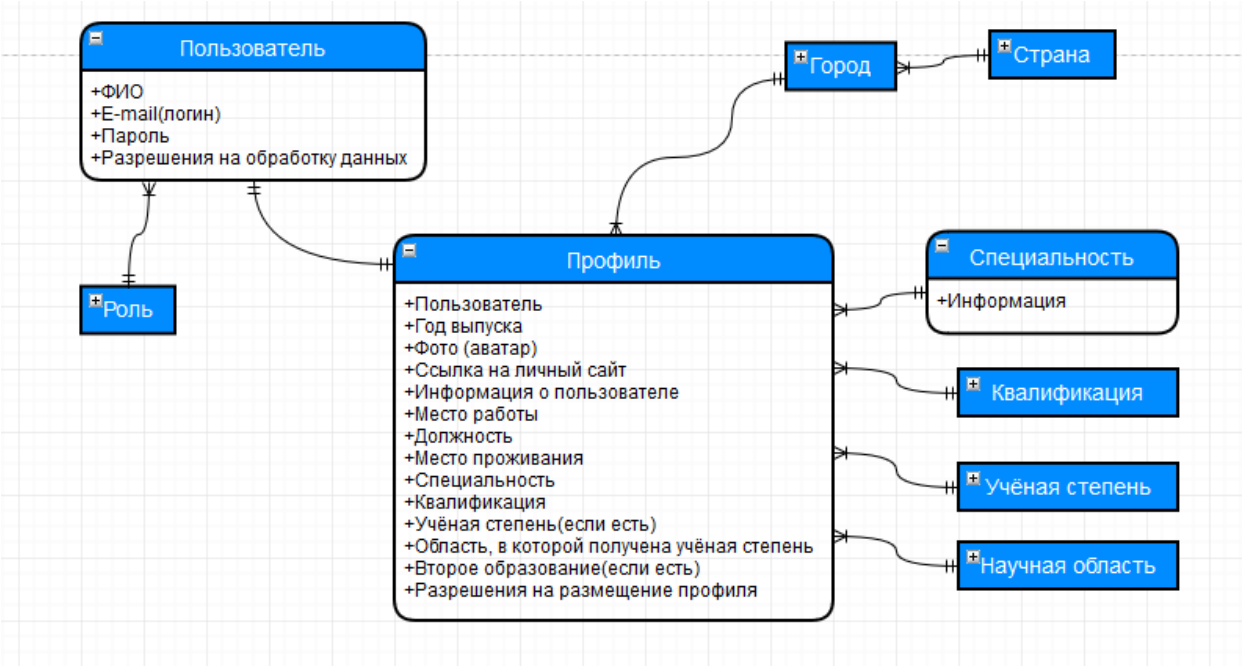

Рисунок 2.1 - ER-диаграмма

2.1.2 Логическое проектирование

На этапе логического проектирования каждую сущность необходимо представить в виде таблицы. Кроме таблиц-сущностей в базе данных создаётся таблица «password_resets» для сброса пароля, в случае, если пользователь его забыл. Эта таблица не связана с другими. Итак, база данных будет состоять из 10 таблиц: password_resets, roles, users, profiles, qualifications, specialties, fields, degrees, countries, cities.

Все таблицы разработанной базы данных находятся в третьей нормальной форме [21,22]. Каждая таблица имеет только один первичный ключ, то есть разработанная база данных находится в нормальной форме Бойса-Кодда. Реляционная схема базы данных приведена на рисунке 2.2.

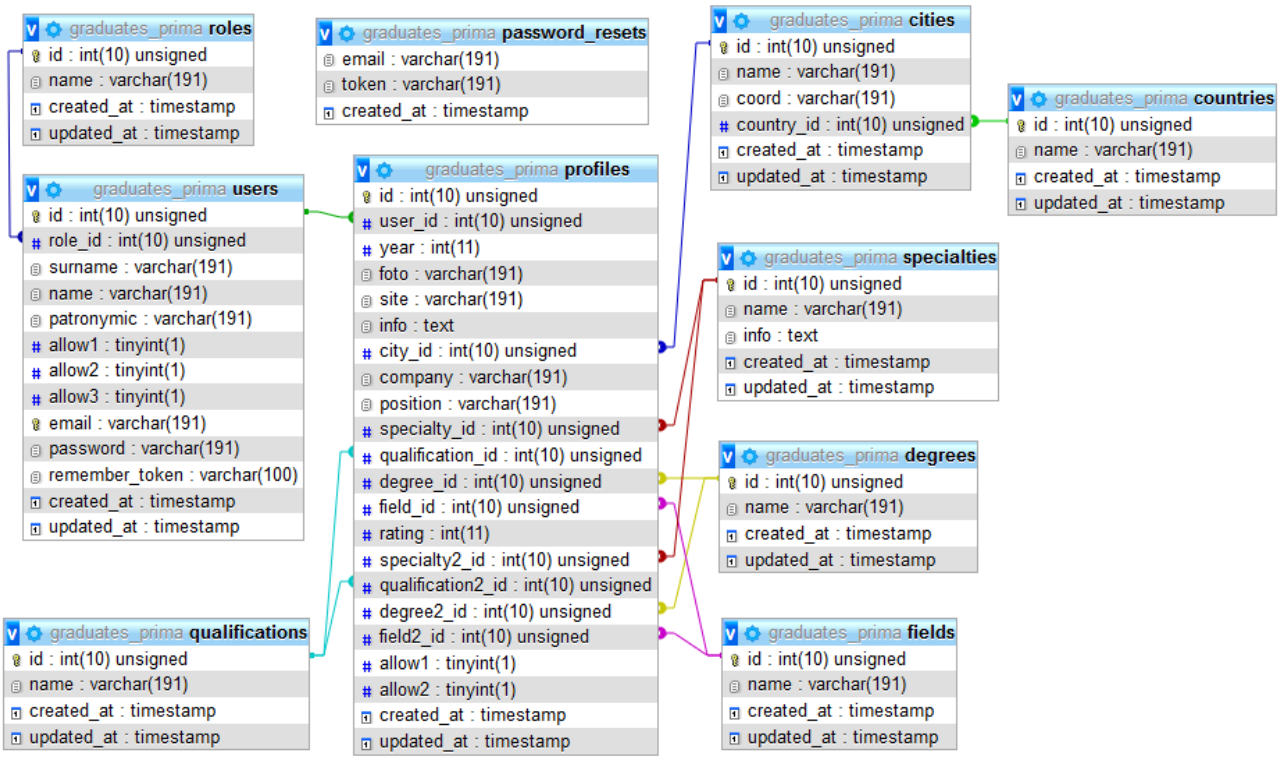

Рисунок 2.2 – Схема реляционной БД

Ключевой таблицей нашей БД является таблица «profiles». Она хранит информацию профилей, создаваемых пользователями.

Описания назначения и свойств полей, а также взаимосвязей таблиц базы данных приведены далее в таблицах 2.1-2.11

Таблица 2.1

Таблица «roles» – роли

	Тип данных	Значение по умолчанию	Допустимо NULL	Первичный ключ	Внешний ключ	Ограничения
id	int	-	-	+	-	-
name	varchar(255)	-	-	-	-	-
created_at	datetime	-	+	-	-	-
updated_at	datetime	-	+	-	-	-

Эта таблица содержит роли, которые могут присваиваться зарегистрированным пользователям. На данный момент роли всего две: студент и выпускник. Но таблица необходима в случае, если последующие версии приложения будут поддерживать новые роли.

Таблица 2.2

Таблица «cities» – города

	Тип данных	Значение по умолчанию	Допустимо NULL	Первичный ключ	Внешний ключ	Ограничения
id	int	-	-	+	-	-
name	varchar(255)	-	-	-	-	-
country_id	int	-	+	-	+	-
coord	varchar(255)	-	+	-	-	-
created_at	datetime	-	+	-	-	-
updated_at	datetime	-	+	-	-	-

При заполнении своего профиля выпускники указывают место проживания (страну и город). Данные о городах представлены в виде: названия города(name) и координат (coord). Каждый город связан внешним ключом со страной, в которой находится.

Таблица 2.3

Таблица «users» – зарегистрированные пользователи

	Тип данных	Значение по умолчанию	Допустимо NULL	Первичный ключ	Внешний ключ	Ограничения
id	int	-	-	+	-	-
role_id	int	-	-	-	+	-
surname	varchar(255)	-	-	-		Только буквы кириллицы
name	varchar(255)	-	-	-	-	Только буквы кириллицы
patronymic	varchar(255)	-	+	-	-	Только буквы кириллицы
allow1	boolean	0	-	-	-	0 или 1
allow2	boolean	0	-	-	-	Только «1»
allow3	boolean	0	-	-	-	0 или 1
email	varchar(255)	-	-	-	-	Шаблон, уникальный
password	varchar(255)	-	-	-	-	Минимум 6 символов
remember_token	varchar(100)	-	+	-	-	-
created_at	datetime	-	+	-	-	-
updated_at	datetime	-	+	-	-	-

При регистрации пользователь должен указать свою роль (студент или выпускник), ввести свою фамилию(surname), имя(name), отчество(patronymic), электронный адрес(email), который будет использоваться, как логин, и пароль(password), длиной не менее 6-ти символов. Каждый пользователь связан внешним ключом с таблицей «roles».

Так же пользователь может дать разрешения на:

- распечатку его данных в списках выпускников на бумаге (allow1);
- публикацию данных на сайтах НИУ ЮУрГУ (allow2). Это разрешение обязательно должно быть дано, ведь в случае запрета, регистрация выпускника не имеет смысла;
- публикацию на других ресурсах в интернете (например, на «Яндекс-карты») (allow3).

В столбец «remember_token» будет записан токен для хранения идентификаторов «запомнить меня»

Таблица 2.4

Таблица «countries» – страны

	Тип данных	Значение по умолчанию	Допустимо NULL	Первичный ключ	Внешний ключ	Ограничения
id	int	-	-	+	-	-
name	nvarchar(100)	-	-	-	-	-
created_at	datetime	-	+	-	-	-
updated_at	datetime	-	+	-	-	-

В этой таблице хранятся названия (name) стран. Каждой стране могут принадлежать 1 и более городов.

Таблица 2.5

Таблица «specialties» – специальности

	Тип данных	Значение по умолчанию	Допустимо NULL	Первичный ключ	Внешний ключ	Ограничения
id	int	-	-	+	-	-
name	varchar(255)	-	-	-	-	-
info	text	-	+	-	-	-
created_at	datetime	-	+	-	-	-
updated_at	datetime	-	+	-	-	-

2 поля таблицы «profiles» связаны внешним ключом с таблицей «specialties». Эти поля отвечают за полученную выпускником специальность и специальность 2-го образования, если оно есть. Каждая специальность характеризуется названием (name) и информацией о ней (info).

Таблица 2.6

Таблица «fields» – научные области

	Тип данных	Значение по умолчанию	Допустимо NULL	Первичный ключ	Внешний ключ	Ограничения
id	int	-	-	+	-	-
name	varchar(255)	-	-	-	-	-
created_at	datetime	-	+	-	-	-
updated_at	datetime	-	+	-	-	-

В таблице хранятся названия (name) областей, в которых получены учёные степени. Например, если выпускник указывает в профиле, что является доктором наук в области философии, то профиль связывается внешним ключом с записью «философия» таблицы «fields».

Таблица 2.7

Таблица «qualifications» – квалификации

	Тип данных	Значение по умолчанию	Допустимо NULL	Первичный ключ	Внешний ключ	Ограничения
id	int	-	-	+	-	-
name	varchar(255)	-	-	-	-	-
created_at	datetime	-	+	-	-	-
updated_at	datetime	-	+	-	-	-

Таблица содержит 3 наименования – бакалавр, специалист, магистр. В профиле выпускник указывает какую квалификацию он получил.

Таблица 2.8

Таблица «degrees» – учёные степени

	Тип данных	Значение по умолчанию	Допустимо NULL	Первичный ключ	Внешний ключ	Ограничения
id	int	-	-	+	-	-
name	varchar(255)	-	-	-	-	-
created_at	datetime	-	+	-	-	-
updated_at	datetime	-	+	-	-	-

Содержит наименования: кандидат наук, доктор наук. Если выпускник имеет учёную степень, он также может указать это в своём профиле.

Таблица 2.9

Таблица «password_resets» – таблица для сброса паролей

	Тип данных	Значение по умолчанию	Допустимо NULL	Первичный ключ	Внешний ключ	Ограничения
email	varchar(255)	-	-	+	-	-
token	varchar(255)	-	-	-	+	-
created_at	datetime	-	+	-	-	-

Поле email хранит электронный адрес пользователя, на который высылается токен, хранящийся в поле «token». Перейдя на страницу сайта с использованием этого токена, пользователь сможет восстановить забытый пароль.

Таблица 2.10

Таблица «profiles» – профили пользователей

	Тип данных	Значение по умолчанию	Допустимо NULL	Первичный ключ	Внешний ключ	Ограничения
id	int	-	-	+	-	-
user_id	int	-	-	-	+	-
year	int	-	-	-	-	-
foto	varchar(255)	-	+	-	-	-
site	varchar(255)	-	+	-	-	-
info	text	-	+	-	-	-
city_id	int	-	+	-	+	-
company	varchar(255)	-	+	-	-	-
position	varchar(255)	-	+	-	-	-
specialty_id	int	-	+	-	+	-
qualification_id	int	-	+	-	+	-
degree_id	int	-	+	-	-	-
field_id	int	-	+	-	+	-
rating	int	-	+	-	-	-
specialty2_id	int	-	+	-	+	-
qualification2_id	int	-	+	-	+	-
degree2_id	int	-	+	-	-	-
field2_id	int	-	+	-	+	-
allow1	boolean	0	-	-	-	0 или 1
allow2	boolean	0	-	-	-	0 или 1
created_at	datetime	-	+	-	-	-
updated_at	datetime	-	+	-	-	-

Каждый профиль связан внешним ключом с пользователем связью «один к одному» т.е. 1 пользователь может иметь только 1 профиль, а профиль может быть привязан только к 1 пользователю. В анкете выпускник должен обязательно указать год выпуска (year) и, по желанию, информацию о месте проживания, месте работы (company), должности (position), своей специальности, научной степени и втором образовании. Ещё пользователь может оценить, насколько ему пригодилось образование, полученное на кафедре (rating), ответив на вопрос

«Работаете по специальности?» и выбрав 1 из 4-х предложенных вариантов ответа.

- Да (соответствует значению rating, равному 4).
- Нет, но в той же области (соответствует значению rating, равному 3).
- Уже нет (соответствует значению rating, равному 2).
- Нет (соответствует значению rating, равному 1).

Также при создании профиля пользователю необходимо дать разрешение на использование своих персональных данных (allow1), и определить, можно ли незарегистрированным пользователям просматривать его профиль (allow2).

2.1.3 Физическое проектирование. Миграции в Laravel.

Фреймворк Laravel предоставляет возможность использования миграций. Миграции – это своеобразный механизм контроля версий для базы данных, он позволяет команде программистов легко изменять и совместно использовать схему базы данных приложения.

С помощью миграций осуществляется доступ к базе данных прямо из приложения, без использования интерфейсов СУБД.

Каждая миграция реализуется как класс и содержит в себе 2 метода – up() и down(). Метод up() срабатывает при запуске миграции, а метод down() при откате. В методе up() мы можем определить тип полей таблицы, свойства этих полей и даже первичные и внешние ключи, связывающие создаваемую таблицу с другими.

Например, полный текст миграции для создания таблицы «users» приведён на рис. 2.3. Полный текст всех миграций расположен в приложении 2.

```
class CreateUsersTable extends Migration{
    public function up(){
        Schema::create('users', function (Blueprint $table) {
            $table->engine = 'InnoDB';
            $table->increments('id');
            $table->integer('role_id')->unsigned();
            $table->foreign('role_id')->references('id')->on('roles')->onDelete('cascade');
            $table->string('surname');
            $table->string('name');
            $table->string('patronymic')->nullable();
            $table->boolean('allow1');
            $table->boolean('allow2');
            $table->boolean('allow3');
            $table->string('email')->unique();
            $table->string('password');
            $table->rememberToken();
            $table->timestamps();
        });
    }

    public function down(){
        Schema::dropIfExists('users');
    }
}
```

Рисунок 2.3 – Текст миграции для создания таблицы Users

Для использования преимуществ механизма миграций необходимо создание в каждой таблице БД 2-х дополнительных полей: «created_at» и «updated_at». Эти поля содержат информацию о том, когда была создана запись таблицы, и когда она была в последний раз изменена.

Создание таблицы «migrations» позволить следить за миграциями (см. рисунок 2.4 и табл. 2.11)

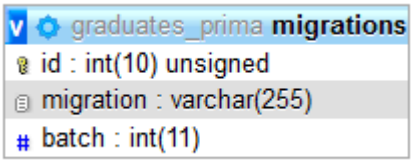

Рисунок 2.4 – Созданная в среде MySQL таблица Migrations

Таблица 2.11

Таблица «migrations» – миграции

	Тип данных	Значение по умолчанию	Допустимо NULL	Первичный ключ	Внешний ключ	Ограничения
id	int	-	-	+	-	-
migration	varchar(255)	-	-	-	-	-
batch	int	-	-	-	-	-
updated_at	datetime	-	+	-	-	-

Наша реляционная база данных создана при помощи системы управления базами данных MySQL 5.5 [23]. Эта СУБД наиболее удобна при размещении сайта на хостинг и поддерживается почти всеми хостинг-провайдерами.

2.2 Разработка архитектуры приложения

Для разработки архитектуры приложения был использован язык графического описания UML (Unified Modeling Language). UML используется для абстрактного описания модели системы. Диаграмма размещения является представлением взаимосвязи между программными и аппаратными компонентами системы, демонстрирует маршруты перемещения объектов в системах [24].

В разделе 1.5 было принято решение реализовывать приложение, как трёхуровневую клиент-серверную систему на основе шаблона MVC (Model-View-Controller). На рисунке 2.5 изображена диаграмма размещения нашего приложения.

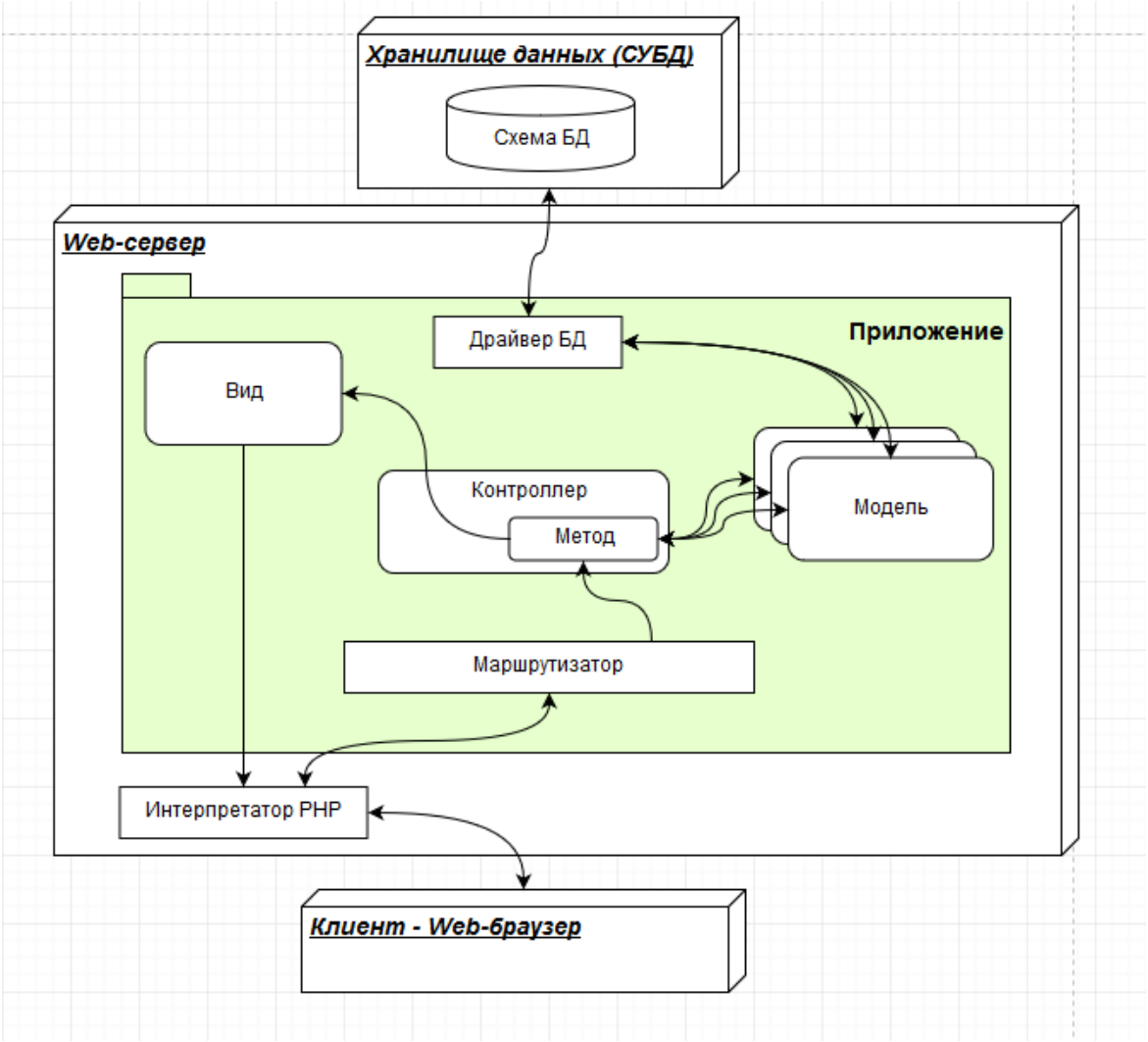

Рисунок 2.5 – Диаграмма размещения (с реализацией модели MVC)

На диаграмме обозначены 3 уровня клиент-серверной системы: клиент (браузер, через который пользователь просматривает веб-сайты), сервер (на котором размещено само наше приложение) и сервер базы данных, используемой приложением.

В блоке «Web-сервер» описан механизм взаимодействия компонентов программы согласно архитектурному шаблону MVC. Рассмотрим этот механизм подробнее. Клиент посылает запрос (например, печатает в адресной строке некий URL-адрес), этот запрос передаётся маршрутизатору. Маршрутизатор (router) – это класс, часть кода, которая определяет, куда (какому контроллеру или какому методу контроллера) передать управление при получении того или иного запроса. Согласно прописанным правилам маршрутизации, роутер передаёт управление методу контроллера. Внутри метода описан алгоритм, осуществляющий взаимодействие с моделями. Модели, согласно шаблону MVC, соответствуют таблицам-сущностям базы данных. На диаграмме изображено взаимодействие моделей с базой данных через драйвер. Драйвер БД – часть кода, реализующая подключение и взаимодействие с базой данных. Алгоритм, описанный внутри

метода может добавлять, изменять, удалять модели, и осуществлять любые другие действия, предусмотренные логикой приложения. Обычно метод возвращает вид, который следует передать пользователю по завершению алгоритма.

Определив, как пользователь взаимодействует с системой, можно перейти к более подробному описанию приложения. Например, определить его компоненты. Компонент – физически существующая часть системы. Компоненты могут включать в себя реализацию класса, реализацию функциональных возможностей системы и др. [25]. Диаграмма компонентов (см. рисунки 2.6, 2.7) служит для представления системы в виде структурных компонентов связей и зависимости между ними.

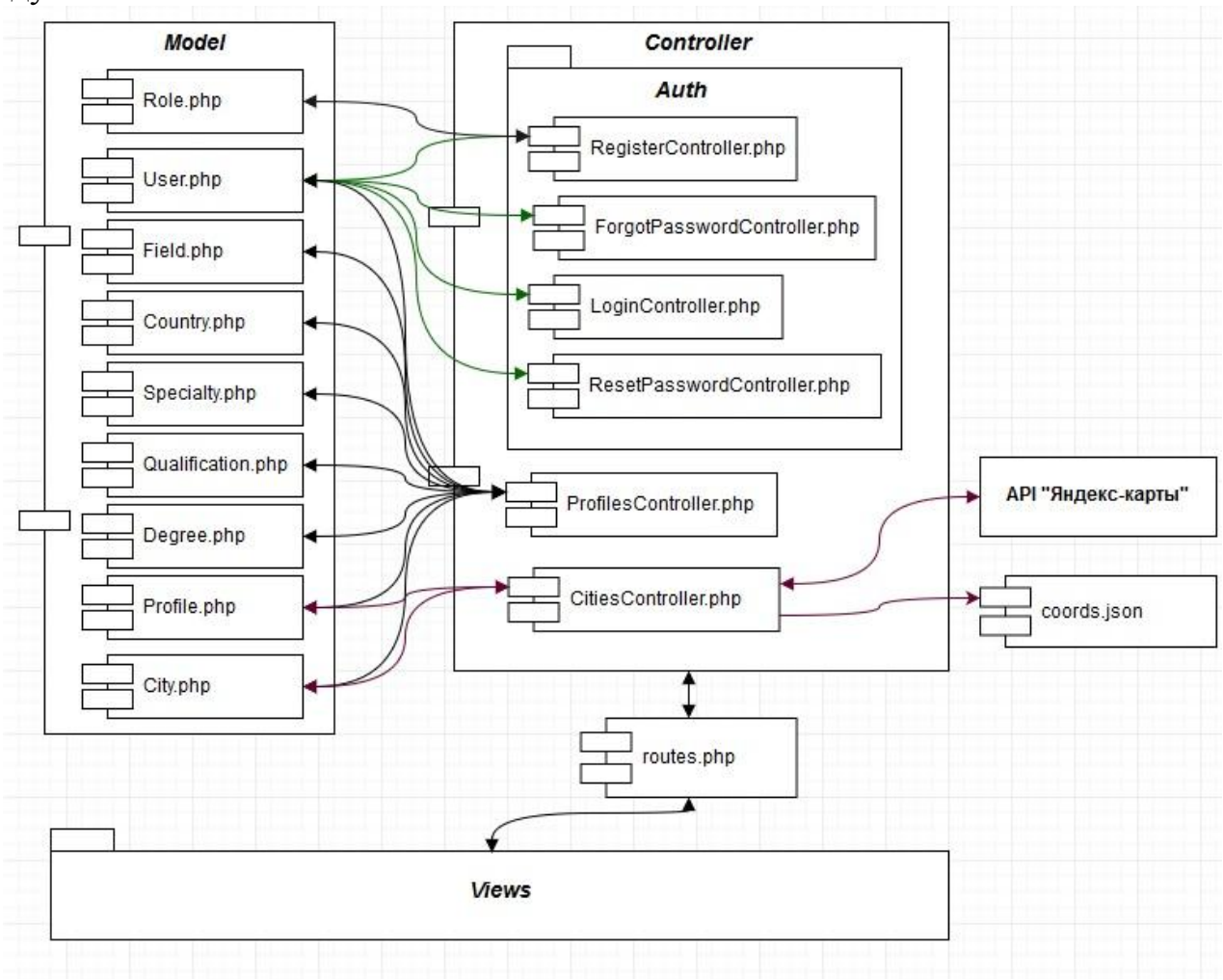

Рисунок 2.6 – Диаграмма компонентов (модели, контроллеры)

На рисунке 2.6 отображены компоненты-модели и компоненты-контроллеры нашего приложения и показаны связи взаимодействия между этими компонентами. Компонент «CityController.php» связан с API (Application Programming Interface) «Яндекс-карты». Этот контроллер отсылает запрос геокодеру, чтобы получить координаты города по его названию. Также City-контроллер отсылает данные в файл coords.json, который потребуется нам позже для вывода отметок на интерактивную карту. Подробное содержание компонента

«Views» и его взаимодействие с другими компонентами отображено на рисунке 2.7.

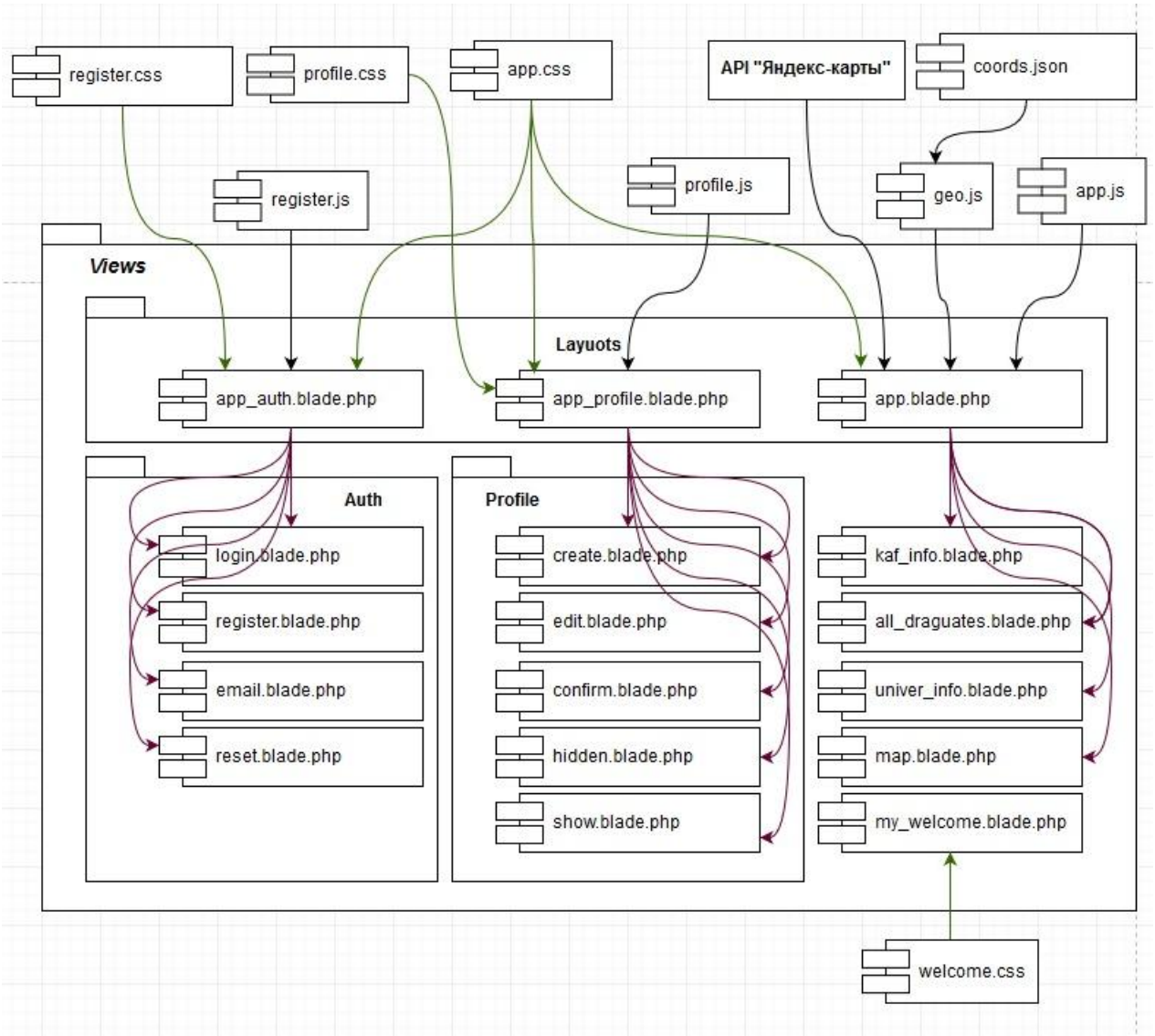

Рисунок 2.7– Диаграмма компонентов (виды)

2.3 Разработка интерфейса пользователя

Основным требованием к нашему ресурсу является удобство для пользователя. Сайт должен быть приятным в использовании, навигация – интуитивно понятной. У выпускника не должно возникать никаких сложностей при желании оставить в системе информацию о себе. Поэтому особое внимание при разработке следует уделить интерфейсу. Дизайн всех форм выполнен в едином стиле. Макеты форм и профиля созданы в среде Adobe Photoshop.

2.3.1 Главная страница

Главная страница сайта (см. рисунок 2.8) должна содержать кнопку входа, кнопку регистрации, основное меню, логотип университета, логотип кафедры,

раздел новостей, интерактивную карту и некоторые элементы статистики (например, информация о том, сколько выпускников уже зарегистрированы в системе).

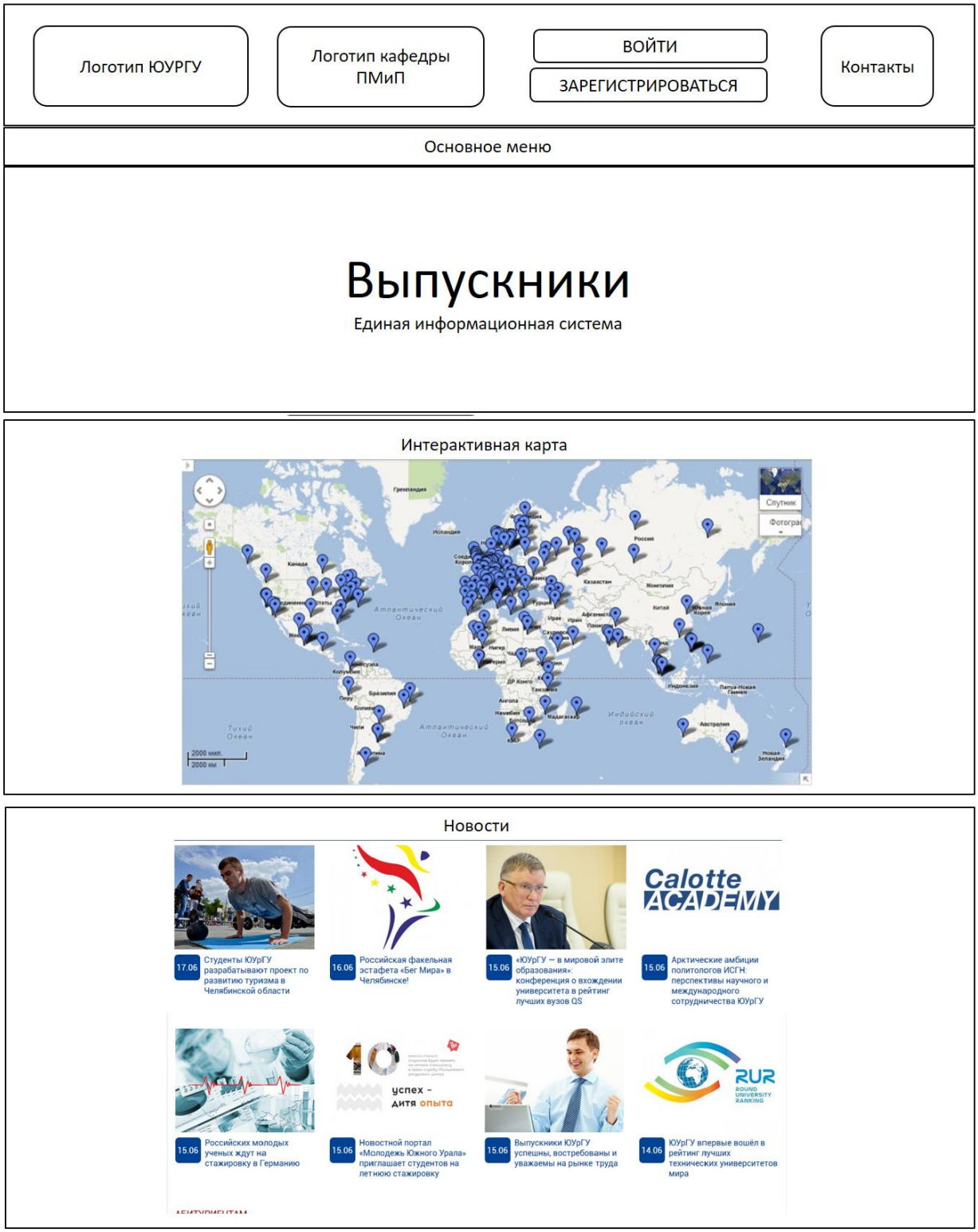

Рисунок 2.8 – Макет главной страницы сайта

Если же на главную страницу переходит уже зарегистрированный и вошедший в систему пользователь, то кнопки «Войти» и «Зарегистрироваться» должны заменяться кнопками «Мой профиль» и «Выйти» (см. рисунок 2.9).

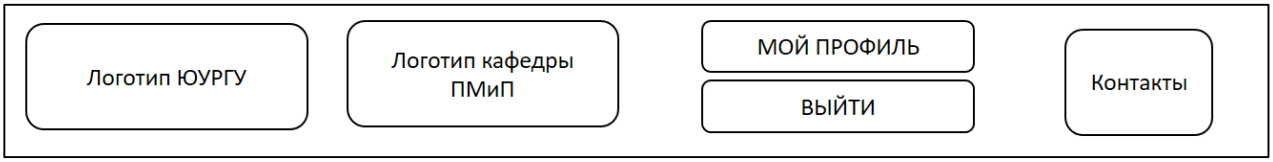

Рисунок 2.9 – Вид шапки сайта для вошедшего в систему пользователя

Основное меню (см. рисунок 2.10) должно содержать такие разделы как:

- об университете;
- о кафедре;
- студенты;
- выпускники;
- интерактивная карта.

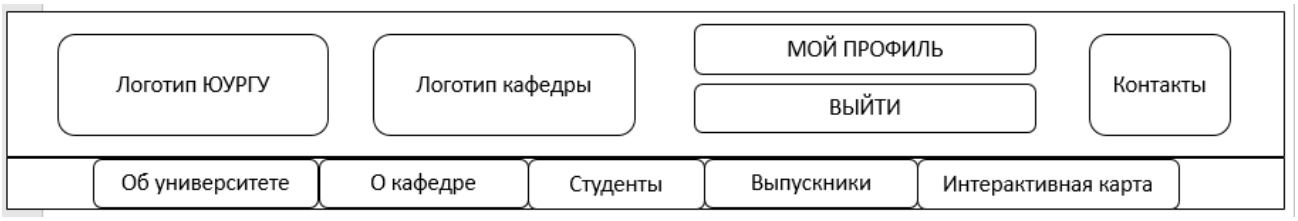

Рисунок 2.10 – Разделы основного меню

2.3.2 Форма регистрации

При нажатии на кнопку «Зарегистрироваться» пользователю должна быть показана форма регистрации (см. рисунок 2.11). На форме должны располагаться 6 полей ввода.

- Фамилия
- Имя
- Отчество
- Электронный адрес
- Пароль
- Подтверждение пароля

2-е кнопки

- Сбросить
- Продолжить

Также пользователю должна предоставляться возможность дать разрешения на использование своих данных и выбрать, кем он является: выпускником или студентом.

При нажатии на кнопку «Сбросить» все поля должны очищаться, при нажатии на кнопку «Продолжить» пользователь должен переходить к следующей форме – форме создания профиля или, другими словами, к анкете выпускника.

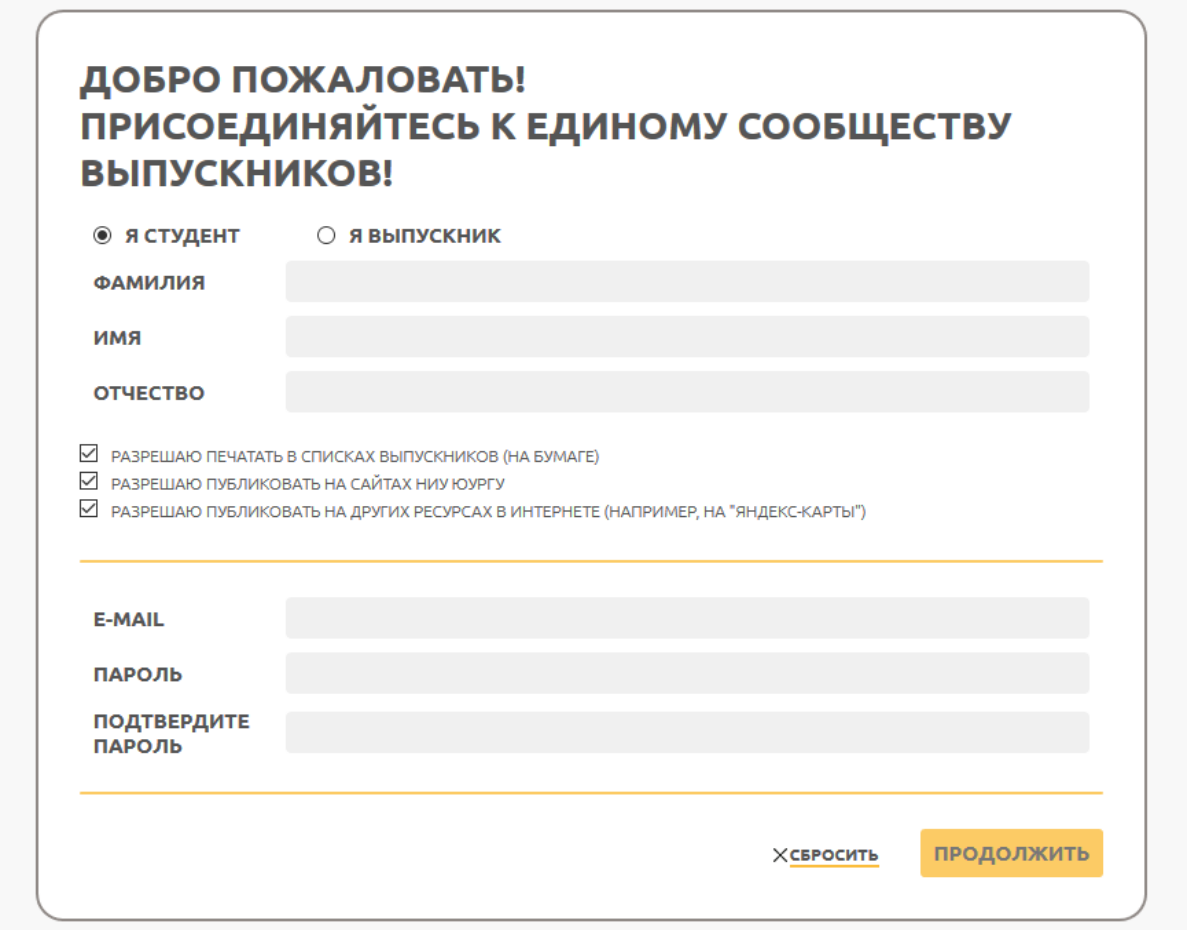

Рисунок 2.11 – Макет формы регистрации

2.3.3 Анкета выпускника

На форме создания профиля (см. рисунок 2.12) должны находится 9 полей ввода.

- Год выпуска
- Направление подготовки (специальность)
- Научная область, в которой выпускником получена учёная степень (в случае, если она есть)
- Страна, в которой живёт и работает выпускник
- Город, в котором живёт и работает выпускник
- Место работы (название компании или предприятия)
- Должность, которую занимает выпускник
- Ссылка на личный сайт, страницу на сайте предприятия или в социальной сети.

- Информация о себе (в этом поле выпускник может указать любую информацию, которую пожелает оставить о себе в своём профиле)

2 поля ввода с выпадающим списком
- Квалификация (бакалавр, специалист или магистр)
- Учёная степень (доктор наук или кандидат наук)

4 кнопки
- Сбросить
- Продолжить
- Кнопка для загрузки фото
- Кнопка «Ещё образование»

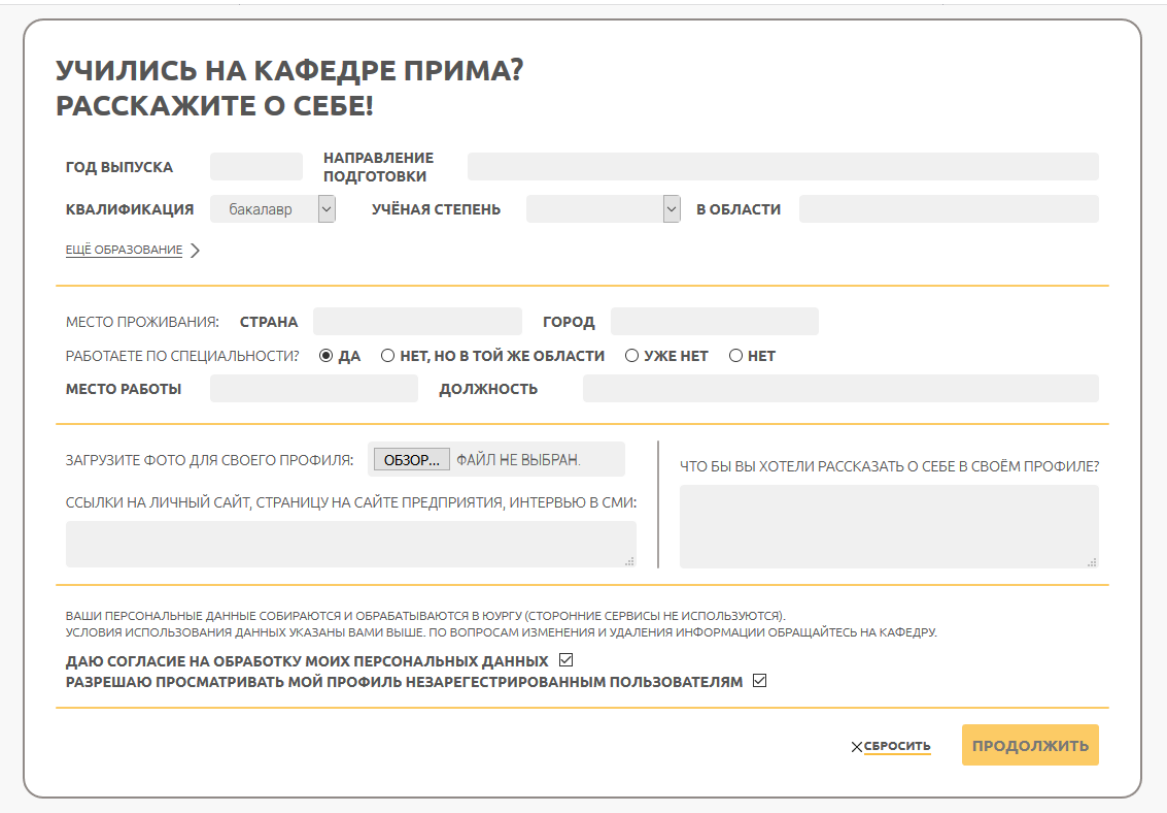

Рисунок 2.12 – Макет анкеты выпускника

По нажатию на кнопку «Ещё образование» открываются дополнительные поля ввода, чтобы выпускник мог указать информацию о своём втором высшем образовании (см. рисунок 2.13)

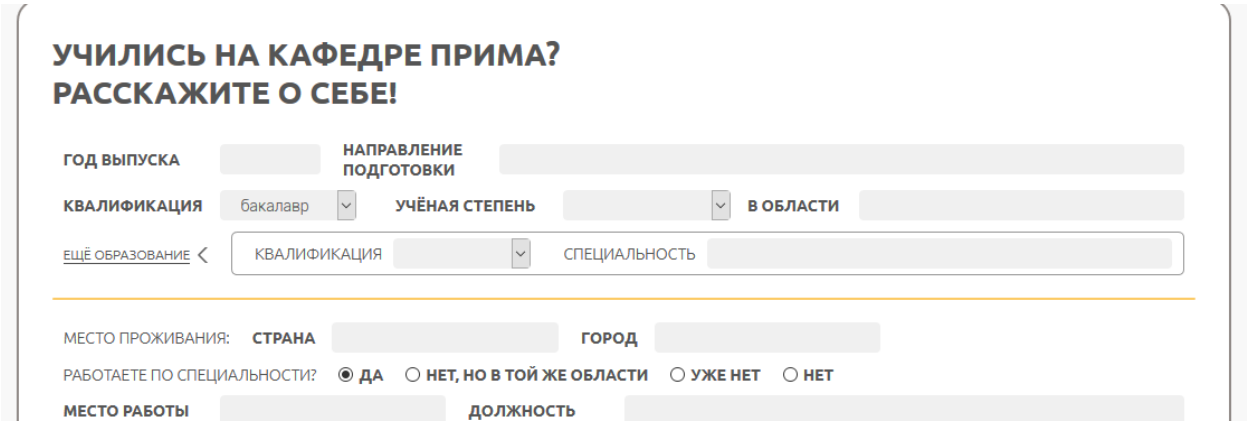

Рисунок 2.13 – Макет анкеты выпускника с открытыми дополнительными полями ввода

Также выпускник может оставить информацию для статистики, ответив, работает ли он по специальности, и дать разрешения:

- на использование своих персональных данных;
- на просмотр своей анкеты незарегистрированными пользователями.

При корректном заполнении анкеты, после нажатия на кнопку «Продолжить» пользователь должен переходить в только что созданный профиль.

2.3.4 Профиль пользователя

В профиле пользователя (см. рисунок 2.14) должна размещаться вся информация, указанная в анкете.

Если пользователь заходит в свой профиль, ему становится доступна кнопка «Редактировать профиль». По нажатию на неё открывается форма схожая с формой создания профиля, в которой можно изменить введённые ранее данные.

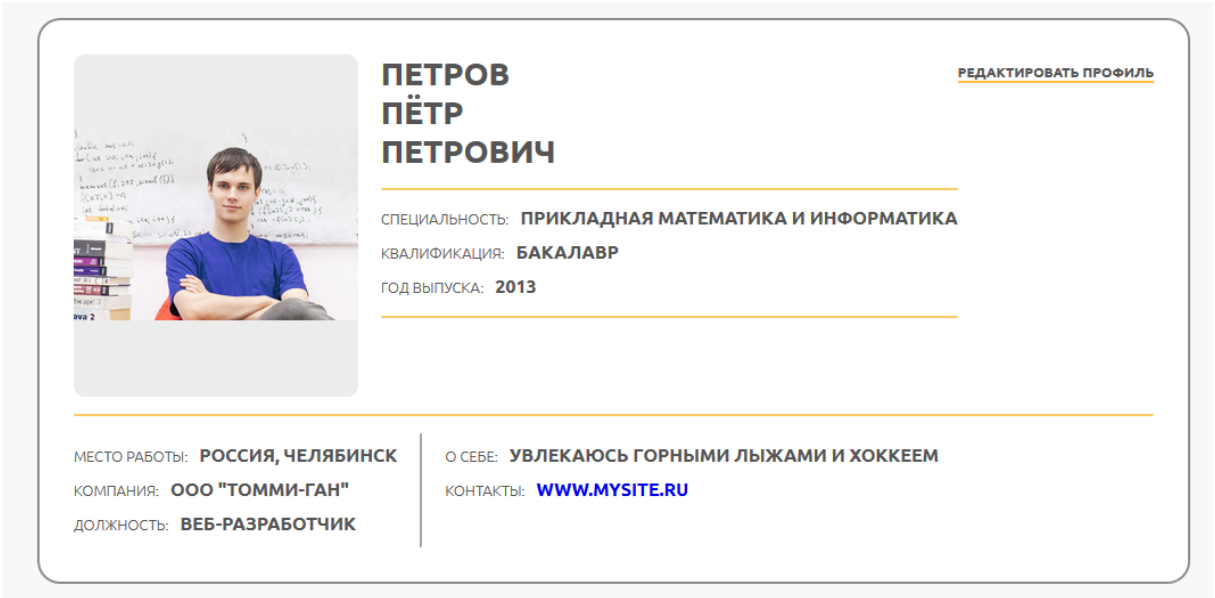

Рисунок 2.14 – Макет профиля пользователя

При переходе на страницу чужого профиля, пользователю не будет доступна кнопка редактирования.

Если незарегистрированный пользователь попытается просмотреть анкету, скрытую владельцем от незарегистрированных посетителей сайта, то появится сообщение о необходимости войти или зарегистрироваться в системе (см. рисунок 2.15). Нажатием на кнопку «Войти» пользователь должен переходить к форме входа, нажатие на кнопку «Зарегистрироваться» – к форме регистрации (см. рисунок 2.11)

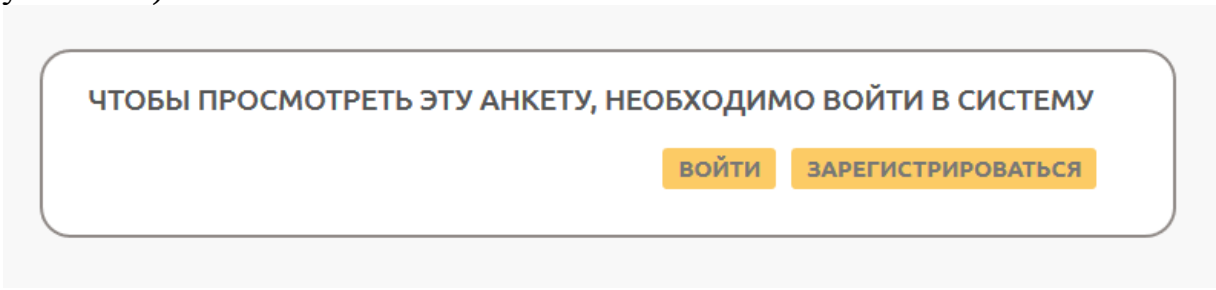

Рисунок 2.15 – Сообщение с предложением войти в систему

2.3.5 Форма входа

Форма входа в систему (см. рисунок 2.16) содержит стандартные поля ввода для электронного адреса и пароля, а также предусматривает возможность поставить отметку «запомнить меня». Кнопка «Войти» перенаправляет посетителя сайта на главную страницу, но уже в роли авторизованного пользователя (конечно, если электронный адрес и пароль оказываются корректными). Кнопка «Забыли пароль?» переводит пользователя к процедуре восстановления пароля.

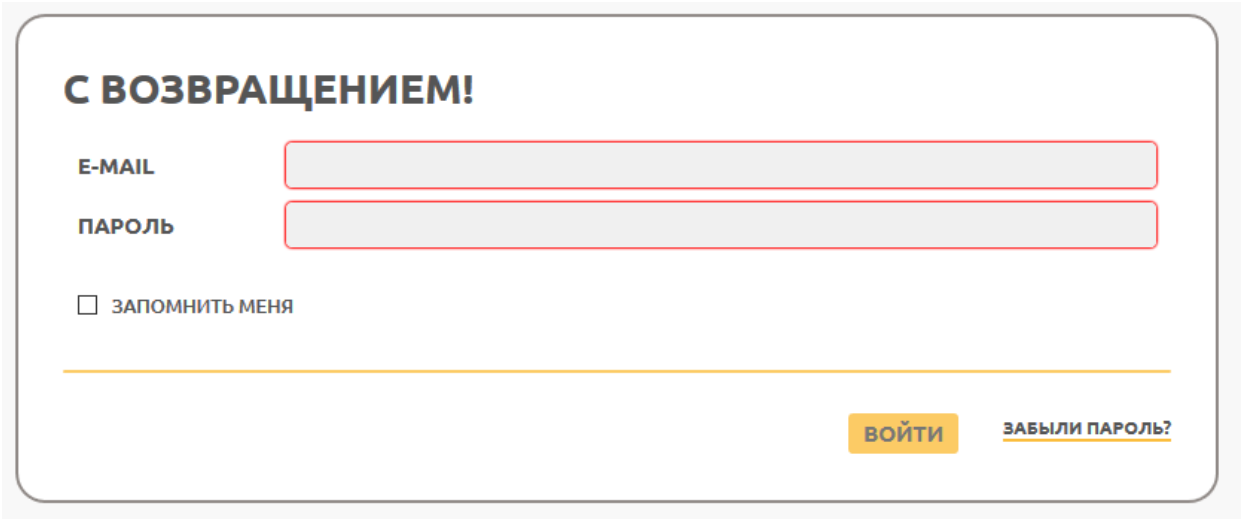

Рисунок 2.16 – Макет формы входа в систему

2.4 Разработка алгоритмов

Алгоритм – это последовательность команд для решения какой-либо задачи. Алгоритмы создаются для того, чтобы решить конкретно поставленную задачу. Например, алгоритм Евклида для нахождения наименьшего общего делителя двух чисел. Исполнитель алгоритма выполняет команды, пока не достигнет конца алгоритма. Алгоритмы в данном разделе были созданы в среде Microsoft Office Visio 2010.

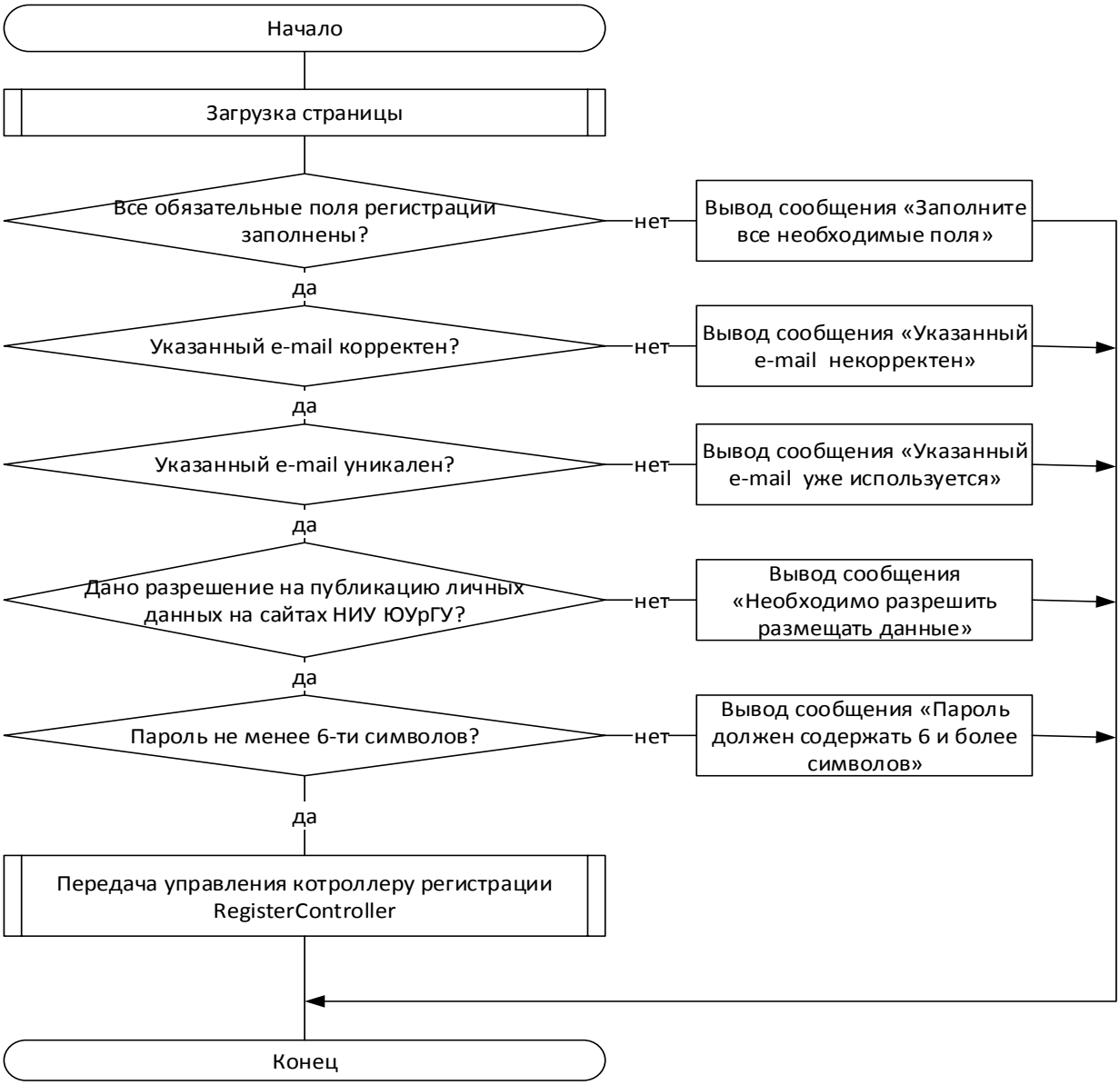

Рисунок 2.17 – Схема алгоритма регистрации пользователя

2.4.1 Алгоритм регистрации пользователя

На рисунке 2.17 приведена схема работы алгоритма регистрации для незарегистрированных пользователей. Чтобы зарегистрироваться пользователь должен заполнить все обязательные поля регистрации, ввести пароль больше шести символов, указать уникальный и корректный адрес электронной почты, дать разрешение на публикацию своих личных данных на сайтах НИУ ЮУрГУ. Если регистрация будет неудачной, то пользователь получит сообщение о соответствующей ошибке. Если регистрация пройдет успешно, то данные введённые пользователем будут отправлены контроллеру RegisterController.

Алгоритм работы контроллера «RegisterController» приведён на рисунке 2.18. Суть алгоритма заключается в создании новой модели User и сохранении её в базе данных.

Рисунок 2.18 – Схема алгоритма работы контроллера RegisterController

2.4.2 Алгоритм размещения отметок на карте

Для реализации интерактивной карты после заполнения или редактирования анкеты выпускника, необходим запуск алгоритма, схема которого изображена на рисунке 2.19. Итак, алгоритм запускается нажатием кнопки «Продолжить» в случае, если анкета заполнена корректно.

Рисунок 2.19 – Схема алгоритма, запускаемого кнопкой «Продолжить» после заполнения анкеты выпускника

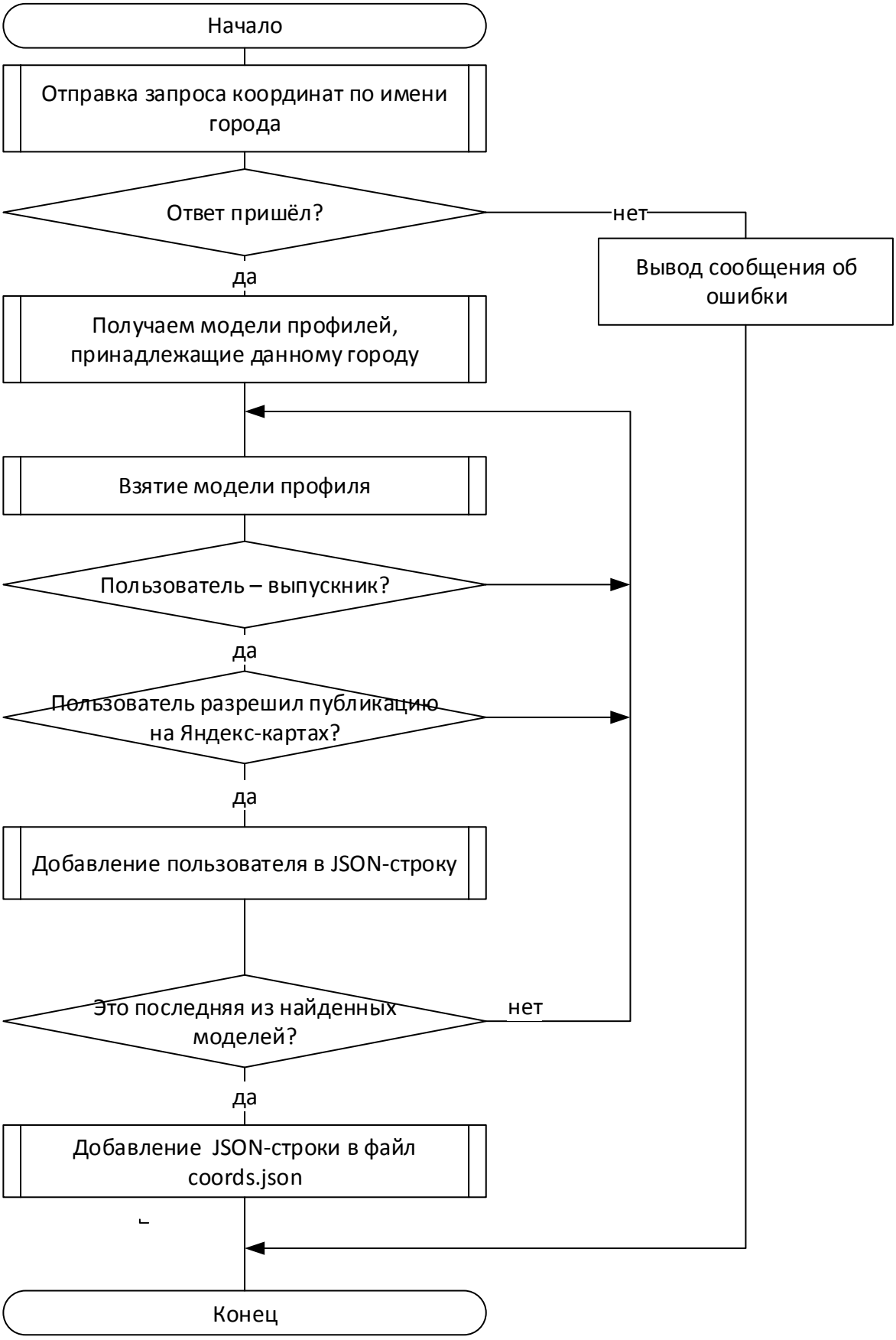

Рисунок 2.20 – Схема алгоритма метода setCoord

Схема алгоритма, реализованного методом setCoord приведена на рисунке 2.20. Сначала отправляется запрос к геокодеру, принадлежащему API «Яндекс-карты». В случае успешного возвращения ответа алгоритм продолжает свою работу. Находятся все модели профилей, принадлежащие данному городу, т.е. все те профили, в которых указан данный город. Потом перебираются все найденные профили. Если профиль принадлежит выпускнику (а не студенту) и если владелец дал разрешение на публикацию его личных данных на Яндекс-картах, то имя и фамилия выпускника добавляются в JSON-строку, описывающую будущую отметку на карте. После перебора всех найденных профилей сформированная JSON-строка помещается в файл coords.json, который в последствии передаётся скрипту, реализующему отображение карты на странице.

2.5 Выводы по разделу

В подразделе 2.1 была разработана база данных для приложения. Описаны все 3 этапа создания реляционной базы данных: концептуальный, логический и физический. Кроме того, в разделе присутствуют описания всех таблиц, созданных в базе с указанием их полей, ограничений и связей.

В разделе 2.2 уточнена ранее выбранная архитектура приложения. Она представлена на диаграмме размещения и диаграмме компонентов.

В разделе 2.3 разработаны макеты основных страниц сайта, формы регистрации и анкеты выпускника.

В разделе 2.4 описан процесс разработки алгоритмов системы. В результате были реализованы основные модули системы.

3 ТЕСТИРОВАНИЕ ПРИЛОЖЕНИЯ

3.1 Методика тестирования

Для проверки работоспособности приложения был выбран метод функционального тестирования. «Функциональное тестирование – это тестирование ПО в целях проверки реализуемости функциональных требований, то есть способности ПО в определённых условиях решать задачи, нужные пользователям. Функциональные требования определяют, что именно делает ПО, какие задачи оно решает» [26]. Основное преимущество функционального тестирования – имитация фактического использования. Поэтому, чтобы выяснить, насколько система удовлетворяет поставленным на первоначальном этапе разработки требованиям, мы представим себя в роли простого пользователя и попробуем совершить несколько разных действий.

3.2 Вход в систему, восстановление пароля

На главной странице сайта (см. рисунок 3.1), помимо меню, информации и логотипов университета и кафедры, располагаются 2-е кнопки. По нажатию на кнопку «Войти» открывается форма входа (см. рисунок 3.2).

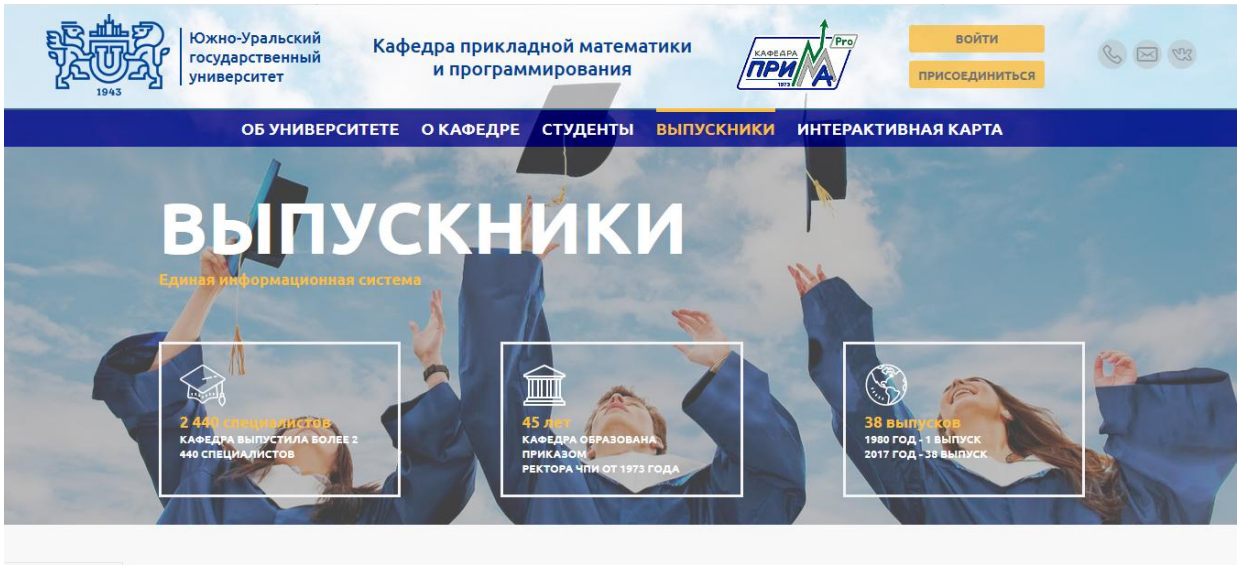

Рисунок 3.1 – Главная страница сайта

Если введённые в форме входа электронный адрес и пароль не соответствуют ни одной из записей таблицы «users», выводится сообщение об ошибке «Неверный логин или пароль» (см. рисунок 3.3). В случае, если пользователь забыл пароль, есть возможность его восстановления по e-mail адресу, указанному при регистрации. Для восстановления пароля необходимо нажать на кнопку «Забыли пароль?». Форма восстановления пароля изображена на рисунке 3.4.

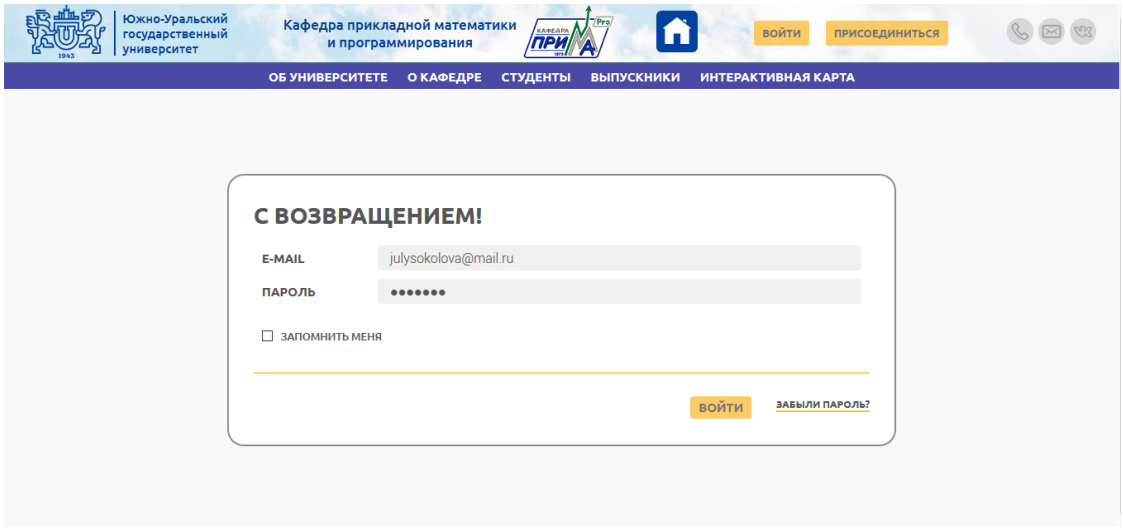

Рисунок 3.2 – Форма входа

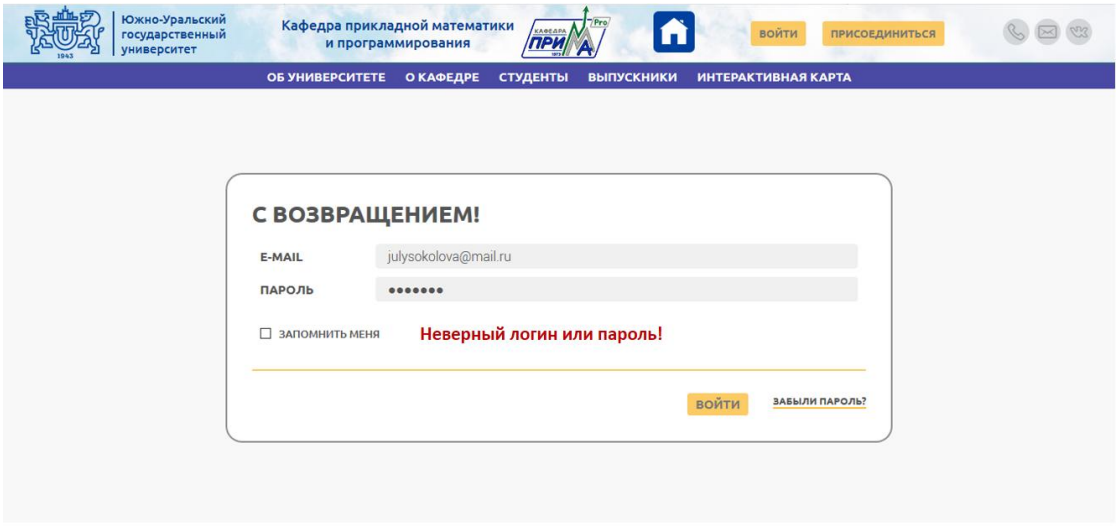

Рисунок 3.3 – Форма входа при попытке войти с неправильно введённым паролем

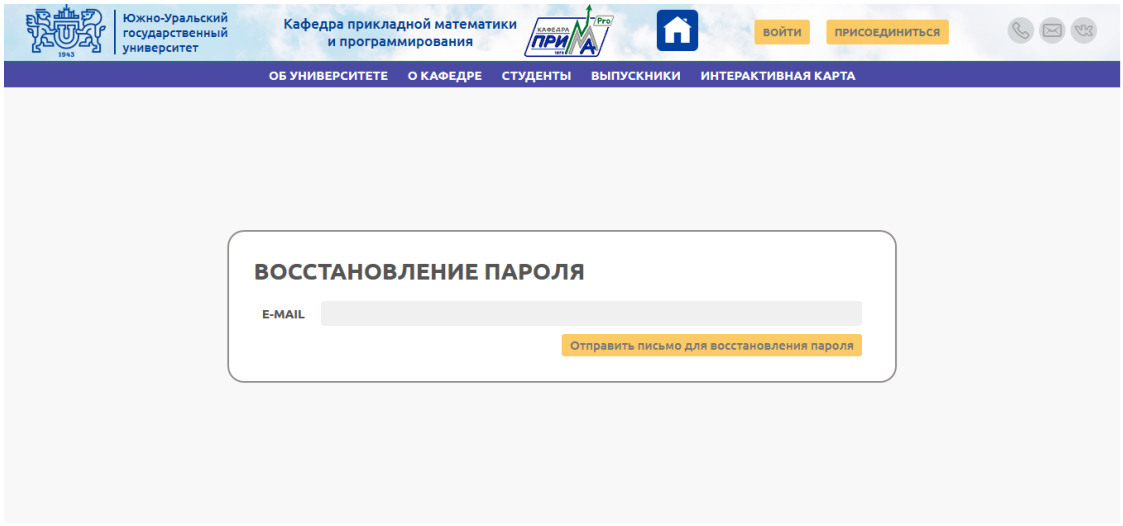

Рисунок 3.4 – Восстановление пароля

Если логин и пароль введены верно, посетитель сайта снова попадает на главную страницу, но уже в роли авторизованного пользователя. Будучи авторизованным, посетитель сайта видит другие 2-е кнопки в шапке страницы: «Мой профиль» и «Выйти». По нажатию на кнопку «Выйти», он также останется на главной странице, поменяв свою роль.

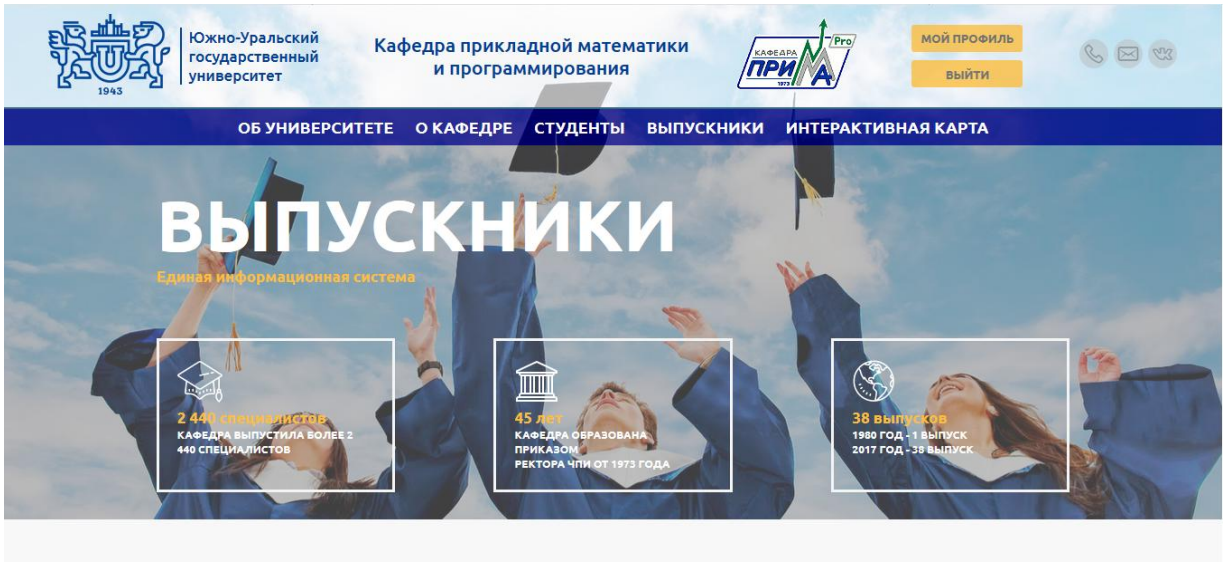

Рисунок 3.5 – Главная страница для авторизованного пользователя

3.3 Регистрация нового пользователя

Нажав кнопку «Присоединиться», гость видит форму регистрации (см. рисунок 3.6).

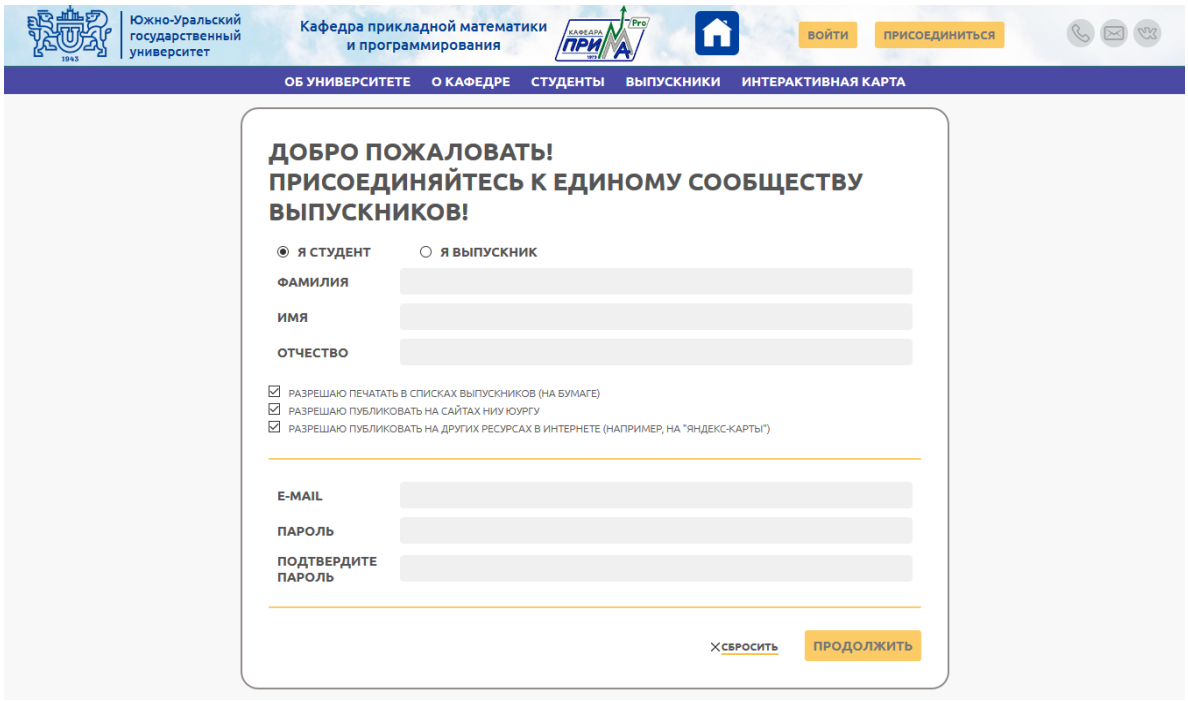

Рисунок 3.6 – Форма регистрации

При вводе некорректных данных получаем соответствующие сообщения об ошибках (см. рисунки 3.7 – 3.9). Обязательными являются все поля, кроме поля «Отчество». Также необходимо поставить отметку – разрешение на публикацию своих данных на сайтах НИУ ЮУрГУ.

Рисунок 3.7 – Попытка использовать латинские буквы для имени

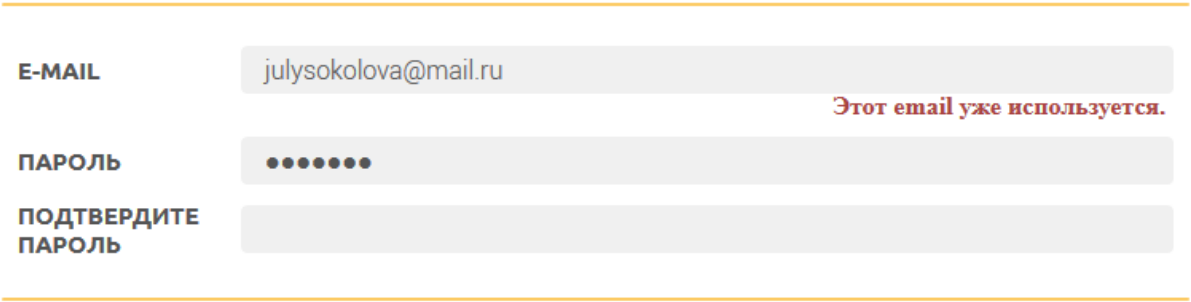

Рисунок 3.8 – Попытка использовать не уникальный электронный адрес

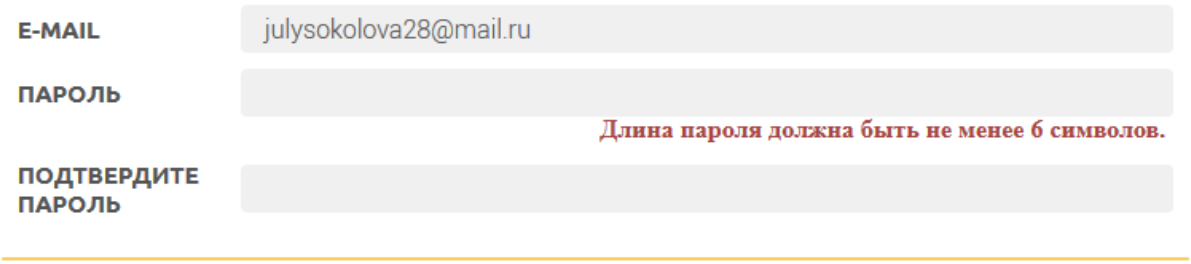

Рисунок 3.9 – Попытка зарегистрироваться со слишком коротким паролем

Если же данные введены правильно, регистрация проходит успешно и пользователь попадает на форму создания профиля.

3.4 Создание профиля

Форма создания профиля или, другими словами, анкета выпускника (см. рисунок 3.10) также требует корректного заполнения. Обязательным является поле «Год выпуска» и галочка согласия на обработку персональных данных. Остальные поля заполняются по желанию.

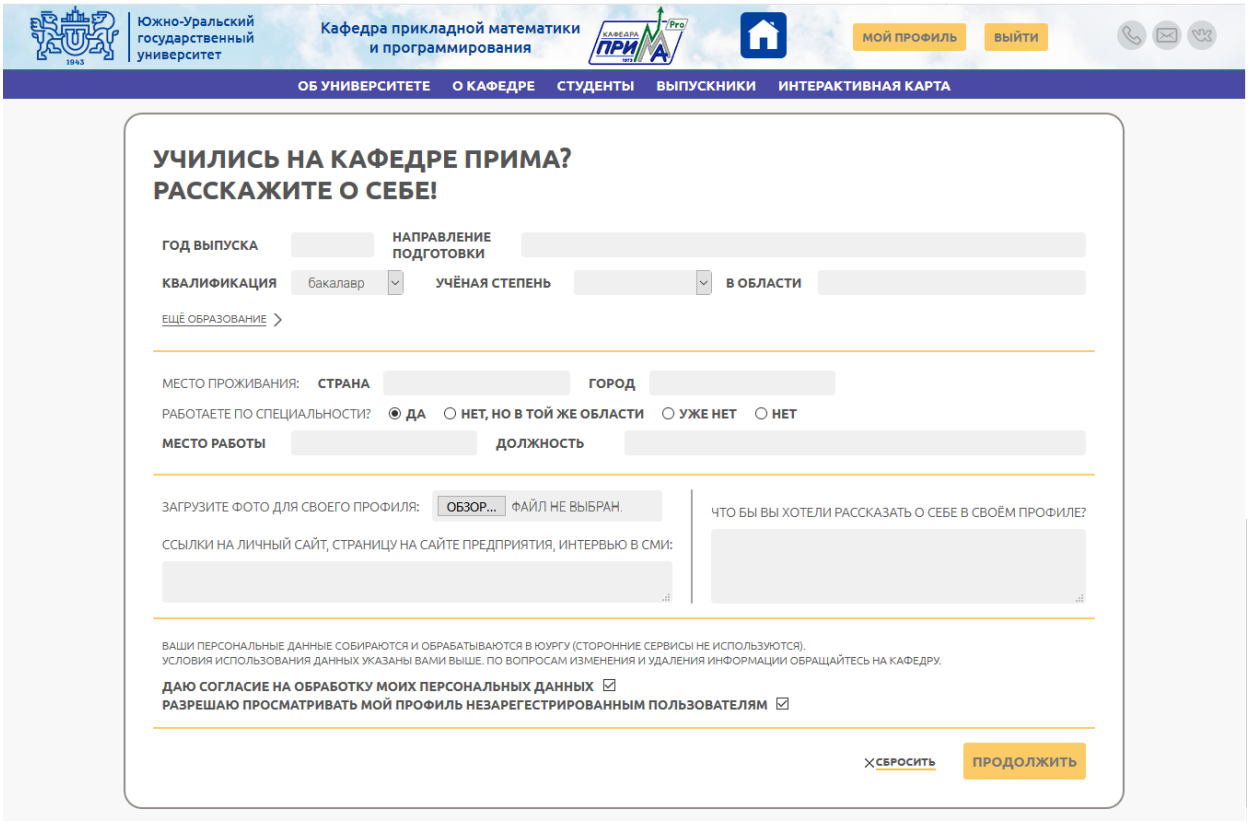

Рисунок 3.10 – Форма создания профиля

После заполнения полей и нажатия кнопки «Продолжить», пользователь переходит в свой профиль (см. рисунок 3.11). Здесь он может загрузить новый аватар для профиля, просто нажав на картинку, заменяющую отсутствующее фото (см. рисунок 3.12).

3.4 Редактирование и удаление профиля

Пользователю также доступна функция редактирования своего профиля. Нажав на кнопку «Редактировать профиль», он переходит к форме изменения введённых ранее данных (см. рисунок 3.13).

Помимо редактирования, пользователь может удалить свой профиль из системы. По нажатию на кнопку «Удалить профиль» в нижнем левом углу страницы, пользователь должен будет подтвердить своё действие, либо вернуться обратно к странице редактирования профиля (см. рисунок 3.14).

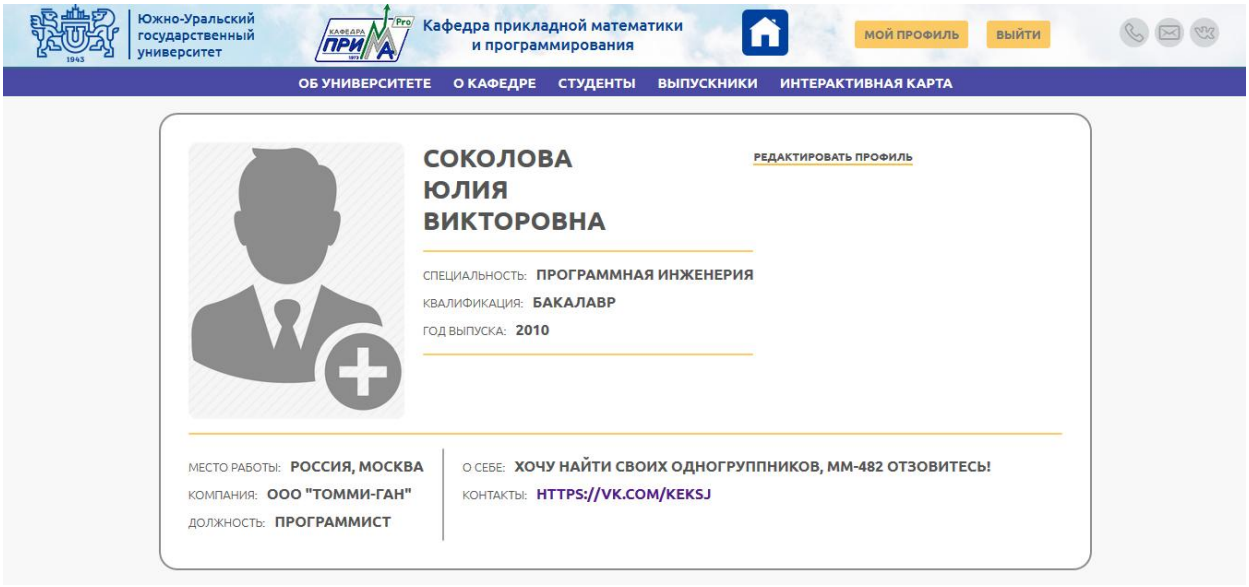

Рисунок 3.11 – Страница профиля

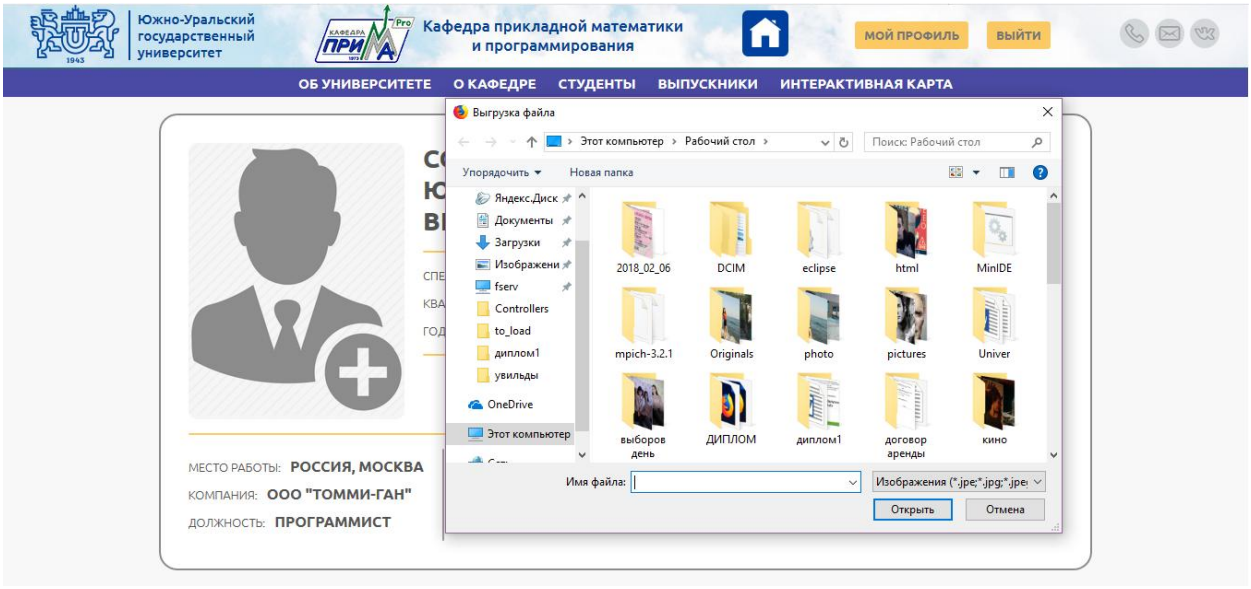

Рисунок 3.12 – Загрузка аватара

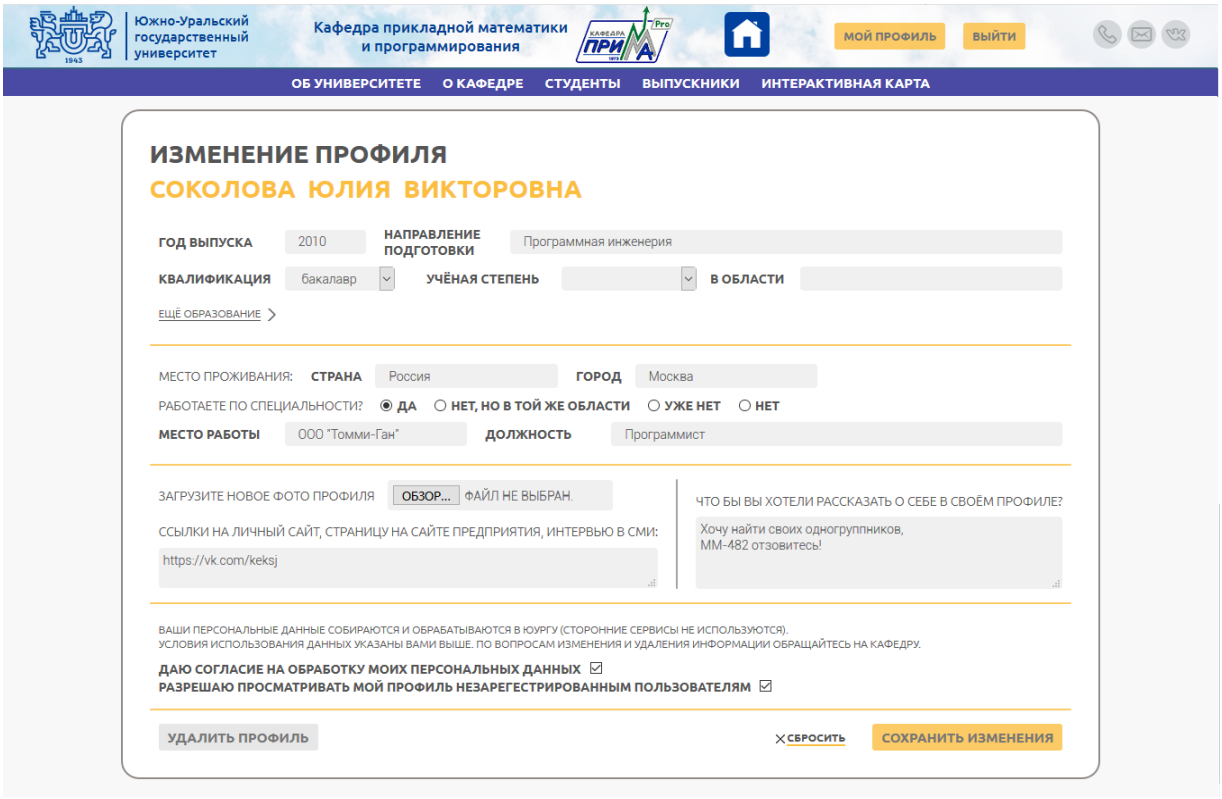

Рисунок 3.13 – Редактирование профиля

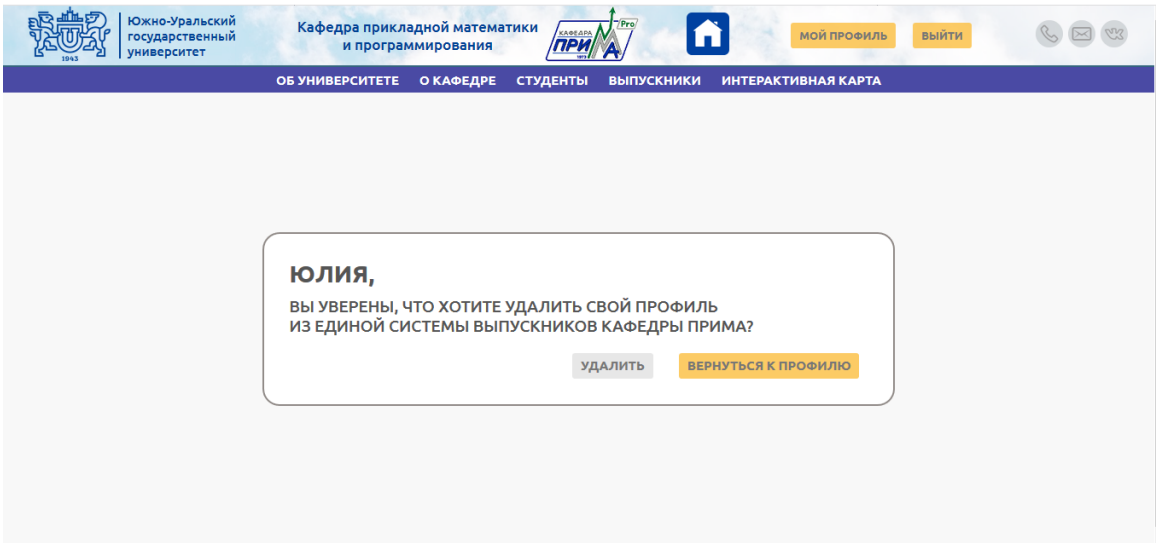

Рисунок 3.14 – Подтверждение удаления профиля

3.5 Интерактивная карта, списки выпускников

После регистрации и заполнения своего профиля, информация о выпускнике должна отобразиться в статистике. В случае, если выпускник указал город, в котором проживает и работает и дал согласие на размещение своих персональных данных на сервисе «Яндекс-карты», информация о нём должна отобразиться на интерактивной карте.

Переходим во вкладку меню «Интерактивная карта» (см. рисунок 3.15). При регистрации мы указали место проживания – город Москва. Щёлкнув на отметке этого города мы увидим ссылку на свой профиль и информацию о том, сколько зарегистрированных выпускников проживают в этом городе (см. рисунок 3.16).

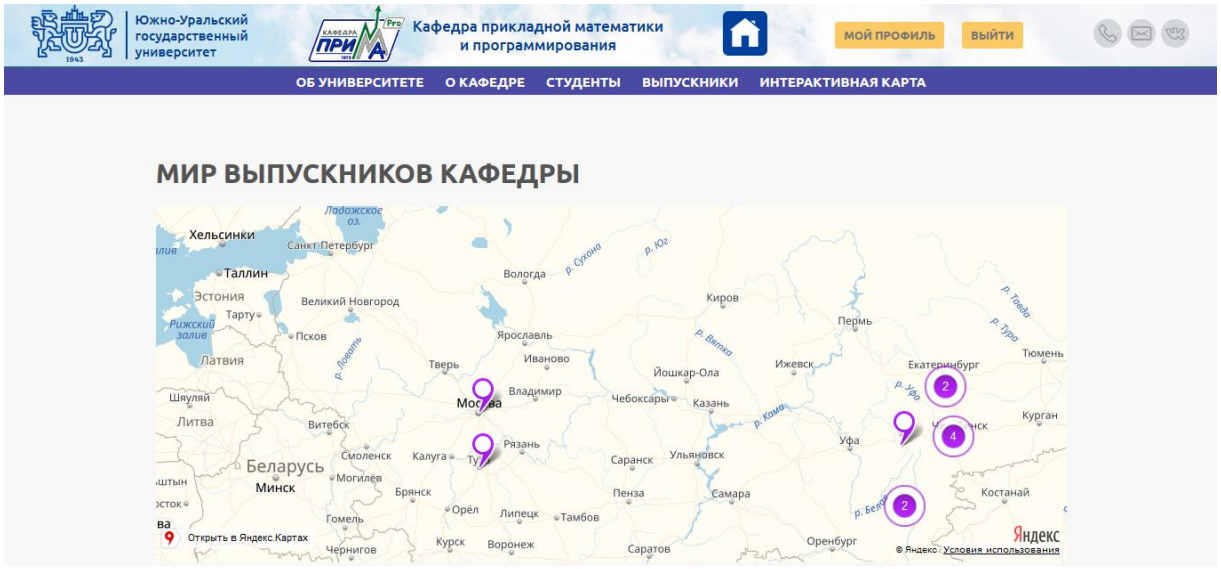

Рисунок 3.15 – Интерактивная карта

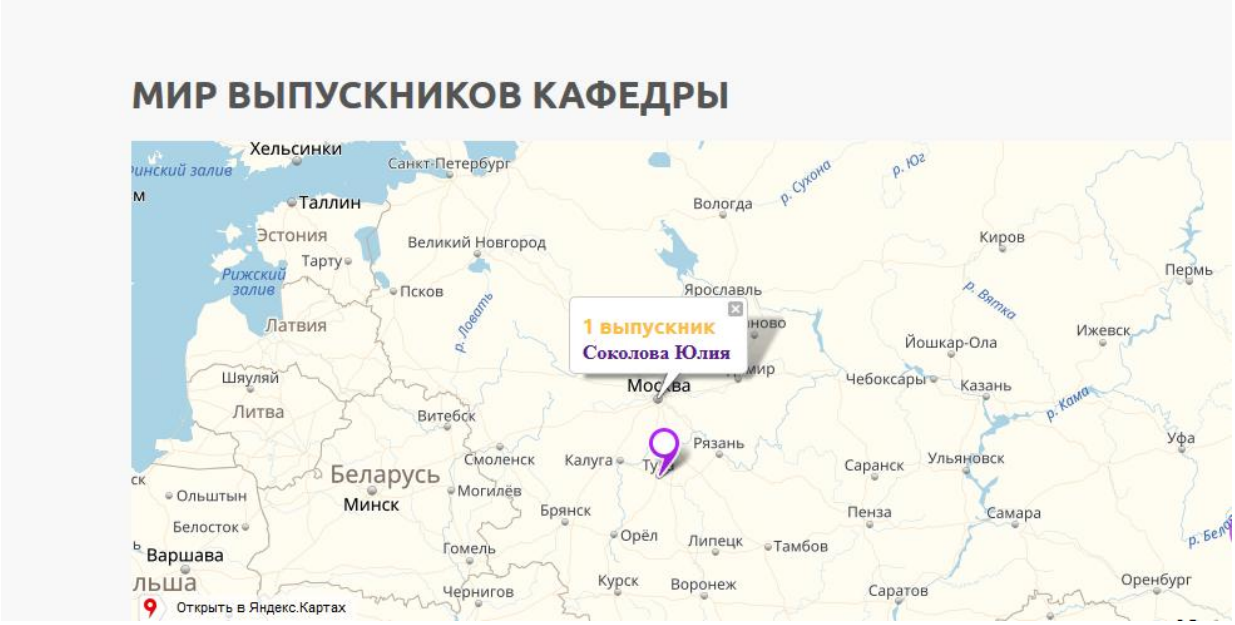

Рисунок 3.16 – Интерактивная карта, щелчок по отметке

Пройдя в раздел «Выпускники» главного меню сайта мы также должны увидеть отображение информации о себе (см. рисунок 3.17).

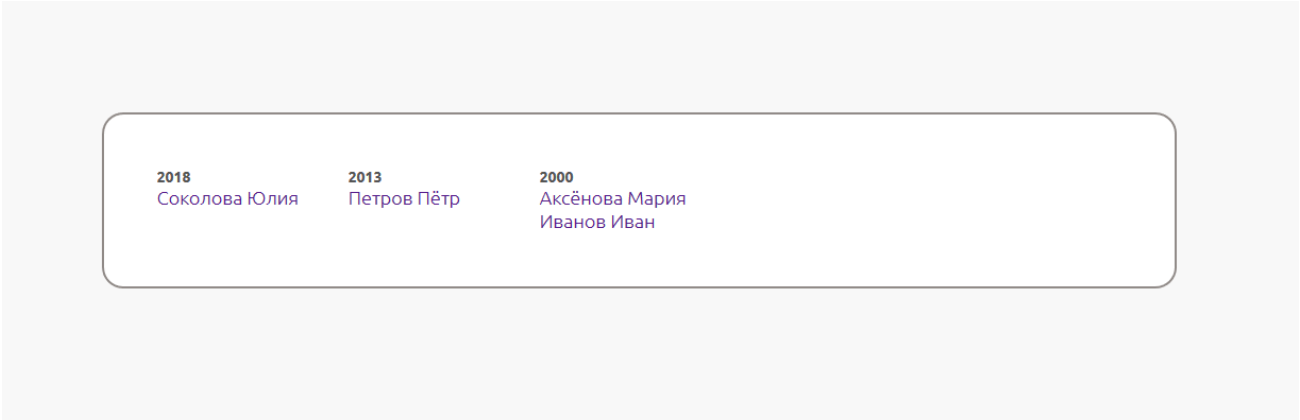

Рисунок 3.17 – Раздел «Выпускники»

3.6 Выводы по разделу

В разделе 3 была проведена проверка работоспособности и корректность обработки ошибок, связанных с регистрацией и авторизацией пользователя. По результатам проверки можно сделать вывод, что реализованная часть функционала работает корректно.

ЗАКЛЮЧЕНИЕ

В данной работе рассмотрена структура информационной системы, ориентированная на сбор, обработку и предоставление данных о выпускниках кафедры прикладной математики и программирования (ПМиП). Проведен анализ потребностей пользователей, рассмотрены уже существующие подобные ресурсы, и на основе этого определены необходимые функции и другие требования к системе.

Основным приоритетом при разработке стала необходимость сделать систему максимально доступной и удобной для пользователя. Исходя из этого было принято решение реализовывать систему, как веб-ресурс с использованием PHP-фреймворка Laravel.

Была спроектирована архитектура приложения, основанная на шаблоне MVC, создана база данных. В результате был разработан визуальный макет сайта, а также серверная часть, отвечающая за обработку запросов и реализацию функционала.

Реализованы базовые возможности: регистрация пользователей, создание профилей, редактирование и удаление профилей. Вся оставленная выпускниками информация хранится в серверной базе данных MySQL. Использованы преимущества фреймворка Laravel: механизм миграций, защита от кроссбраузерных атак.

На заключительном этапе разработки была проведена проверка работоспособности и корректность обработки ошибок, связанных с регистрацией, авторизацией пользователей. Также проверена корректность обработки оставленных данных и представления их в рамках интерактивной карты и списка выпускников.

В результате полученное приложение получилось легко масштабируемым и открытым для дальнейшего развития. Эти преимущества стали следствием выбора архитектуры MVC и использования фреймворка. В дальнейшем приложение может быть расширено до масштабов факультета или даже всего университета. Также оно может быть оснащено дополнительными функциями, такими как, поиск выпускников, отправка сообщений между пользователя, создание групп.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. ГОСТ 33707-2016(ISO/IEC 2382:2015). Информационные технологии. Словарь. – М.: Стандартинформ, 2016. – 206 с.
2. William, S. Davis The Information System Consultant's Handbook. Systems Analysis and Design. / William S. Davis, David C. Yen – CRC Press, 1998. — 800 с.
3. Избачков, Ю.С. Информационные системы: Учебник для вузов. 3-е издание. / Ю.С. Избачков, В.Н. Петров, А.А. Васильев, И.С. Телина – СПб.: Питер, 2010. – 544 с.
4. Фаулер, М. Архитектура корпоративных программных приложений / М. Фаулер; пер. с англ. – М.: «Вильямс», 2006. — 544 с.
5. Фримен А. ASP.NET MVC 5 с примерами на C# 5.0 для профессионалов, 5-е издание / А. Фримен – М.: «Вильямс», 2015. – 736 с.
6. Рогачев, С. Обобщённый Model-View-Controller. URL: http://rsdn.org/article/patterns/generic-mvc.xml#EDAAC (дата обращения – 21.03.2018)
7. Котеров, Д.В. PHP 7: наиболее полн. рук. / Д. В. Котеров, И. В. Симдянов. – СПб.: БХВ-Петербург, 2016. – 1088 с.
8. TIOBE. The software quality company. URL: https://www.tiobe.com/tiobe-index (дата обращения - 14.05.2018)
9. PHP: Manual. URL: http://php.net/manual/en/preface.php (дата обращения - 01.04.2018)
10. PHP: News Archive – 2015. URL: http://php.net/archive/2015.php#id2015-12-03-1 (дата обращения – 28.04.2018)
11. W3Techs. Web Thechnology Surveys. URL: https://w3techs.com/technologies/overview/programming_language/all (дата обращения – 12.05.2018)
12. Ульман, Л. MySQL / Л. Ульман.– М. : ДМК Пресс, 2008. – 352 с
13. Справочник по HTML. URL: htmlbook.ru/html (дата обращения: 24.04.2018).
14. Справочник CSS. URL: htmlbook.ru/css (дата обращения: 28.04.2018).
15. Хоган, Б. HTML5 и CSS3. Веб-разработка по стандартам нового поколения / Б. Хоган.– СПб.: Питер, 2014. – 320 с.: ил.
16. Зервас, Квентин Web 2.0: создание приложений на PHP. / Квентин Зервас; пер. с анг. – М.: ООО «И.Д. Вильямс», 2010. – 544 с.
17. Официальный сайт фреймворка Laravel. URL: https://laravel.com (дата обращения: 02.04.2018)
18. Русскоязычное сообщество Laravel. URL: https://laravel.ru (дата обращения – 02.04.2018)
19. URL: https://www.sitepoint.com (дата обращения – 13.05.2018)

20. Scott, W. Ambler Mapping Objects to Relational Databases: O/R Mapping In Detail. URL: http://www.agiledata.org/essays/mappingObjects.html (дата обращения - 14.05.2018)
21. Тарасов, С.В. СУБД для программиста. Базы данных изнутри. / С.В. Тарасов. – М.: СОЛОН-Пресс, 2015. – 320 с.
22. Астахова, И.Ф. СУБД: язык SQL в примерах и задачах. / И.Ф. Астахова, В.М. Мельников, А.П. Толстобров, В.В. Фертиков. – М. : Физматлит, 2009. – 168 с.
23. Хоган Б. HTML5 и CSS3. Веб-разработка по стандартам нового поколения / Б. Хоган.– СПб.: Питер, 2014. – 320 с.: ил.
24. Новиков, Ф.А. Учебно-методическое пособие по дисциплине «Анализ и проектирование на UML». / Ф.А. Новиков.– СПб.: НИУ ИТМО, 2007. – 286 с.
25. Буч, Г. Язык UML. Руководство пользователя. / Г. Буч, Д. Рамбо, И. Якобсон. – М.: ДМК Пресс, 2008. – 496 с.
26. Майерс, Г. Д. Искусство тестирования программ / Пер. с англ. под ред. Б. А. Позина / Г. Д. Майерс. – М. : Финансы и статистика, 1982. – 176 с.

ПРИЛОЖЕНИЕ 1

ОПИСАНИЕ ПРОГРАММЫ

1.Общие сведения

Сайт «Выпускники кафедры ПМиП». Для работы программы необходимы: сервер Apache, интерпретатор PHP, СУБД MySQL, поддержка фреймворка Laravel. Визуальная структура сайта была реализована с помощью HTML, CSS и JavaScript. Серверная часть была написана на PHP.

2. Функциональное назначение

Данная система предназначена для хранения, предоставления и структурирования информации о выпускниках кафедры.

3. Описание логической структуры

Файлы City.php, Country.php, Degree.php, Field.php, Profile.php, Qualification.php, Role.php, Specialty.php, User.php содержат реализацию одноимённых классов-моделей.

Реализация классов-контроллеров содержится в файлах ProfilesController.php, CitiesController.php, RegisterController.php, LoginController.php, ResetPasswordController.php, ForgotPasswordController.php.

Представления хранятся в папке views:

my_welcome.blade.php – вид главной страницы сайта;

map.blade.php – вид страницы с интерактивной картой;

all_grad.blade.php – вид страницы со списком всех выпускников/студентов;

profile/show.blade.php – вид профиля пользователя;

profile/create.blade.php – вид анкеты выпускника;

profile/edit.blade.php – вид редактирования профиля пользователя;

profile/confirm.blade.php – вид подтверждения удаления профиля;

profile/hidden_profile.blade.php – вид сообщения, о том, что запрашиваемый профиль скрыт;

login.blade.php – вид формы входа в систему;

register.blade.php – вид формы регистрации;

reset.blade.php – вид формы сброса пароля.

Файл routes/web.php содержит маршрутизацию запросов пользователя.

В модуле .env установлены основные настройки приложения: база данных, с которой работает приложение, адрес рассылки электронных писем.

4. Используемые технические средства

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

5. Вызов и загрузка

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

6. Входные и выходные данные

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

ПРИЛОЖЕНИЕ 2

ИСХОДНЫЙ ТЕКСТ ПРОГРАММЫ

1.1 Миграции

1.1.1 2014_10_12_100000_create_password_resets_table.php

```
<?php

use Illuminate\Support\Facades\Schema;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreatePasswordResetsTable extends Migration
{
    public function up()
    {
        Schema::defaultStringLength(191);
        Schema::create('password_resets', function (Blueprint $table) {
            $table->string('email')->index();
            $table->string('token');
            $table->timestamp('created_at')->nullable();
        });
    }
    public function down()
    {
        Schema::dropIfExists('password_resets');
    }
}
```

1.1.2 2018_06_08_084755_create_countries_table.php

```
<?php
use Illuminate\Support\Facades\Schema;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreateCountriesTable extends Migration
{
    public function up()
    {
       Schema::create('countries', function (Blueprint $table) {
            $table->engine = 'InnoDB';
            $table->increments('id');
            $table->string('name');
            $table->timestamps();
        });
    }
    public function down()
    {
        DB::statement('drop table countries cascade');
```

```
        Schema::dropIfExists('countries');
    }
}
```

1.1.3 2018_06_08_093628_create_degrees_table.php

```
<?php
use Illuminate\Support\Facades\Schema;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreateDegreesTable extends Migration
{
    public function up()
    {
        Schema::create('degrees', function (Blueprint $table) {
            $table->engine = 'InnoDB';
            $table->increments('id');
            $table->string('name');
            $table->timestamps();
        });
    }
    public function down()
    {
        Schema::dropIfExists('degrees');
    }
}
```

1.1.4 2018_06_08_095318_create_fields_table.php

```
<?php
use Illuminate\Support\Facades\Schema;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreateFieldsTable extends Migration
{
    public function up()
    {
        Schema::create('fields', function (Blueprint $table) {
            $table->engine = 'InnoDB';
            $table->increments('id');
            $table->string('name');
            $table->timestamps();
        });
    }
    public function down()
    {
        Schema::dropIfExists('fields');
    }
}
```

1.1.5 2018_06_08_095500_create_roles_table.php

```
<?php
use Illuminate\Support\Facades\Schema;
```

```
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreateRolesTable extends Migration
{
    public function up()
    {
       Schema::create('roles', function (Blueprint $table) {
            $table->engine = 'InnoDB';
            $table->increments('id');
            $table->string('name');
            $table->timestamps();
        });
    }
    public function down()
    {
        Schema::dropIfExists('roles');
    }
}
```

1.1.6 2018_06_08_102902_create_users_table.php

```
<?php
use Illuminate\Support\Facades\Schema;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreateUsersTable extends Migration
{
    public function up()
    {
        Schema::create('users', function (Blueprint $table) {
            $table->engine = 'InnoDB';
            $table->increments('id');
            $table->integer('role_id')->unsigned();
            $table->foreign('role_id')->references('id')->on('roles')-
>onDelete('cascade');
            $table->string('surname');
            $table->string('name');
            $table->string('patronymic')->nullable();
            $table->boolean('allow1');
            $table->boolean('allow2');
            $table->boolean('allow3');
            $table->string('email')->unique();
            $table->string('password');
            $table->rememberToken();
            $table->timestamps();
        });
    }
    public function down()
    {
        Schema::dropIfExists('users');
    }
}
```

1.1.7 2018_06_08_103212_create_specialties_table.php

```
<?php
use Illuminate\Support\Facades\Schema;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreateSpecialtiesTable extends Migration
{
    public function up()
    {
        Schema::create('specialties', function (Blueprint $table) {
            $table->engine = 'InnoDB';
            $table->increments('id');
            $table->string('name');
            $table->text('info')->nullable();
            $table->timestamps();
        });
    }
    public function down()
    {
        Schema::dropIfExists('specialties');
    }
}
```

1.1.8 2018_06_09_023626_create_qualifications_table.php

```
<?php
use Illuminate\Support\Facades\Schema;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreateQualificationsTable extends Migration
{
    public function up()
    {
        Schema::create('qualifications', function (Blueprint $table) {
            $table->engine = 'InnoDB';
            $table->increments('id');
            $table->string('name');
            $table->timestamps();
        });
    }
    public function down()
    {
        Schema::dropIfExists('qualifications');
    }
}
```

1.1.9 2018_06_09_090000_create_cities_table.php

```
<?php
use Illuminate\Support\Facades\Schema;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;
```

```
    class CreateCitiesTable extends Migration
    {
        public function up()
        {
            Schema::create('cities', function (Blueprint $table) {
                $table->engine = 'InnoDB';
                $table->increments('id');
                $table->string('name');
                $table->string('coord')->nullable()->default(NULL);
                $table->integer('country_id')->unsigned()->nullable();
                $table->foreign('country_id')->references('id')-
>on('countries')->onDelete('cascade');
                $table->timestamps();
            });

        }
        public function down()
        {
            Schema::dropIfExists('cities');
        }
    }
```

1.1.10 2018_06_10_103646_create_profiles_table.php

```
    <?php
    use Illuminate\Support\Facades\Schema;
    use Illuminate\Database\Schema\Blueprint;
    use Illuminate\Database\Migrations\Migration;

    class CreateProfilesTable extends Migration
    {
        public function up()
        {
            Schema::create('profiles', function (Blueprint $table) {
                $table->engine = 'InnoDB';
                $table->increments('id');
                $table->integer('user_id')->unsigned();
                $table->foreign('user_id')->references('id')->on('users')-
>onDelete('cascade');
                $table->integer('year');
                $table->string('foto')->nullable();
                $table->string('site')->nullable();
                $table->text('info')->nullable();
                $table->integer('city_id')->unsigned()->nullable();
                $table->foreign('city_id')->references('id')->on('cities')-
>onDelete('cascade');
                $table->string('company')->nullable();
                $table->string('position')->nullable();
                $table->integer('specialty_id')->unsigned()->nullable();
                $table->foreign('specialty_id')->references('id')-
>on('specialties')->onDelete('cascade');
                $table->integer('qualification_id')->unsigned()-
>nullable();
                $table->foreign('qualification_id')->references('id')-
>on('qualifications')->onDelete('cascade');
                $table->integer('degree_id')->unsigned()->nullable();
```

```
                $table->foreign('degree_id')->references('id')-
>on('degrees')->onDelete('cascade');
                $table->integer('field_id')->unsigned()->nullable();
                $table->foreign('field_id')->references('id')-
>on('fields')->onDelete('cascade');
                $table->integer('rating')->nullable();
                $table->integer('specialty2_id')->unsigned()->nullable();
                $table->foreign('specialty2_id')->references('id')-
>on('specialties')->onDelete('cascade');
                 $table->integer('qualification2_id')->unsigned()-
>nullable();
                $table->foreign('qualification2_id')->references('id')-
>on('qualifications')->onDelete('cascade');
                $table->integer('degree2_id')->unsigned()->nullable();
                $table->foreign('degree2_id')->references('id')-
>on('degrees')->onDelete('cascade');
                $table->integer('field2_id')->unsigned()->nullable();
                $table->foreign('field2_id')->references('id')-
>on('fields')->onDelete('cascade');
                $table->boolean('allow1');
                $table->boolean('allow2');
                $table->timestamps();
            });
        }
        public function down()
        {
            Schema::dropIfExists('profiles');
        }
    }
```

1.2 Модели

1.2.1 City.php

```
    <?php
    namespace App;
    use Illuminate\Database\Eloquent\Model;
    class City extends Model
    {
        protected $fillable = [
            'name','country_id','coord'
        ];

        protected $hidden = [
            'id'
        ];

        public function setCoord(){
                $response   =json_decode(file_get_contents('https://geocode-
maps.yandex.ru/1.x/?format=json&geocode='.$this->name));
                if                ($response->response->GeoObjectCollection-
>metaDataProperty->GeocoderResponseMetaData->found > 0){
                    $this->coord=$response->response->GeoObjectCollection-
>featureMember[0]->GeoObject->Point->pos;
                }
```

```
                else{
                    dd('Что-то пошло не так...');
                }
                //ФОРМИРУЕМ ГЕО-ПОЗИЦИЮ
                $geometry['type']='Point';
                $arr=explode(' ' ,$this->coord);
                $geometry['coordinates']=[(float)$arr[1],(float)$arr[0]];
                $feature['geometry']=$geometry;

                $profiles=$this->profile;
                $i=0;
                $str='';
                $str2='';
                $check=0;
                foreach($profiles as $profile){
                    if($profile->user->role->name=='student'){continue;}
                    else{
                        $check=1;
                        if($profile->user->allow3==1){
                            $str1="<div><font                    size=3><b><a
href='/profile/".$profile->id."'>".$profile->user->surname.'    '.$profile-
>user->name."</a></b></font></div>";
                            $str2=$str2.$str1;
                        }
                        $i=$i+1;
                    }
                }
                if($check==0){return;}

                if($i==1){$str="<h2>".(string)$i." выпускник</h2>".$str2;}
                else{
                    if($i<5){$str="<h2>".(string)$i."
выпускника</h2>".$str2;}
                    else{$str="<h2>".(string)$i." выпускников</h2>".$str2;}
                }
                $properties['balloonContent']=$str;
                $properties['hintContent']=$this->name;
                $properties['clusterCaption']="выпускник";
                $feature['properties']=$properties;
                $options['preset']='twirl#violetIcon';
                $tosend['options']= $options;
                $tosend['feature']=$feature;

                $filename=public_path().'\js\coords.json';
                $mas=json_decode(file_get_contents($filename),true);

                $check=0;
                foreach($mas as $key=>$location){
                    if
($location['feature']['properties']['hintContent']==$this->name){
                        $mas[$key]=$tosend;
                        $check=1;
                        break;
                    }
                }
                if( $check==0) array_push($mas,$tosend);
                file_put_contents( $filename, PHP_EOL .json_encode($mas));
```

```
    }

    public function country() {
        return $this->belongsTo('App\Country');
    }
    public function profile() {
        return $this->hasMany('App\Profile');
    }

    public function all_users() {
        $profiles=$this->profile;
        $i=0;
        $users=[];
        foreach($profiles as $profile){
            $users[$i]=$profile->user;
            $i=$i+1;
        }
        return $users;
    }
}
```

1.2.2 Country.php

```
<?php
namespace App;
use Illuminate\Database\Eloquent\Model;
class Country extends Model
{
    protected $fillable = [
        'name'
    ];

    protected $hidden = [
        'id'
    ];

    public function city() {
        return $this->hasMany('App\City');
    }
}
```

1.2.3 Degree.php

```
<?php
namespace App;
use Illuminate\Database\Eloquent\Model;
class Degree extends Model
{
    protected $fillable = [
        'name'
    ];

    protected $hidden = [
        'id'
    ];
```

```
    public function profile() {
        return $this->hasMany('App\Profile');
    }
}
```

1.2.4 Field.php

```
<?php
namespace App;
use Illuminate\Database\Eloquent\Model;
class Field extends Model
{
    protected $fillable = [
        'name'
    ];

    protected $hidden = [
        'id'
    ];

    public function profile() {
        return $this->hasMany('App\Profile');
    }
}
```

1.2.5 Profile.php

```
<?php
namespace App;
use Illuminate\Database\Eloquent\Model;
class Profile extends Model
{
   protected $fillable = [
        'user_id',      'year','foto','site','info','city_id','company',
'position',

'specialty_id','qualification_id','degree_id','field_id','rating','specialt
y2_id',
          'qualification2_id','degree2_id','field2_id', 'allow1','allow2'
    ];

    protected $hidden = [
        'id'
    ];

    public function user(){
        return $this->belongsTo('App\User');
    }
    public function city(){
        return $this->belongsTo('App\City');
    }
    public function specialty(){
        return $this->belongsTo('App\Specialty');
    }
    public function qualification(){
        return $this->belongsTo('App\Qualification');
```

```
    }
    public function degree(){
        return $this->belongsTo('App\Degree');
    }
    public function field(){
        return $this->belongsTo('App\Field');
    }
    public function specialty2(){
        return $this->belongsTo('App\Specialty','specialty2_id');
    }
    public function qualification2(){
        return $this->belongsTo('App\Degree','qualification2_id');
    }
    public function degree2(){
        return $this->belongsTo('App\Degree','degree2_id');
    }
    public function field2(){
        return $this->belongsTo('App\Field','field2_id');
    }
}
```

1.2.6 Qualification.php

```
<?php
namespace App;
use Illuminate\Database\Eloquent\Model;
class Qualification extends Model
{
    protected $fillable = [
        'name'
    ];

    protected $hidden = [
        'id'
    ];

    public function profile() {
        return $this->hasMany('App\Profile');
    }
}
```

1.2.7 Role.php

```
<?php
namespace App;
use Illuminate\Database\Eloquent\Model;
class Role extends Model
{
    public function user(){
        return $this->hasMany('App\User');
    }

    public function profile(){
        return $this->hasManyThrough('App\Profile', 'App\User');
    }
}
```

1.2.8 Specialty.php

```
<?php
namespace App;
use Illuminate\Database\Eloquent\Model;
class Specialty extends Model
{
    protected $fillable = [
        'name','info','year'
    ];

    protected $hidden = [
        'id'
    ];

    public function profile() {
        return $this->hasMany('App\Profile');
    }
}
```

1.2.9 User.php

```
<?php
namespace App;
use Illuminate\Notifications\Notifiable;
use Illuminate\Foundation\Auth\User as Authenticatable;

class User extends Authenticatable
{
    use Notifiable;

    protected $fillable = [
        'role_id','surname',
'name','patronymic','allow1','allow2','allow3','email', 'password',
    ];
    protected $hidden = [
        'password', 'remember_token',
    ];

    public function profile(){
        return $this->hasOne('App\Profile');
    }
    public function role(){
        return $this->belongsTo('App\Role');
    }
}
```

1.3 Контроллеры

1.3.1 ProfilesController.php

```
<?php
namespace App\Http\Controllers;

use Illuminate\Http\Request;
```

```
use Auth;
use App\City;
use App\Qualification;
use App\Role;
use App\Specialty;
use App\Degree;
use App\Profile;
use App\Field;
use App\Country;
use App\Http\Controllers\Controller;
use Illuminate\Support\Facades\Validator;

class ProfilesController extends Controller
{
    protected function validator(array $data){
        return Validator::make($data, [
            'year' => 'required|numeric|between:1946,2019',
            'foto' => 'nullable|mimes:jpeg,jpg,bmp,png',
            'site' => 'nullable|string',
            'info'=>'nullable|string',
            'city'=>'nullable|string|max:70|regex:/[А-Яа-яЁё]/u',
            'country'=>'nullable|string|max:100|regex:/[А-Яа-яЁё]/u',
            'company'=>'',
            'position' => '',
            'speciality'  =>  'nullable|string|max:255|regex:/[А-Яа-яЁёA-
Za-z]/u',
            'degree'=>'',
            'field'=>'nullable|string|max:255|regex:/[А-Яа-яЁёA]/u',
            'rating'=>'',
            'degree2'=>'',
            'second_specialty'=>'nullable|string|max:255|regex:/[А-Яа-
яЁёA-Za-z]/u',
            'allow1'=>'accepted',
            'allow2'=>'int',
        ]);
    }

    public function create(){
        return view('profile.create');
    }


    protected function make_data(Request $request){

            if($request->hasFile('foto')) {
                $file = $request->file('foto');
                $foto=$file->move(public_path().'\images\fotos',
'user'.Auth::user()->id.'.jpg');
                $data['foto']=$foto;
            }
            else $data['foto']=null;


        $validator=$this->validator($request->input());

        if ($validator->fails()) {
            return redirect('profile/create')
```

```
                ->withErrors($validator)
                ->withInput();
        }
        $inp=$request->input();

        $data['user_id']=Auth::user()->id;
        $data['year']=$inp['year'];
        $data['site']=$inp['site'];
        $data['info']=$inp['info'];
   //-----------------------------------------------------------------
        if (is_null($inp['city'])){
            $data['city_id']=NULL;
        }
        else{
            $city=City::firstOrCreate(['name'=>$inp['city']]);
            $data['city_id']=$city->id;
        }
        if (is_null($inp['country'])){}
        else{
            $country=Country::firstOrCreate(['name'=>$inp['country']]);
            $city->country_id=$country->id;
            $city->save();
        }
   //-----------------------------------------------------------------
        $data['company']=$inp['company'];
        $data['position']=$inp['position'];
   //-----------------------------------------------------------------
        if (is_null($inp['specialty'])){
            $data['specialty_id']=NULL;
        }
        else{

$specialty=Specialty::firstOrCreate(['name'=>$inp['specialty']]);
            $data['specialty_id']=$specialty->id;
        }
   //-----------------------------------------------------------------
        if (is_null($inp['qualification'])){
            $data['qualification_id']=NULL;
        }
        else{

$qualification=Qualification::firstOrCreate(['name'=>$inp['qualification']]
);
            $data['qualification_id']=$qualification->id;

        }
        //-----------------------------------------------------------------
-
        if (is_null($inp['degree'])){
            $data['degree_id']=NULL;
        }
        else{
            $degree=Degree::firstOrCreate(['name'=>$inp['degree']]);
            $data['degree_id']=$degree->id;
        }
   //-----------------------------------------------------------------
        if (is_null($inp['field'])){
```

```
            $data['field_id']=NULL;
        }
        else{
            $field=Field::firstOrCreate(['name'=>$inp['field']]);
            $data['field_id']=$field->id;
        }
    //-----------------------------------------------------------------
        if (!array_key_exists('rating',$inp)) $data['rating']=NULL;
        else $data['rating']=$inp['rating'];
    //-----------------------------------------------------------------
        if (is_null($inp['specialty2'])){
            $data['specialty2_id']=NULL;
        }
        else{

$specialty2=Specialty::firstOrCreate(['name'=>$inp['specialty2']]);
            $data['specialty2_id']=$specialty2->id;
        }
    //-----------------------------------------------------------------
        if              (is_null($inp['qualification2'])             ||
$inp['qualification2']==''){
            $data['qualification2_id']=NULL;
        }
        else{

$qualification2=Qualification::firstOrCreate(['name'=>$inp['qualification2'
]]);
            $data['qualification2_id']=$qualification2->id;
        }

            $data['degree2_id']=NULL;
            $data['field2_id']=NULL;

    //-----------------------------------------------------------------
        if (!array_key_exists('allow1',$inp)) $data['allow1']=0;
        else $data['allow1']=1;
        if (!array_key_exists('allow2',$inp)) $data['allow2']=0;
        else $data['allow2']=1;

        return $data;
    }

    protected function store(Request $request){
        $data=$this->make_data($request);
        $profile=Profile::create($data);

         $inp=$request->input();
        if (is_null($inp['city'])){}
        else{
            $city=City::where('name',$inp['city'])->first();
            $city->setCoord();
        }
        return redirect('profile/'.$profile->id);
    }

    public function update(Profile $profile,Request $request){
        $data=$this->make_data($request);
```

```
            $profile->update($data);

            $inp=$request->input();
            if (is_null($inp['city'])){}
            else{
                $city=City::where('name',$inp['city'])->first();
                $city->setCoord();
            }

            return redirect('profile/'.$profile->id);
         }

        public function edit(Profile $profile) {
             return view('profile.edit',compact('profile'));
        }

        public function show(Profile $profile){
            if(Auth::check())                                                return
view('profile.show',compact('profile'));
            else{
                if ($profile->allow2 || $profile->user==Auth::user()) return
view('profile.show',compact('profile'));
                else return view('profile.hidden_profile');
            }
        }

         public function destroy(Profile $profile){
             $profile->delete;
             return view('profile.show');
         }

         public function confirm (Profile $profile){
             return view('profile.confirm',compact('profile'));
         }

        public function add_foto(Profile $profile, Request $request){
            if($request->hasFile('foto')) {
                $file = $request->file('foto');
                $foto=$file->move(public_path().'\images\fotos',
'user'.Auth::user()->id.'.jpg');
                $profile->foto=$foto;
                $profile->save();
            }
            return redirect('profile/'.$profile->id);
        }

        public function show_all_grads(){
            $profiles=Role::where('name','=','graduate')->first()->profile-
>sortByDesc('year');
            return view('all_grad',compact('profiles'));
        }
        public function show_all_studs(){
            $profiles=Role::where('name','=','student')->first()->profile-
>sortBy('year');
            return view('all_grad',compact('profiles'));
        }
```

```
    }
```

1.3.2 RegisterController.php

```
<?php

namespace App\Http\Controllers\Auth;

use App\User;
use App\Role;
use App\Http\Controllers\Controller;
use Illuminate\Support\Facades\Validator;
use Illuminate\Foundation\Auth\RegistersUsers;

class RegisterController extends Controller
{
    protected function validator(array $data)
    {
        return Validator::make($data, [
            'surname' => 'required|string|max:70|regex:/[А-Яа-яЁё]/u',
            'name' => 'required|string|max:70|regex:/[А-Яа-яЁё]/u',
            'patronymic'     =>     'nullable|string|max:70|regex:/[А-Яа-
яЁё]/u',
            'allow1'=>'int',
            'allow2'=>'accepted',
            'allow3'=>'int',
            'email' => 'required|string|email|max:100|unique:users',
            'password' => 'required|string|min:6|confirmed',
        ]);
    }

    /**
     * Create a new user instance after a valid registration.
     *
     * @param  array  $data
     * @return \App\User
     */
    protected function create(array $data)
    {
        if (!array_key_exists('allow1',$data)) $data['allow1']=0;
        if (!array_key_exists('allow2',$data)) $data['allow2']=0;
        if (!array_key_exists('allow3',$data)) $data['allow3']=0;
        $role = Role::where('name', '=',$data['role'] )->get()->first();
        return User::create([
            'role_id' => $role->id,
            'surname' => $data['surname'],
            'name' => $data['name'],
            'patronymic' => $data['patronymic'],
            'allow1' => $data['allow1'],
            'allow2' => $data['allow2'],
            'allow3' => $data['allow3'],
            'email' => $data['email'],
            'password' => bcrypt($data['password']),
        ]);
    }
}
```

1.4 Виды

1.4.1 all_grad.blade.php

```
@extends('layouts.app1')
@section('content')
    <div  class="conteiner"  style="display:flex;  flex-direction:row;
flex-wrap: wrap;">
        <?php $lastyear=1000 ?>
        @foreach($profiles as $profile)
            @if($profile->year !=$lastyear)
            <div  style="display:flex;  flex-direction:  column;  margin:
20px;" id="{{ $profile->year }}">
                <span class="vvod-text">{{ $profile->year }}</span>
                @foreach($profiles->where('year','=',$profile->year)  as
$grad)
                        <a      class="s1"      href="/profile/{{$grad-
>id}}">{{$grad->user->surname}} {{$grad->user->name}}</a>
                <?php $lastyear=$profile->year ?>
                @endforeach
            </div>
            @endif

        @endforeach

    </div>
@endsection
```

1.4.2 my_welcome.php

```
<!doctype html>
<html lang="{{ app()->getLocale() }}">
    <head>
        <meta charset="utf-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge">
        <meta   name="viewport"   content="width=device-width,   initial-
scale=1">

        <title>Выпускники ПриМа</title>

        <!-- Fonts -->
        <link
href="https://fonts.googleapis.com/css?family=Raleway:100,600"
rel="stylesheet" type="text/css">

        <!-- Styles -->
        <link href="{{ asset('css/my_welcome.css') }}" rel="stylesheet">
        <script src="http://code.jquery.com/jquery-1.8.3.js"></script>
        <script src="{{ asset('js/coords.json') }}"></script>
        <script                    src="https://api-maps.yandex.ru/2.0-
stable/?load=package.full&lang=ru-RU" type="text/javascript"> </script>
        <script src="{{ asset('js/geo.js') }}"></script>
    </head>

    <body>
  <div class="header">
```

```
            <div class="contacts">
                <div class="susu contacts-cell">
                    <div><a           href="https://www.susu.ru/ru"><img
class="logo-susu" src={{asset('images/susu.png')}} alt="" /></a></div>
                    <div             class="text-style-1">Южно-Уральский
<br/>государственный</br>университет</div>
                </div>
                <div     class="prima_text     text-style-1     contacts-
cell">Кафедра прикладной математики<br/>и программирования</div>
                <div                                class="contacts-cell"><a
href="http://prm.susu.ru/"><img                                class="logo-prima"
src={{asset('images/prima.png')}} alt="" /></a></div>
                @if (Route::has('login'))
                    <div class="reg contacts-cell">
                        @if (Auth::check())
                            @if(isset(Auth::user()->profile))
                                <form                        id="my_profile"
action="/profile/{{Auth::user()->profile->id}}" style="display: none;">
                                    {{ csrf_field() }}
                                </form>
                            @else
                                <form                        id="my_profile"
action="/profile/create" style="display: none;">
                                    {{ csrf_field() }}
                                </form>
                            @endif
                            <button    type='submit'    form="my_profile"
class="orange-button"><span class="text">МОЙ ПРОФИЛЬ</span></button>
                            <button    onClick='location.href="/logout";
event.preventDefault();    document.getElementById("logout-form").submit()'
class="orange-button"><span class="text">ВЫЙТИ</span></button>
                            <form      id="logout-form"      action="{{
url('/logout') }}" method="POST" style="display: none;">
                                {{ csrf_field() }}
                            </form>
                        @else
                            <button  onclick='location.href  =  "/login"'
class="orange-button"><span class="text">ВОЙТИ</span></button>
                            <button       onclick='location.href       =
"/register"'                                   class="orange-button"><span
class="text">ПРИСОЕДИНИТЬСЯ</span></button>
                        @endif
                    </div>
                @endif
                <div class="knopki contacts-cell">
                    <a      href="#"><img      src={{asset('images/icon-
phone.png')}} alt="" /></a>
                    <a      href="#"><img      src={{asset('images/icon-
mail.png')}} alt="" /></a>
                    <a              href="https://vk.com/susu_IETN"><img
src={{asset('images/icon-vk.png')}} alt="" /></a>
                </div>
            </div>

            <div class="menu">
                <ul>
                    <li><a href="#">ОБ УНИВЕРСИТЕТЕ</a></li>
```

```
                        <li><a href="#">О КАФЕДРЕ</a></li>
                        <li><a href="/all_studs">СТУДЕНТЫ</a></li>
                        <li><a href="/all_grad">ВЫПУСКНИКИ</a></li>
                        <li><a href="/map">ИНТЕРАКТИВНАЯ КАРТА</a></li>

                    </ul>
                </div>
                <div  class="head2"  style="width:  100%;  background-color:
rgba(128,128,128,0.5); height: calc(100vh - 0.1*100vh - 1px);">
                <div class="head2 wrapper">
                    <div class="head_text">
                        <h1>ВЫПУСКНИКИ</h1>
                        <h2>Единая информационная система</h2>
                    </div>

                    <div class="inform">
                        <div class="contacts-cell">
                            <div class="inf">
                                <img                                class="hat"
src={{asset('images/hat.png')}} alt=""/>
                                <p class="inf-text">2 440 специалистов</p>
                                <p class="podtext">Кафедра выпустила более 2
440 специалистов</p>
                            </div>
                        </div>
                        <div class="contacts-cell">
                            <div class="inf">
                                <img      src={{asset('images/univer.png')}}
alt=""/>
                                <p class="inf-text">45 лет</p>
                                <p    class="podtext">Кафедра    образована
приказом<br/>ректора ЧПИ от 1973 года</p>
                            </div>
                        </div>
                        <div class="contacts-cell">
                            <div class="inf">
                                <img    src='{{asset('images/globus.png')}}'
alt=""/>
                                <p class="inf-text">38 выпусков</p>
                                <p    class="podtext">1980    год    -    1
выпуск<br/>2017 год - 38 выпуск </p>
                            </div>
                        </div>
                    </div>
                </div>
                </div>
            </div>
    <!-- --------КАРТА------------ -->
            <div  class="grey big-block">
                <div id="offers1" class="wrapper" class="grey">
                    <a id="map"></a>
            <h3 class="big-text">МИР ВЫПУСКНИКОВ КАФЕДРЫ</h3>
            <div id="YMapsID" style="width: 100%; height: 400px;"></div>
                </div>
      </div>
      <div  class="grey big-block">
                <div id="offers1" class="wrapper" class="grey">
```

```
                    <a id="alumni"></a>
            <h3 class="big-text">Наши выпускники</h3>
            <div id="YMapsID" style="width: 100%; height: 400px;"></div>
                </div>
      </div>

   <!-- ---------------------------- -->
            <button            onClick='location.href="/update_from_excel";'
class="orange-button"><span class="text">ЗАГРУЗИТЬ ДАННЫЕ</span></button>
            <div class="footer">
            </div>
            <div class="footer1">

            </div>
        </body>
   </html>
```

1.5 Маршрутизатор routes.php

```
   <?php

   Route::get('/', function () {
       return view('my_welcome');
   });

   Auth::routes();

   Route::get('profile/create', 'ProfilesController@create');
   Route::get('profile/{profile}', 'ProfilesController@show');
   Route::post('profile/create', 'ProfilesController@store');
   Route::get('profile/{profile}/edit', 'ProfilesController@edit');
   Route::patch('profile/{profile}', 'ProfilesController@update');
   Route::get('profile/{profile}/confirm', 'ProfilesController@confirm');
   Route::delete('profile/{profile}', 'ProfilesController@destroy');
   Route::post('profile/{profile}/add-foto',
'ProfilesController@add_foto');

   Route::get('/home', function () {
       return view('home');
   });
   Route::get('/all_grad', 'ProfilesController@show_all_grads');
   Route::get('/all_studs', 'ProfilesController@show_all_studs');
   Route::get('/map', function(){
       return view('map');
   });
   Route::get('/update_from_excel',
'UpdateDataController@updateFromExcel');
```