

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет
вычислительной техники

Кафедра
МОиПЭВМ

Направление подготовки 09.03.02 «Информационные системы и технологии»

Направленность Информационные системы и технологии

БАКАЛАВРСКАЯ РАБОТА

на тему

Приложение «Аудиторный гид» по
университетскому кампусу

Студент _____ Смирнова Ангелина Евгеньевна
(подпись, дата) (ФИО полностью)

Руководитель _____ Шибанов С.В.
(подпись, дата) (фамилия, инициалы)

Нормоконтролёр _____ Такташкин Д.В.
(подпись, дата) (фамилия, инициалы)

Работа допущена к защите (протокол заседания кафедры от _____ № _____)

Заведующий кафедрой _____ Козлов А.Ю.
(подпись) (фамилия, инициалы)

Работа защищена с отметкой _____ (протокол заседания ГЭК от _____ № _____)

Секретарь ГЭК _____ Попова Н.А.
(подпись) (фамилия, инициалы)

Пенза, 2024

Федеральное государственное бюджетное образовательное учреждение
высшего образования
ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет

Вычислительной техники

Кафедра

МО и ПЭВМ

«Утверждаю»
Заведующий кафедрой
А.Ю. Козлов «25»
декабря 2023 г

**ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
БАКАЛАВРА**

1. Студент Смирнова Ангелина Евгеньевна гр. 20ВИ1 факультета ВТ
<Фамилия, имя, отчество полностью>

2. Тема работы Приложение «Аудиторный гид» по университетскому кампусу

Тема утверждена приказом ПГУ № _____ от " _____ " _____ 20 _____

3. Руководитель работы к.т.н., доцент кафедры «МО и ПЭВМ» Шибанов С.В.

4. Задание на работу (назначение разработки, исходные данные и т.п.)

Назначение: приложения для помощи в ориентации внутри корпусов, осуществление
поиска кабинета по фильтрам, представлении подробной информации о кабинете и
кафедрах.

Средства выполнения: Операционная система Windows

Средства разработки: IntelliJ IDEA 2023.3.4, PostgreSQL 15

Язык программирования: Java

Технология разработки: ООП, UML

5. Перечень подлежащих разработке вопросов

1) Анализ предметной области и постановка задачи на разработку

2) Сравнительный анализ приложений аналогов

3) Разработка требований к приложению пользователя «Аудиторный гид»

4) Проектирование программных средств приложения «Аудиторный гид»

5) Разработка программных средств приложения «Аудиторный гид»

6) Контроль качества программного обеспечения приложения «Аудиторный гид»

7) Оформление пояснительной записки

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Анализ предметной области и требований к системе	10%	08.04.2024 – 13.04.2024	
2	Проектирование программных средств	20%	15.04.2024 – 25.04.2024	
3	Реализация программных средств	30%	26.04.2024 – 12.05.2024	
4	Тестирование программных средств	20%	13.05.2024 – 22.05.2024	
5	Оформление пояснительной записки	20%	23.05.2024 – 03.06.2024	

Дата выдачи задания «21» декабря 2023г.

Руководитель бакалаврской работы _____
(подпись, дата)

Задание к исполнению принял студент _____
(подпись, дата)

Работу к защите допустить
Декан ФВТ _____ Л.Р. Фионова
(подпись, дата)

Реферат

Пояснительная записка содержит 144 листа, 32 таблицы, 60 рисунков, 17 использованных источников и 7 приложений.

УНИВЕРСИТЕТСКИЙ КАМПУС, КАФЕДРА, ФАКУЛЬТЕТ, КОРПУС, АУДИТОРНЫЙ ГИД, ПОИСК, ФИЛЬТР, БАЗА ДАННЫХ, SQL, JAVA, SPRING, СЕРВЕР, ФУНКЦИОНАЛЬНОЕ ТЕСТИРОВАНИЕ, МЕТРИКИ КОДА.

Объект исследования: приложение пользователя для навигации внутри корпусов университетского кампуса.

Цель разработки: реализовать приложение для помощи в ориентации внутри корпусов и на территории кампуса, а также осуществление поиска кабинета по фильтрам, представлении подробной информации о кабинете и кафедрах, расположенных в корпусах.

Технологии разработки: java, css, html, IntelliJ IDEA, PostgreSQL, Spring, TomCat.

Результаты работы: спроектированное и разработанное приложение «Аудиторный гид» по университетскому кампусу.

					ПГУ 09.03.02 – 8КР201.16 ПЗ										
Изм.	Лист	№ докум.	Подпись	Дата	Приложение «Аудиторный гид» по университетскому кампусу				Лит.		Лист	Листов			
Разраб.		Смирнова А.Е.													
Провер.		Шибанов С.В.										4		144	
Реценз.									группа 20ВИ1						
Н. Контр.		Такташкин Д.В.													
Утверд.		Козлов А.Ю.													

Содержание

Введение.....	8
1 Анализ предметной области и постановка задачи на разработку приложения «Аудиторный гид»	10
1.1 Анализ структуры университетского кампуса и задач поиска помещений и аудиторий.....	10
1.1.1 Структура университетского кампуса на примере Пензенского государственного университета	10
1.1.2 Классификация помещений и аудиторий в университетском кампусе.....	13
1.1.3 Задачи поиска, просмотра и выборки помещений и аудиторий	15
1.2 Сравнительный анализ приложений-аналогов	16
1.2.1 Цели и задачи сравнительного анализа	16
1.2.2 Выбор приложений-аналогов для анализа	18
1.2.3 Обзор приложений-аналогов	19
1.2.4 Результаты сравнительного анализа	22
1.3 Постановка задачи на разработку сервиса	24
1.3.1 Цели разработки и назначение сервиса	24
1.3.2 Классы и роли пользователей сервиса.....	24
1.3.3 Автоматизируемые бизнес-процессы	26
1.4 Планирование процесса разработки сервиса	29
2 Проектирование приложения «Аудиторный гид».....	32
2.1 Моделирование функциональных возможностей приложения	32
2.2 Разработка концептуальной модели системы аудиторного гида университета.....	36
2.3 Разработка программной архитектуры приложения.....	38
2.4 Разработка прототипа пользовательского интерфейса	39
2.5 Разработка требований к аппаратно-программному окружению	42
2.6 Обоснование выбора средств разработки.....	43

2.6.1 Обоснование выбора средств автоматизации	43
2.6.2 Обоснование выбора инструментальных средств разработки программного обеспечения.....	44
2.6.3 Обоснование выбора языковых средств разработки.....	45
2.6.4 Обоснование выбора средств управления данными	47
3 Разработка программного обеспечения приложения «Аудиторный гид»	49
3.1 Разработка базы данных.....	49
3.2 Разработка пользовательского интерфейса.....	53
3.3 Разработка программного обеспечения.....	59
4 Контроль качества программного обеспечения приложения «Аудиторный гид»	66
4.1 Планирование процесса контроля качества программного обеспечения..	66
4.1.1 Мероприятия по контролю качества программного обеспечения....	66
4.1.2 Обоснование выбора модели тестирования	67
4.1.3 Обоснование выбора метрик программного кода	69
4.2 Тестирование программных средств	70
4.2.1 Тестирование черным ящиком	70
4.2.2 Тестирование белым ящиком	73
4.3 Оценка качества программного кода на основе метрик	78
4.3.1 Метрика SLOC.....	78
4.3.2 Метрика Холстеда.....	78
4.3.3 Метрика Чепина	80
Заключение	82
Список используемой литературы	84
Приложение А (справочное) Глоссарий	86
Приложение Б (рекомендуемое) Рисунки ВКР	88
Приложение В (рекомендуемое) Сценарии процессов	92
Приложение Г (рекомендуемое) Атрибуты сущностей базы данных	95
Приложение Д (рекомендуемое) Шаблоны пользовательского интерфейса....	101
Приложение Е (рекомендуемое) Код разработки базы данных	105

Приложение Ж (рекомендуемое) Листинг программы.....	112
---	-----

Введение

Университетские территории и построенные здания корпусов порой имеют сложную и нестандартную архитектуру. У студентов первого курса, преподавателей и гостей университета возникают проблемы с ориентацией на территории и внутри зданий.

Аудиторный гид – это графическая иллюстрация того, как расположены аудитории, столовые, гардеробы, туалеты и залы конференций в уже построенных корпусах ПГУ. Как и любой другой план, создание плана здания имеет несколько преимуществ. Начнем с того, что планы помогут вам визуализировать здание в целом еще до того, как вы в него попадете. Этот процесс экономит время блужданий по корпусам в целях нахождения нужного кабинета. Также наглядное представление схемы помогает понять как расположены объекты относительно друг друга.

Целью выпускной квалификационной работы является разработка приложения для помощи в ориентации внутри корпусов и на территории кампуса, а также осуществление поиска кабинета по фильтрам, представлении подробной информации о кабинете и кафедрах, расположенных в корпусах.

Для решения поставленной цели необходимо выполнить ряд задач:

- анализ структуры университетского кампуса и задач поиска помещений и аудиторий;
- обзор и сравнение приложений-аналогов;
- проектирование и разработка программного обеспечения приложения «Аудиторный гид»;
- разработка и заполнение данными базу данных;
- тестирование приложения «Аудиторный гид».

В ходе выполнения дипломной работы проведены:

- исследование предметной области университетский кампус;
- разработка и заполнение базы данных верными данными;
- разработка и тестирование веб приложения для конечного пользователя.

Разработанное веб-приложение поможет легко ориентироваться на территории университета и в корпусах. В приложении можно будет найти информацию о корпусах, планах эвакуации в них, кафедрах, расположенных в корпусах, а также подробную информацию и расположение каждого кабинета.

Приложение можно будет внедрить в официальный сайт Пензенского государственного университета pnzgu.ru, что позволит расширить функционал и применение сайта. Также оно универсальное и при заполнении базы данных информацией о другом университете может вписаться в любой другой сайт университета Российской Федерации с подобной структурой ведения образования.

В процессе работы над ВКР были опубликованы две статьи «Сравнительный анализ бесплатных программ по генерации тестовых данных в БД» и «Сравнительный анализ бесплатных программ по нагрузке базы данных», выполнены доклады на двенадцатом международном конкурсе научных работ в направлении технические науки и восьмой международной научно-практической конференции студентов и молодых ученых «Модели, системы, сети в технике».

1 Анализ предметной области и постановка задачи на разработку приложения «Аудиторный гид»

1.1 Анализ структуры университетского кампуса и задач поиска помещений и аудиторий

1.1.1 Структура университетского кампуса на примере Пензенского государственного университета

Пензенский государственный университет представляет из себя несколько территорий, разделенных автомобильными дорогами. Все здания расположены в пешей доступности друг от друга.

Были найдены и изучены технические паспорта корпусов, взятые из архива ПГУ. В них предоставлены сведения о местоположении корпуса, поэтажные планы с размерами и описанием помещений на них, информация о площади и объеме строений, а также технические описания основных строений. Перечень найденных технических паспортов представлен таблица 1.

Таблица 1 – Документы по корпусам

Название ТП здания	Дата утверждения	Количество страниц	Адрес корпуса
Учебный корпус №1	31.08.2020	17	г. Пенза Первомайский р-н улица Красная 40
Учебный корпус №2	13.01.2022	13	г. Пенза Первомайский р-н улица Чкалова 57
Учебный корпус №3	05.06.2012	15	г. Пенза Первомайский р-н улица Красная 46
Учебный корпус №4	03.04.2012	16	г. Пенза Первомайский р-н улица Красная 46
Учебный корпус №5	11.05.2018	17	г. Пенза Первомайский р-н улица Красная 40
Учебный корпус №6	28.08.2012	8	г. Пенза Первомайский р-н улица Красная 40
Учебный корпус №7	30.03.2021	22	г. Пенза Первомайский р-н улица Лермонтова 26
Учебный корпус №8	01.03.2022	26	г. Пенза Первомайский р-н улица Красная 46
Учебный корпус №9	11.05.2018	24	г. Пенза Первомайский р-н улица Чкалова 68
Учебный корпус №10	22.02.2012	18	г. Пенза Ленинский р-н улица Лермонтова 3

Продолжение таблицы 1

Название ТП здания	Дата утверждения	Количество страниц	Адрес корпуса
Учебный корпус №11	01.07.2011	19	г. Пенза Ленинский р-н улица Лермонтова 37
Учебный корпус №12	11.01.2012	18	г. Пенза Ленинский р-н улица Лермонтова 37
Учебный корпус №13	11.01.2012	17	г. Пенза Ленинский р-н улица Лермонтова 37
Учебный корпус №14	11.01.2012	18	г. Пенза Ленинский р-н улица Лермонтова 37
Учебный корпус №15	11.01.2012	17	г. Пенза Ленинский р-н улица Лермонтова 37
Учебный корпус №16	11.01.2012	12	г. Пенза Ленинский р-н улица Карла Маркса 4
Учебный корпус №17	14.05.2008	12	г. Пенза Ленинский р-н улица Чкалова 68/Красная 31
Учебный корпус №18	31.01.2020	14	г. Пенза Ленинский р-н улица Лермонтова 37

Учебный корпус № 1 – это главное здание университета, где расположен ректорат и проректорат, административные офисы, службы университета, а также учебные кабинеты.

В Пензенском государственном университете девятнадцать учебных корпусов, где проходят занятия по разным факультетам и направлениям. В корпусах присутствуют буфеты, а в некоторых оборудованы библиотеки.

На территории семь общежитий для иногородних студентов, оснащенных необходимыми удобствами.

Для занятия спортом построены спортивные сооружения, тренажерный зал, бассейн, теннисная площадка, лыжная база и стадион.

Отдельное здание представляет из себя столовую.

Несколько медицинских центров для оказания медицинской помощи студентам, преподавателям и университетским работникам.

Помимо зданий на территории предназначены места для отдыха, парковые зоны, мемориал, памятник и парковочные места для студентов и персонала.

К прочим заданиям относятся склады, гаражи, теплица, сарай, ангар, котельная, контрольно-пропускные пункты и другие.

На территории Пензенского государственного университета есть все необходимое для обеспечения безопасной учебы.

Университет объединяет в себе несколько факультетов, объединенных по общим областям знаний. Такие как естественные, гуманитарные, технические, социальные науки и т. д. Факультеты в свою очередь поделены на кафедры, которые специализируются на определенных дисциплинах и направлениях. Кафедры включают в себя группу преподавателей, кафедральное руководство, занимающееся вопросами и делами студентов и преподавателей своей кафедры. Кафедры предлагают различные направления подготовки студентов в рамках своей специализации. Направления подготовки могут включать в себя бакалавриат, магистратуру, аспирантуру и другие образовательные программы, предназначенные для углубленного изучения конкретных областей знаний и подготовки специалистов в соответствующей области.

Пензенский государственный университет поделен на следующие основные образовательные структуры:

- Политехнический институт;
- Педагогический институт им. В.Г. Белинского;
- Медицинский институт;
- Институт физической культуры и спорта;
- Институт непрерывного образования;
- Институт международного сотрудничества;
- Институт экономики и управления;
- Юридический институт;
- Военный учебный центр.

Была рассмотрена подробная информация о политехническом институте. Он включает в себя три факультета: факультет промышленных технологий, электроэнергетики и транспорта, факультет информационных технологий и

электроники и факультет вычислительной техники, которые в свою очередь поделены на кафедры.

Кафедры факультета вычислительной техники следующие:

- кафедра «Компьютерные технологии»;
- кафедра «Математика и суперкомпьютерное моделирование»;
- кафедра «Системы автоматизированного проектирования»;
- кафедра «Высшая и прикладная математика»;
- кафедра «Информационно-вычислительные системы»;
- кафедра «Вычислительная техника»;
- кафедра «Информационное обеспечение управления и производства»;
- кафедра «Математическое обеспечение и применение ЭВМ».

По данным, взятым с официального сайта <https://pnzgu.ru/sveden/struct> о структуре общеуниверситетской организации обучения, в ПГУ 8 факультетов и 91 кафедра [1]. В эти подсчеты не входят направления подготовки студентов в многопрофильном колледже и военном учебном центре.

Таким образом, при такой системе университет обучает специалистов в различных областях. Разнообразие учебных программ предоставляет студентам выбрать себе направление в соответствии с их интересами и предпочтениями.

1.1.2 Классификация помещений и аудиторий в университетском кампусе

Все помещения и аудитории в корпусе можно классифицировать по различным критериям. Например, по расположению в здании, по оборудованию, занимаемой территории или по назначению.

Самая распространенная классификация учебных аудиторий на лекционные и практические. Лекционные кабинеты, как правило, оснащены досками (меловыми, маркерными) или возможностью показа лекционного материала с помощью проектора. В таких аудиториях должно помещаться большое количество студентов, для проведения, например, потоковых лекций.

Для преподавателя должно быть отведено специальное место, с которого всем студентам будет слышно слушать лекцию.

Кабинеты для практических занятий обычно обладают средними размерами. В них должно располагаться специальное оборудование, необходимо для проведения занятия. В таких аудиториях должна учитываться техника безопасности при планировании расположения той или иной практики.

Практические аудитории можно подразделить на компьютерные и лабораторные классы. Лабораторные кабинеты оборудованы для проведения опытов и исследований, а компьютерные – специальным софтверным обеспечением и доступом в интернет с использованием компьютерных технологий.

В Пензенском государственном университете для деканатов и кафедр выделены специальные аудитории. В кафедральной аудитории преподаватели могут готовятся к учебным занятиям и отдыхать между парами. В кабинетах, где расположены деканаты, работает руководство, документоведы, замы, помощники и другие.

Спортивные помещения еще один тип классификации. Они включают в себя тренажерные залы, бассейны, спортивные площадки, спортивные залы.

Многие корпуса в своем составе имеют библиотеки, где царит тишина и спокойная обстановка. В таких помещениях студенты могут изучать книги, научные статьи, электронные ресурсы, предоставляемые этой библиотекой.

В современном мире университеты в своих корпусах стараются планировать и неформальные площадки. В таких помещениях студенты могут коммуницировать и знакомиться друг с другом, проводить мастер-классы, отдыхать и готовиться к занятиям в комфортной обстановке.

Таким образом, можно выделить следующие типы помещений:

- лекционные;
- практические;
- компьютерные;
- лабораторные;

- спортивные;
- библиотеки;
- столовые;
- для отдыха;
- для деканата;
- кабинеты для работников университета разных должностей.

Классификация помещений и аудиторий может меняться в зависимости от направлений подготовки, финансовых возможностей и других факторов. Важно, чтобы помещения и аудитории каждого типа соответствовали потребностям студентов и преподавателей, обеспечивая комфортные условия для успешного обучения и работы.

1.1.3 Задачи поиска, просмотра и выборки помещений и аудиторий

Студенты, преподаватели и гости университета сталкиваются с рядом проблем в нахождении нужных помещений и аудиторий. Ниже приведен ряд задач, решаемых быстрым поиском:

- при планировании учебного расписания необходимо по различным критериям находить аудитории, которые будут специализироваться на определенных требованиях поиска;
- при проведении конференций, семинаров, мероприятий требуется найти свободное помещение, подходящее под планируемое мероприятие;
- для проведения консультаций преподавателям необходимо подобрать аудиторию;
- проведения опытов и исследований требует оснащение в кабинете нужного оборудования;
- для проведения экзаменов и самостоятельных работы нужно подобрать подходящие аудитории.

Поиск помещений и аудиторий может решить перечисленные проблемы.

Просмотр и отображение информации о кабинетах может решить ряд проблем пользователей и иметь следующие преимущества:

- отображение информации в отсортированном виде;
- просмотр всех кабинетов университета;
- просмотр всех корпусов;
- получение подробной информации о кабинете, включающее в себя различные данные;
- фильтрация кабинетов по определенному критерию для нахождения наилучшего варианта;
- отображение планов этажей, где расположен кабинет.

При выборке кабинета пользователь может выбирать и сравнивать различные характеристики кабинетов.

Так, можно грамотно спланировать учебный процесс и быть осведомленным о различных кабинетах в корпусе.

1.2 Сравнительный анализ приложений-аналогов

1.2.1 Цели и задачи сравнительного анализа

При сравнительном анализе сравниваются одинаковые, в данном случае, приложения, применяемые в одном виде деятельности. Анализ чаще всего используют для сравнения компаний, отдельных направлений бизнеса, разрабатываемых продуктов. По окончании исследования выявляются сильные и слабые стороны сравниваемых продуктов, с целью наилучшей разработки своего продукта. После проведения анализа формируются цели и задачи своей разработки.

Самая главная задача анализа разработка стратегии развития и направления продукта с учетом всех факторов, которые влияют на разработку. Это помогает оценить преимущества и недостатки сравниваемых продуктов, оценить влияние внешней среды и найти эффективные пути развития своего продукта.

Для проведения сравнительного анализа не требуется специальных знаний. При проведении анализа потребуются выявить критерии сравнения, выбрать приложения-аналоги, проанализировать и сформулировать результаты сравнительного анализа.

После проведения сравнительного анализа будет выявлен список сильных сторон разрабатываемого продукта. Эти характеристики будут давать преимущество перед другими приложениями-аналогами.

В данной выпускной квалификационной работе будут применены распространенные критерии анализа для сравнения приложений аналогов. Каждый критерий описан по его значению в сравнительном анализе.

Функциональность – какие функции и возможности предоставляет приложение.

Актуальностью данных – проверка информации, предоставленной в приложении с реальными данными, а также частота обновления информации для предоставления наиболее актуальных данных.

Совместимость с устройствами – поддерживается ли данная разработка на разных операционных системах.

Стоимость и доступность – проверка бесплатных версий, наличие пробных версий, для каких пользователей возможно работы с данным приложением.

Удобство использования – возможно получится найти рейтинг приложения среди других или изучить реальные отзывы пользователей для определения уровня удовлетворенности приложения.

Выбор критериев для анализа будет влиять на определение лучшего варианта среди всех рассматриваемых приложений в сравнительном анализе. А успех разрабатываемого приложения будет зависеть от конечных результатов, на которые оказывает влияние уровень соответствия выбранных критериев требованиям и целям разработки.

1.2.2 Выбор приложений-аналогов для анализа

Были рассмотрены веб-страницы, разработанные университетами для своих официальных сайтов. Список выбранных для анализа приложений представлен в таблице 2.

Таблица 2 – Список программ и их описание

№	Название университета	Тип приложения	Компания разработки	Ссылка для скачивания
1	Российская академия народного хозяйства и государственной службы при Президенте Российской Федерации	Веб-сайт, мобильное приложение	Разработка Президентской академии	https://www.ranepa.ru/ob-akademii/infrastruktura/uchebnye-korpora/
2	University of Cambridge	Веб-сайт	Информационная служба Cambridge университета	https://map.cam.ac.uk
3	Московский государственный университет им. М.В. Ломоносова	Веб-сайт	Разработка МГУ им. М.В. Ломоносова	https://msu.ru/info/map/
4	Высшая школа экономики	Веб-сайт	Разработка ВШЭ	https://www.hse.ru/buildinghse/list#arm
5	политехнического Санкт-Петербургский политехнический университет Петра Великого	Веб-сайт	Разработка Санкт-Петербургского политехнического университета Петра Великого	https://www.spbstu.ru/campus-map/?ysclid=lvgimq6ih4163944977
6	Бурятская государственная сельскохозяйственная академия им. В.Р. Филиппова	Веб-сайт	Разработка БГСА им. В.Р. Филиппова	https://bgsha.ru/sveden/common/?ysclid=lvgkdvaj4953955489
7	Stanford University	Веб-сайт	Разработка Stanford University	https://campus-map.stanford.edu/

Университеты для анализа выбраны не только российские, но и зарубежные. Обучение в зарубежных университетах отличается от российского, но в приложениях, связанных с навигацией людей по кампусу, можно будет выявить свои преимущества.

1.2.3 Обзор приложений-аналогов

Для подробного анализа были выбраны 4 веб-страницы, разработанные для таких университетов, как Российская академия народного хозяйства и государственной службы при Президенте Российской Федерации, Московский государственный университет им. М.В. Ломоносова, University of Cambridge, Stanford University.

На веб-сайте Президентской академии можно узнать информацию о всех учебных корпусах: увидеть фотографии корпуса, информационную справку о количестве кабинетов, особенностях помещений внутри корпуса, а также посмотреть план расположений помещений и аудиторий на каждом этаже. Некоторые кабинеты отмечены их номером, текстом указаны такие помещения как гардероб, спортивный центр, столовая, туалет, фитнес зал, бухгалтерия и помещение коворкинга. У каждого корпуса указан адрес, название и станция метро, находящееся в ближайшей пешей доступности от учебного корпуса [2].

Просмотр данных доступен всем пользователям, посещающим сайт университета. Также университет обладает мобильным приложением.

Главная страница просмотра учебных корпусов университета представлена на рисунке 1.

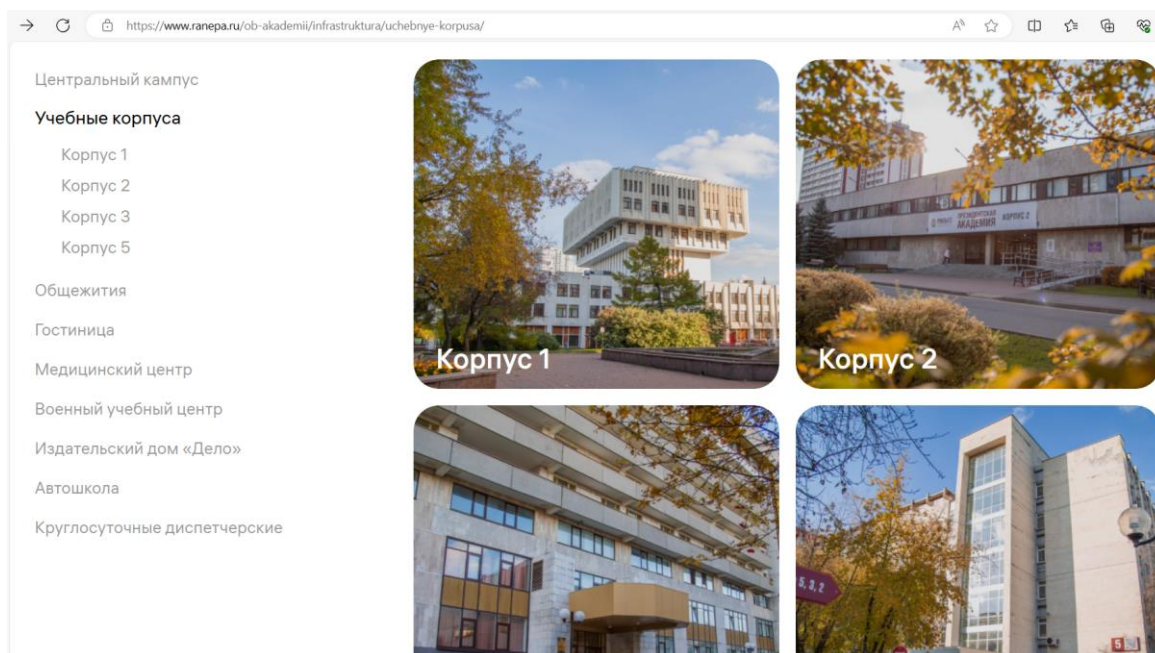


Рисунок 1 – Главная страница Президентской академии

После выбора учебного корпуса, открывается окно с подробной информацией о нем и фотографиями плана расположения помещений внутри корпуса (приложение Б рисунок Б.1 – Б.3).

Дата последнего обновления была проверена из даты выдачи последнего SSL-сертификата и является 02.04.2024 года, что говорит о частом обновлении официального сайта разработчиками.

Информация о корпусах на сайте Московский государственный университет им. М.В. Ломоносова представлены на вкладке общих сведений, схема территорий (рисунок 2). Представлено несколько ссылок схем расположений зданий, при переходе на которые можно узнать более подробную информацию [3].

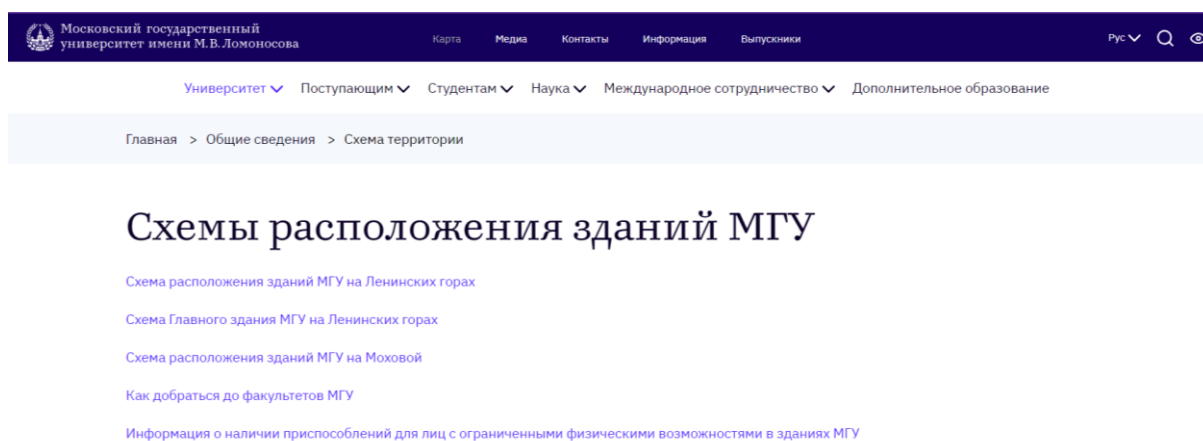


Рисунок 2 – Отображение главной страницы МГУ

По каждому корпусу можно увидеть следующие данные: фотографию корпуса, адрес, станции метро, находящиеся поблизости, основные подразделения, сгруппированные по секторам, карта с выделенным корпусом и обозначенными на ней секторами (приложение Б рисунок Б.4 – Б.5).

При нажатии на секторы, можно увидеть перечень объектов, входящих в этот сектор (приложение Б рисунок Б.6).

Сайт обновлялся недавно – 13.04.2024 года, для предоставления пользователям актуальную и верную информацию.

Рассмотренным зарубежным университетом в анализе стал University of Cambridge. Карта с корпусами и все информация по навигации людей находится на вкладке «University map».

Карта расположения кампуса обладает такими свойствами как, просмотр панорамы, рисование маркером, уменьшение, приближение, функция что здесь (рисунок 3). При выборе функции «Что здесь» в области поиска отображаются объекты, расположенные в выбранном месте [4].

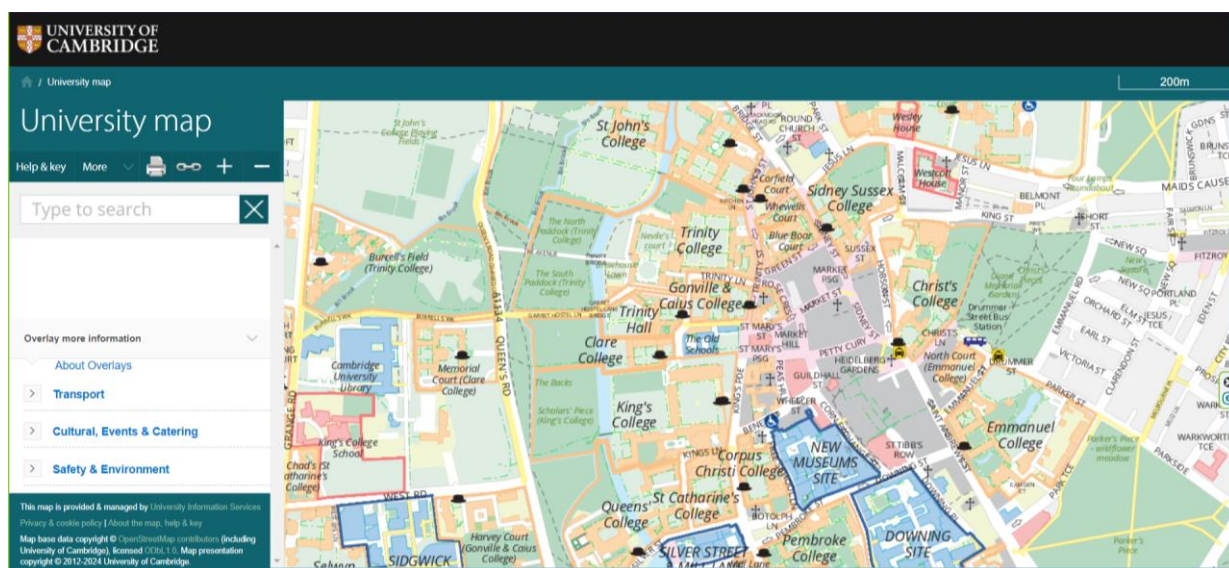


Рисунок 3 – Главная страница University of Cambridge

Отличительной особенностью данного веб-сайта университета является возможность поиска объекта в строке поиска. Из предложенного списка объектов, необходимо выбрать один. В области поиска отобразится информация о выбранном объекте. Пользователи увидят следующие данные: ссылка на главный сайт, название, адрес, телефон, факс, электронная почта (приложение Б рисунок Б.7). Данные на сайте обновляли 16.02.2024.

Еще одно зарубежное приложение-аналог веб-сайт Stanford University. Он обладает теми же функциями, что в Cambridge University (рисунок 4).

Однако отличительной особенностью на данной странице, является возможность просматривать сайт, как зарегистрированный пользователь [5].

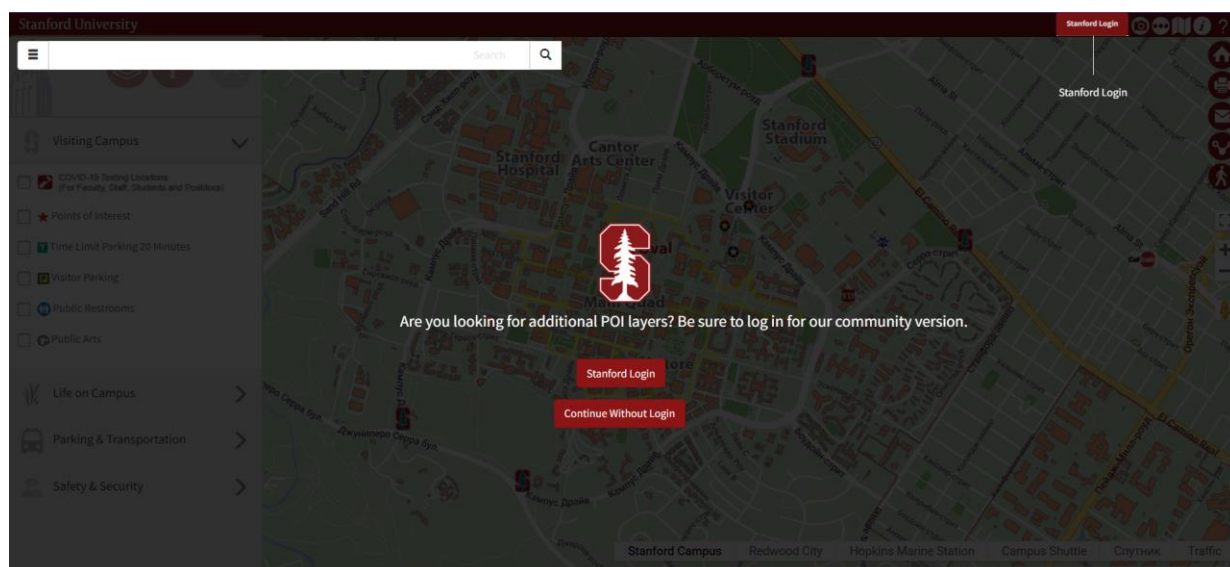


Рисунок 4 – Главная страница Stanford University

При выборе авторизации, открывается окно с вводом идентификатора студента и пароля (приложение Б рисунок Б.8). Данное действие выполнить не удалось.

Рассмотренные приложения обладают как общими критериями, так и своими отличительными особенностями.

1.2.4 Результаты сравнительного анализа

На основе анализа вышеописанных приложений для навигации людей по территории кампуса было выполнено сравнение по различным характеристикам. Результаты сравнения программ по критериям представлены в таблице 3.

Таблица 3 – Сравнительный анализ приложений

Название/критерий сравнения	Президентская академия	МГУ им. М.В. Ломоносова	Cambridge University	Stanford University
Функциональность	Информация о корпусе и его местоположение, планы этажей с кабинетами,	Информация о корпусе, их расположение на карте	Информация о корпусе, просмотр зданий на карте, поиск объектов и отображение их на карте	Информация о корпусе, просмотр зданий на карте, поиск объектов и отображение их на карте

Продолжение таблицы 3

Название/критерий сравнения	Президентская академия	МГУ им. М.В. Ломоносова	Cambridge University	Stanford University
Актуальность данных	Обновлено 02.04.2024	Обновлено 13.04.2024	Обновлено 16.02.2024	Обновлено 08.04.2024
Совместимость с устройствами	Веб-сайт, работающий на всех ОС, есть мобильное приложение, поддерживаемое на IOS, Android	Веб-сайт, работающий на всех ОС	Веб-сайт, работающий на всех ОС	Веб-сайт, работающий на всех ОС
Стоимость и доступность	Бесплатно и доступно всем	Бесплатно и доступно всем	Бесплатно и доступно всем	Бесплатно и доступно всем, но авторизованным пользователям предоставляется расширенная версия
Удобство в использовании	Да	Да	Да	Да
Отличительные черты от других приложений	Наличие подробных планов помещений и аудиторий	На карте доступен просмотр каждого корпуса с указанием основных объектов, входящих в сектор	Наличие поиска объектов, помещений, зданий и корпусов, факультетов и др.	Предоставление авторизованным пользователям расширенный функционал

В ходе проведенного сравнительного анализа был сделан вывод, что среди рассмотренных программ явного лидера не обнаружено.

Если основной критерий выбора – наличие поиска, то преимущество у зарубежных приложений-аналогов. Если необходимо приложение с отображением плана помещений внутри корпуса, то стоит выделить Президентскую Академию.

В остальных функциях приложения имеют схожие структуры отображения всех корпусов университетской территории.

1.3 Постановка задачи на разработку сервиса

1.3.1 Цели разработки и назначение сервиса

Целью разработки является приложение для помощи в ориентации внутри корпусов и на территории кампуса, которое автоматизирует бизнес-процессы, связанные с отображением информации и осуществлением поиска.

Преимущества приложения будут в предоставляемых функциях и назначениях:

- наглядное представление расположения аудиторий;
- помощник в нахождении нужного кабинета;
- собрал в себе все необходимые элементы университетского корпуса, такие как аудитории, столовые, гардеробы, туалеты и др.;
- показывает, как расположены кабинеты относительно друг друга;
- показывает сколько этажей в каждом корпусе;
- даст понять, чего еще не хватает в корпусе и на этажах корпуса;
- поможет спроектировать новые корпуса на основе лучшего варианта расположения объектов зданий;
- показывает, деканаты каких кафедр расположены в корпусе;
- дает краткую информацию о каждом корпусе.

Таким образом, приложение будет наиболее удовлетворять потребности пользователей, включая в себя все преимущества приложений-аналогов, рассмотренных ранее.

1.3.2 Классы и роли пользователей сервиса

В университетской сфере можно выделить такие классы пользователей, как студенты, преподаватели, абитуриенты, учебно-вспомогательный персонал и гости университета.

Для кого будет актуально приложение:

- для студентов, поступивших на первый курс;
- для преподавателей, устроившихся на работу;

- для преподавателей, ведущих пары в неродном корпусе;
- для школьников, планирующих поступать в университет;
- для гостей, посещающих университет по рабочим вопросам.

Вышеописанные роли будут распределены в двух видах пользователей приложения с предоставлением им соответствующего функционала. Студенты и преподаватели относятся к авторизованным пользователям, гости, школьники к незарегистрированным. Описание функции приложения для соответствующей роли описано в Таблица 4. Некоторые функции пользователей совпадают.

Таблица 4 – Описание назначений для ролей пользователей приложения

Функция/роль	Незарегистрированный пользователь	Авторизованный пользователь
Поиск кабинета	Поиск кабинета по наименованию с отображением информации о кабинете и схеме расположения кабинета на этаже относительно других кабинетов	Поиск кабинета по фильтру, выбранному из списка с отображением информации о кабинете и схеме расположения кабинета на этаже относительно других кабинетов
Просмотр списка корпусов	Просмотр списка корпусов с отображением информации о их наименовании, адресе и возможности перехода к подробной информации о них.	
Просмотр подробной информации о корпусе	Доступ к информации о количестве корпусов, их адресах, количествах этажей в здании, кафедрах, расположенных в корпусе.	Доступ к информации о количестве корпусов, их адресах, количествах этажей в здании, кафедрах, расположенных в корпусе, а также планах эвакуации для каждого этажа в здании.
Авторизация	Незарегистрированный пользователь не сможет получить доступ к расширенной версии приложения.	При заполнении верных данных логина и пароль будет осуществлена авторизация пользователя для предоставления ему расширенной версии.
Редактирование данных	Нет доступа к этой функции	Пользователь сможет изменить и сохранить личные данные в приложении.

Таким образом, распределение функций по ролям и предоставление соответствующих им информации об объектах поможет грамотно спланировать разработку приложения.

1.3.3 Автоматизируемые бизнес-процессы

Разрабатываемое приложение предназначено для решения следующих информационных задач:

- регистрация;
- авторизация пользователя;
- просмотр личного кабинета;
- редактирование личных данных;
- просмотр информации о корпусах;
- просмотр расширенной информации о корпусах, их планы эвакуации и фотографии;
- просмотр факультетов и кафедр, деканаты которых расположены в определенных корпусах;
- поиск по наименованию кабинета;
- поиск кабинета по фильтрам с отображение всей информации о нем, включая план расположения кабинета на этаже.

Диаграмма деятельности отображает динамические аспекты поведения системы, как поток управления переходит от одной деятельности к другой. Вершины диаграммы деятельности являются состояния действия или деятельности, а дуги – переходы от одного состояния действия к другому [6].

Автоматизируемыми процессами будут являться действия, связанные с загрузкой данных, отображением информации и получение новых данных от пользователя. Данные о корпусах и кабинетах доступны только для загрузки и отображения в приложении. Личные данные пользователя добавляются в систему, отображаются и доступны для редактирования пользователю, кто авторизовался в системе.

Сценарий функционирования автоматизируемых процессов приложения отображен на диаграмме деятельности, представленной на рисунке 5.

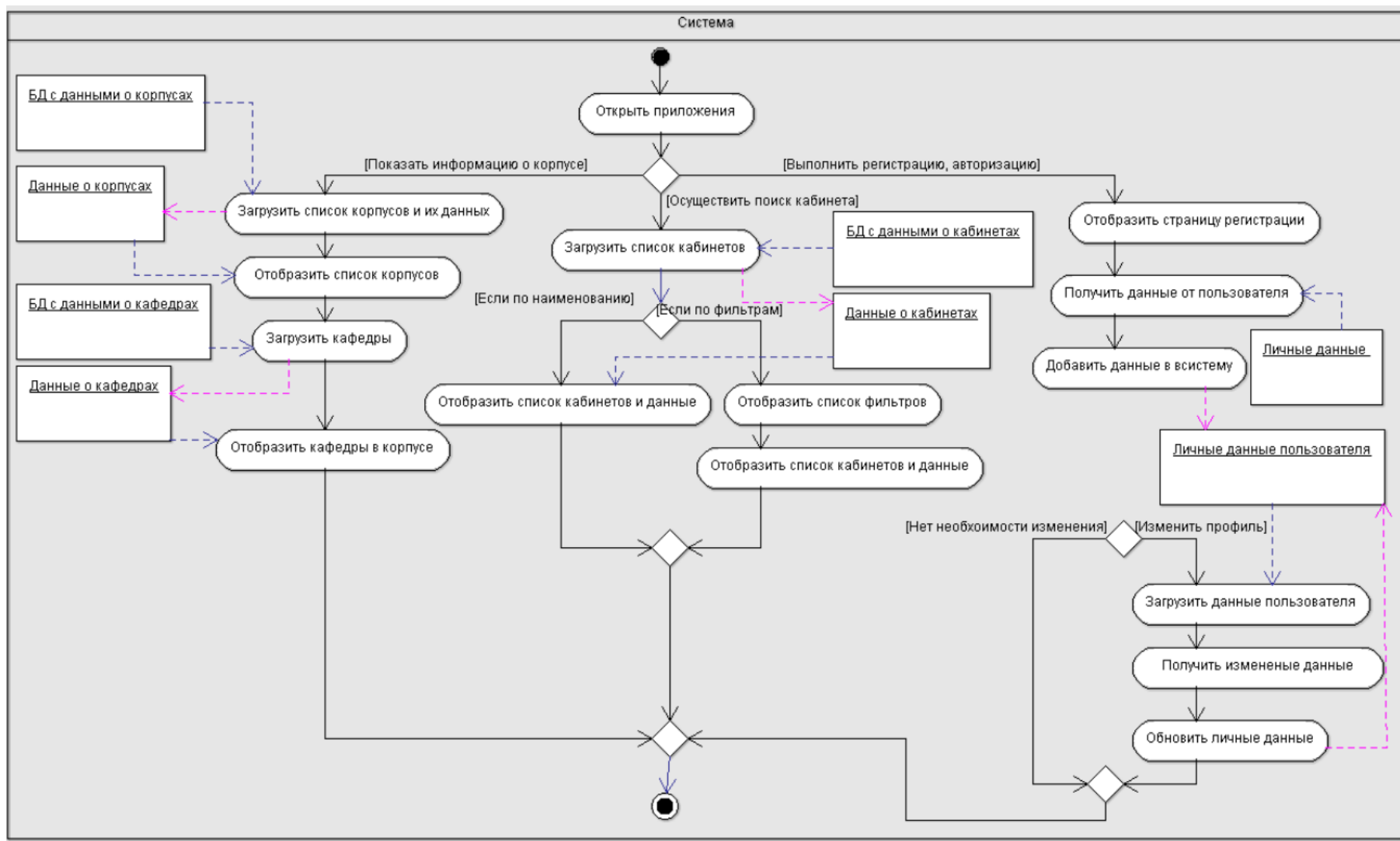


Рисунок 5 – Диаграмма деятельности автоматизируемых процессов

Спецификации информационных задач диаграммы деятельности представлены в таблице 5.

Таблица 5 – Автоматизируемые процессы диаграммы деятельности

Действие (операция)	Описание
Открыть приложения	Система отображает главную страницу приложения
Загрузить список корпусов и их данных	Система загружает список всех корпусов и их данные из базы данных
Отобразить список корпусов	Система отображается список всех корпусов для просмотра пользователями
Загрузить кафедры	Система загружает информацию по кафедрам из БД
Отобразить кафедры в корпусе	Система отображает информацию по кафедрам в выбранном корпусе для просмотра пользователями
Загрузить список кабинетов	Система загружает кабинеты и их информацию из БД
Отобразить список кабинетов и данные	Система отображает список подходящих кабинетов при поиске по наименованию
Отобразить список фильтров	При поиске по фильтрам система отображает список доступных пользователю фильтров
Отобразить список кабинетов и данные	Система отображает список подходящих кабинетов при поиске по фильтрам
Отобразить страницу регистрации	Система отображает страниц регистрации при переходе пользователя по кнопке зарегистрироваться
Получить данные от пользователя	После заполнения полей пользователям при регистрации система считывает личные данные пользователя
Добавить данные в систему	Система добавляет в базу данных личные данные зарегистрированного пользователя
Загрузить данные пользователя	При редактировании личных данных пользователем система загружает из БД данные и отображает на странице
Получить измененные данные	После редактирования пользователем своих данных система считывает их
Обновить личные данные	Система обновляет в БД личные данные пользователя

Таким образом, вся информация и данные загружаются и обновляются автоматически, без вмешательства пользователя в базу данных при использовании приложения.

1.4 Планирование процесса разработки сервиса

Планирование процесса разработки подразумевает выявление ряда задач и установление конкретных сроков их выполнения для получения желаемого результата. Результатом будет приложение для помощи в ориентации внутри корпусов и на территории кампуса, а также осуществление поиска кабинета по фильтрам, представлении подробной информации о кабинете и кафедрах, расположенных в корпусах.

Перед началом разработки приложения были выявлены ресурсы на разработку, которые представляют из себя людей определенных профессий и необходимые оборудования. Список затраченных ресурсов создан с помощью средств построения таблиц в Excel и представлен на рисунке 6.

Название ресурса	Тип	Краткое название	Макс.единица	Стандартная ставка	Начисление	Базовый календарь
Аналитик	Трудовой	А	100%	500,00р./ч	Пропорционально	Стандартный
Программист	Трудовой	П1	100%	625,00р./ч	Пропорционально	Стандартный
Тестировщик	Трудовой	Т	100%	375,00р./ч	Пропорционально	Стандартный
Дизайнер	Трудовой	Д	100%	468,75р./ч	Пропорционально	Стандартный
Компьютер	Материальный	К			Пропорционально	
Печать	Затраты	П			Пропорционально	
Интернет	Затраты	И			Пропорционально	

Рисунок 6 – Ресурсы разработки

Затем было выявлено ряд задач с указанием конкретных сроков их исполнения. Задачи проекта представлены в таблице 6. Длительность рассчитывается в количестве дней между датой начала и датой завершения.

Таблица 6 – График задач и их длительность

Задача	Дата начала	Дата завершения	Длительность	Название ресурса
Начало работы над проектом	01.02.24	02.02.24	1	Компьютер
Анализ предметной области	01.02.24	09.02.24	9	Аналитик
Поиск и сравнение приложений аналогов	16.02.24	17.02.24	2	Аналитик, интернет
Постановка задачи на разработку сервиса	21.02.24	23.02.24	3	Аналитик
Определение автоматизируемых бизнес-процессов	01.03.24	02.03.24	2	Аналитик

Продолжение таблицы 6

Задача	Дата начала	Дата завершения	Длительность	Название ресурса
Проектирование приложения	03.03.24	07.03.24	5	Дизайнер, аналитик, интернет
Разработка базы данных	14.03.24	18.03.24	4	Программист
Разработка пользовательского интерфейса	20.03.24	25.03.24	5	Программист, интернет
Разработка программного обеспечения	26.03.24	26.04.24	30	Программист, интернет
Планирование процесса контроля качества ПО	27.04.24	28.04.24	2	Тестировщик
Тестирование программных средств	01.05.24	02.05.24	2	Тестировщик, интернет
Оценка качества программного кода	03.05.24	04.05.24	2	Тестировщик, интернет
Оформление и сбор документации	16.05.25	30.05.24	15	Аналитик

По результатам составленного плана общее количество потраченных дней на планирование и разработку приложения составляет дней.

С помощью инструмента Excel была составлена диаграмма Ганта, показывающая задачи и количество затраченных дней на них (рисунок 7).

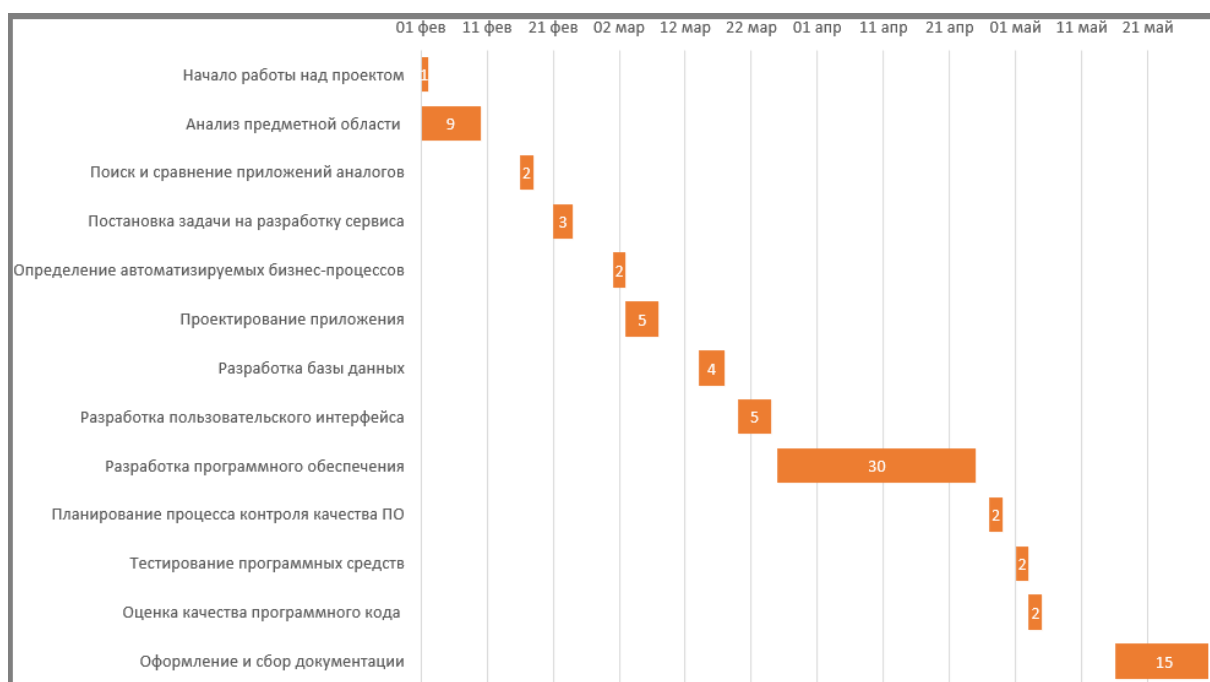


Рисунок 7 – Диаграмма Ганта

Финансовые затраты будут составлять на оплату трудовой деятельности людей, участвующих в проекте. Эти данные наглядно представлены в диаграмме (рисунок 8).

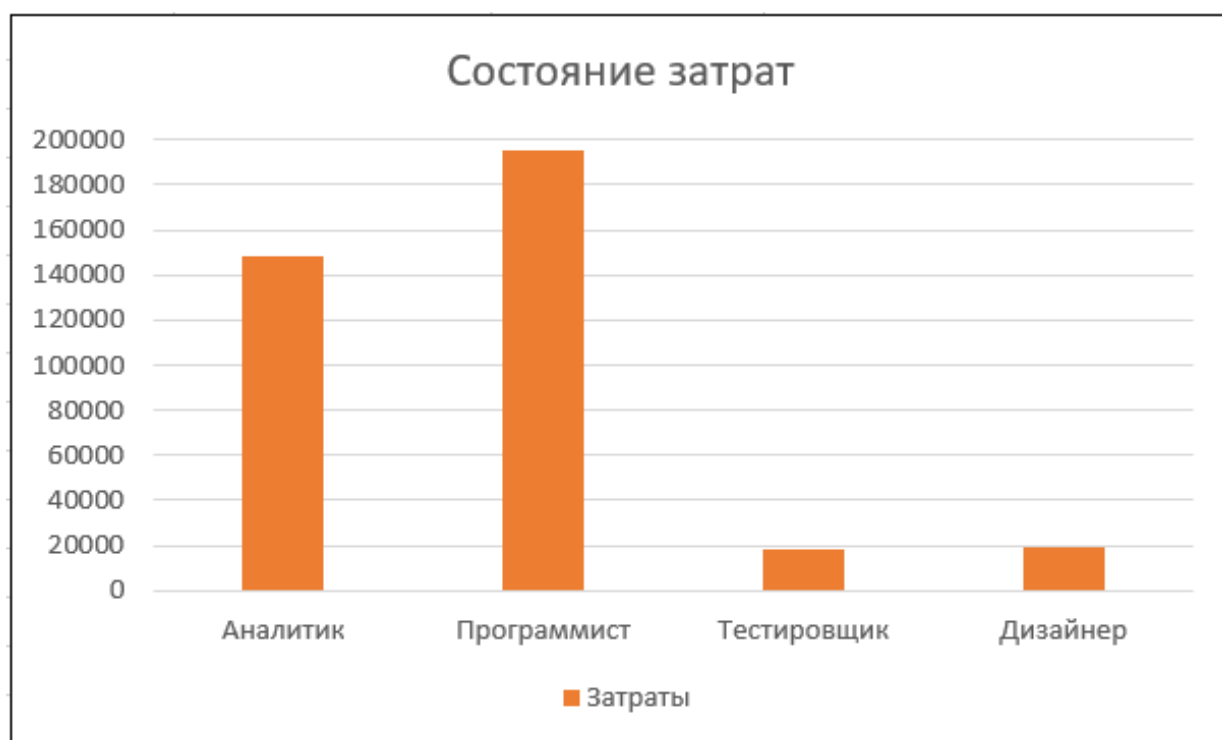


Рисунок 8 – Диаграмма состояния затрат

По результатам планирования проекта по разработки приложения «Аудиторный гид» выявлено, что финансовых ресурсов потребуется триста семьдесят девять тысяч семьсот пятьдесят рублей на оплату четырех работников, которые будут выполнять проект в течение четырех месяцев согласно стандартному рабочему календарному графику.

2 Проектирование приложения «Аудиторный гид»

2.1 Моделирование функциональных возможностей приложения

Диаграмма вариантов использования предоставляет наглядное описание взаимоотношения между группами вариантов использования и действующими лицами, участвующими в процессе.

Диаграмма вариантов использования состоит из следующих элементов:

- актеры, которые взаимодействуют с системой;
- варианты использования, определяемые взаимодействием актеров с системой для достижения определенной цели;
- отношения между актерами и вариантами использования, какие актеры участвуют в выполнении конкретных вариантов использования;
- система, для которой создается диаграмма.

С помощью диаграммы вариантов использования упрощается коммуникация разработчика с заказчиком по требованиям к системе.

Список функциональных возможностей приложения был представлен в виде диаграммы для незарегистрированных пользователей на рисунке 9.

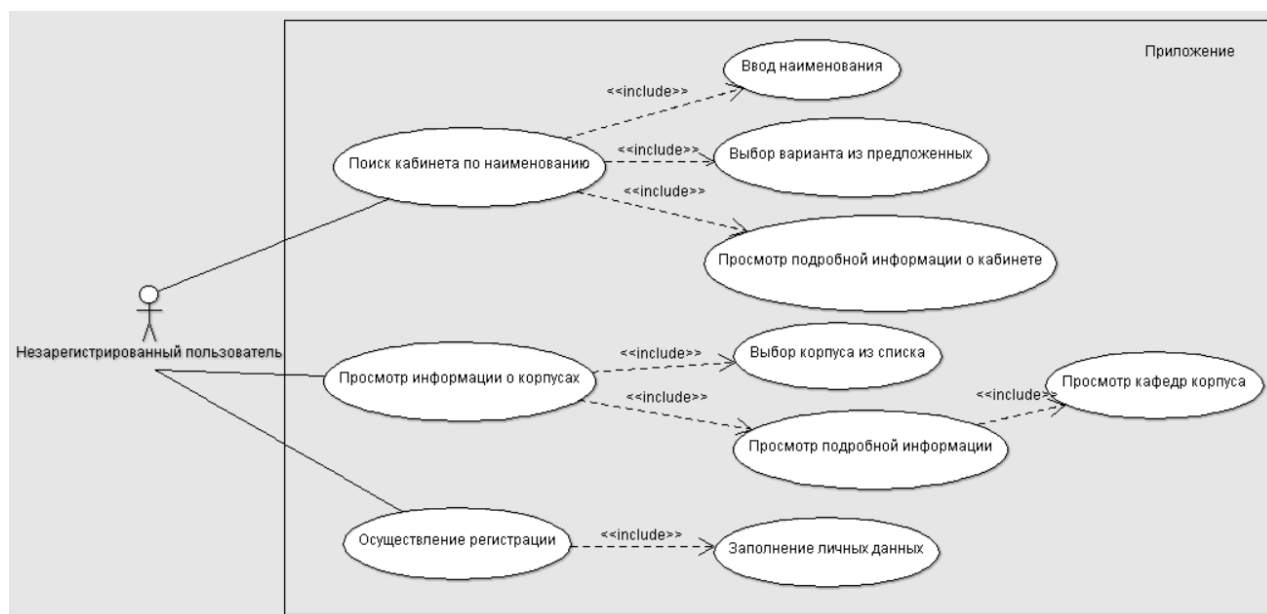


Рисунок 9 – Диаграмма вариантов использования для незарегистрированного пользователя

В данной диаграмме роль актера выполняет незарегистрированный пользователь, который взаимодействует с приложением. Актер связан отношением «Ассоциация» со следующими вариантами использования:

- поиск кабинета по наименованию;
- просмотр информации о корпусах;
- осуществление регистрации.

Остальные варианты использования связаны отношением включения и составляют группу вариантов использования, относящихся к одному и тому же варианту использования. Описание каждого варианта использования диаграммы для незарегистрированного пользователя представлено в таблице 7.

Таблица 7 – Диаграмма вариантов использования «Незарегистрированный пользователь»

Действие (операция)	Описание
Поиск кабинета по наименованию	Незарегистрированный пользователь имеет возможность искать информацию о кабинете по наименованию
Ввод наименования	Незарегистрированный пользователь вводит текст в строку поиска
Выбор варианта из предложенных	Незарегистрированный пользователь имеет возможность просматривать и выбирать кабинет из списка, подобранной системой
Просмотр подробной информации о кабинете	Незарегистрированный пользователь при выборе кабинета имеет возможность просматривать информацию, а именно наименование, корпус, этаж, тип кабинета, оборудование, фото, план этажа, на котором он находится
Просмотр информации о корпусах	Незарегистрированный пользователь имеет возможность смотреть информацию о корпусе
Выбор корпуса из списка	В приложении формируется и отображается список всех корпусов, доступных для выбора и просмотра информации незарегистрированным пользователям
Просмотр подробной информации	Незарегистрированный пользователь имеет возможность просматривать подробную информацию о корпусе, а именно его наименование, номер, адрес, количество этажей
Просмотр кафедр корпуса	Незарегистрированный пользователь при выборе корпуса имеет возможность просматривать информацию о деканатах кафедр, находящихся в выбранном корпусе
Осуществление регистрации	Незарегистрированный пользователь имеет возможность переходить на страницу регистрации
Заполнение личных данных	Незарегистрированный пользователь при выборе регистрации вводит личные данные

Список функциональных возможностей приложения для авторизованных пользователей представлен на рисунке 10.

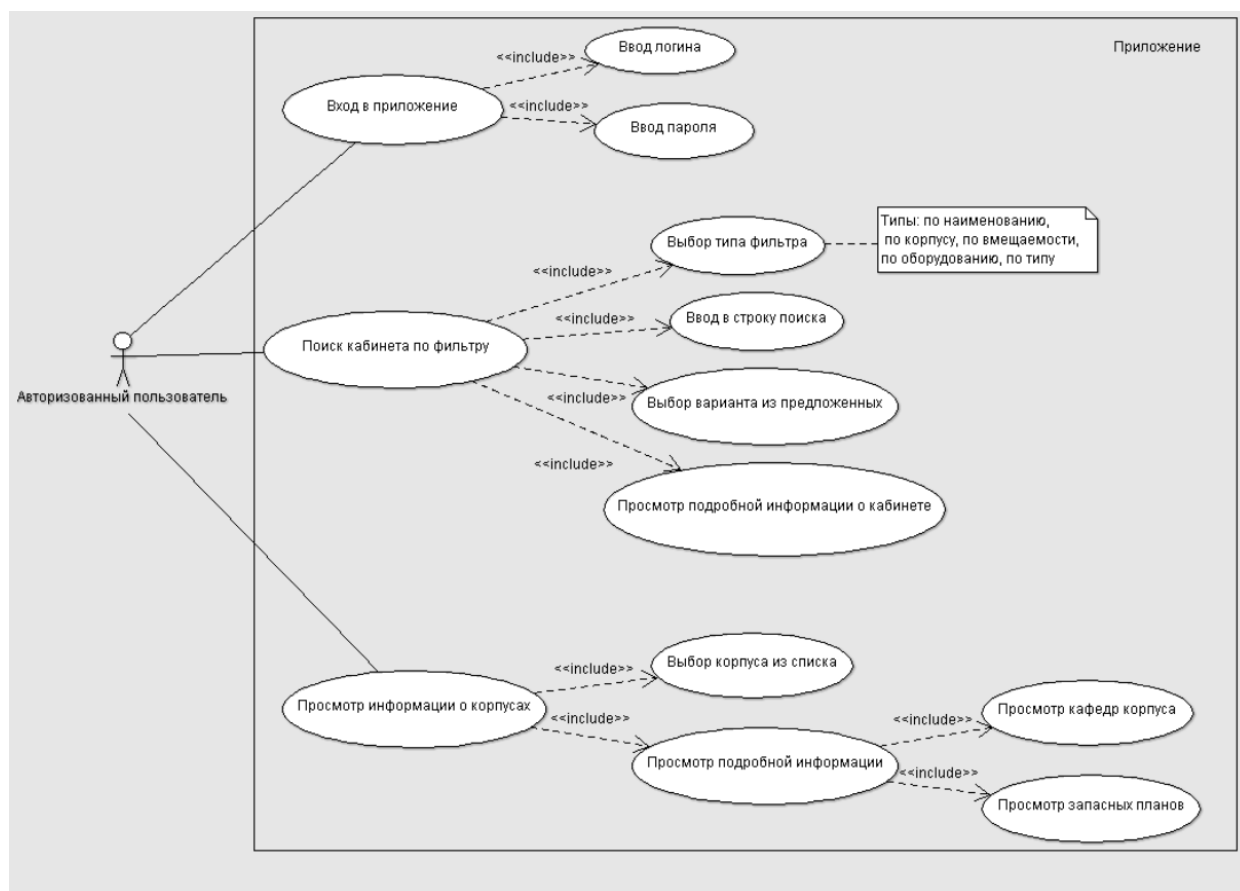


Рисунок 10 – Диаграмма вариантов использования для авторизованного пользователя

В данной диаграмме роль актера выполняет авторизованный пользователь, который взаимодействует с приложением. Актер связан отношением «Ассоциация» со следующими вариантами использования:

- вход в приложение;
- поиск кабинета по фильтру;
- просмотр информации о корпусах.

Остальные варианты использования связаны отношением включения и составляют группу вариантов использования, относящихся к одному и тому же варианту использования. Описание каждого варианта использования диаграммы вариантов использования для авторизованного пользователя представлено в таблице 8.

Таблица 8 – ДВИ «Авторизованный пользователь»

Действие (операция)	Описание
Вход в приложение	Всем пользователем доступен вход в приложение
Ввод логина	Авторизация доступна только авторизованному пользователю, необходимо ввести логин
Ввод пароля	Авторизация доступна только авторизованному пользователю, необходимо ввести пароль
Поиск кабинета по фильтру	Авторизованный пользователь имеет возможность искать кабинет по фильтрам
Выбор типа фильтра	Авторизованный пользователь имеет возможность выбирать тип фильтра из списка: тип кабинета, номер корпуса, оборудование, наименование, вместимость
Ввод в строку поиска	Авторизованный пользователь вводит текст в строку поиска
Выбор варианта из предложенных	Авторизованный пользователь имеет возможность просматривать и выбирать кабинет из списка, подобранной системой
Просмотр подробной информации о кабинете	Авторизованный пользователь при выборе кабинета имеет возможность просматривать информацию, а именно наименование, корпус, этаж, тип кабинета, оборудование, фото, план этажа, на котором он находится
Просмотр информации о корпусах	Авторизованный пользователь имеет возможность смотреть информацию о корпусе
Выбор корпуса из списка	В приложении формируется и отображается список всех корпусов, доступных для выбора и просмотра информации авторизованным пользователям
Просмотр подробной информации	Авторизованный пользователь имеет возможность просматривать подробную информацию о корпусе, а именно его наименование, номер, адрес, количество этажей
Просмотр кафедр корпуса	Авторизованный пользователь при выборе корпуса имеет возможность просматривать информацию о деканатах кафедр, находящихся в выбранном корпусе
Просмотр запасных планов	Авторизованный пользователь имеет возможность просматривать эвакуационные планы каждого этажа выбранного корпуса

Наглядное представление категорий пользователей и сопоставление их с описанием функции, помогает представить реализацию интерфейса и сформулировать дальнейшее планирование разработки.

Также были разработаны сценарии выполнения процессов поиска кабинета, просмотра информации о корпусах и регистрации пользователя в виде таблиц, представленных в приложении В рисунок В.1 – В.3.

Каждый процесс отвечает за свои функции и реализацию в приложении.

2.2 Разработка концептуальной модели системы аудиторного гида университета

Реляционная база данных состоит из таблиц. При этом каждая таблица включает в себя столбцы и строки.

На основе анализа предметной области и результатов моделирования бизнес-процессов была разработана концептуальная схема базы данных приложения для помощи в навигации людей. Концептуальная модель представляет информационное описание предметной области с учетом логических взаимосвязей. Схема представлена на рисунке 11.

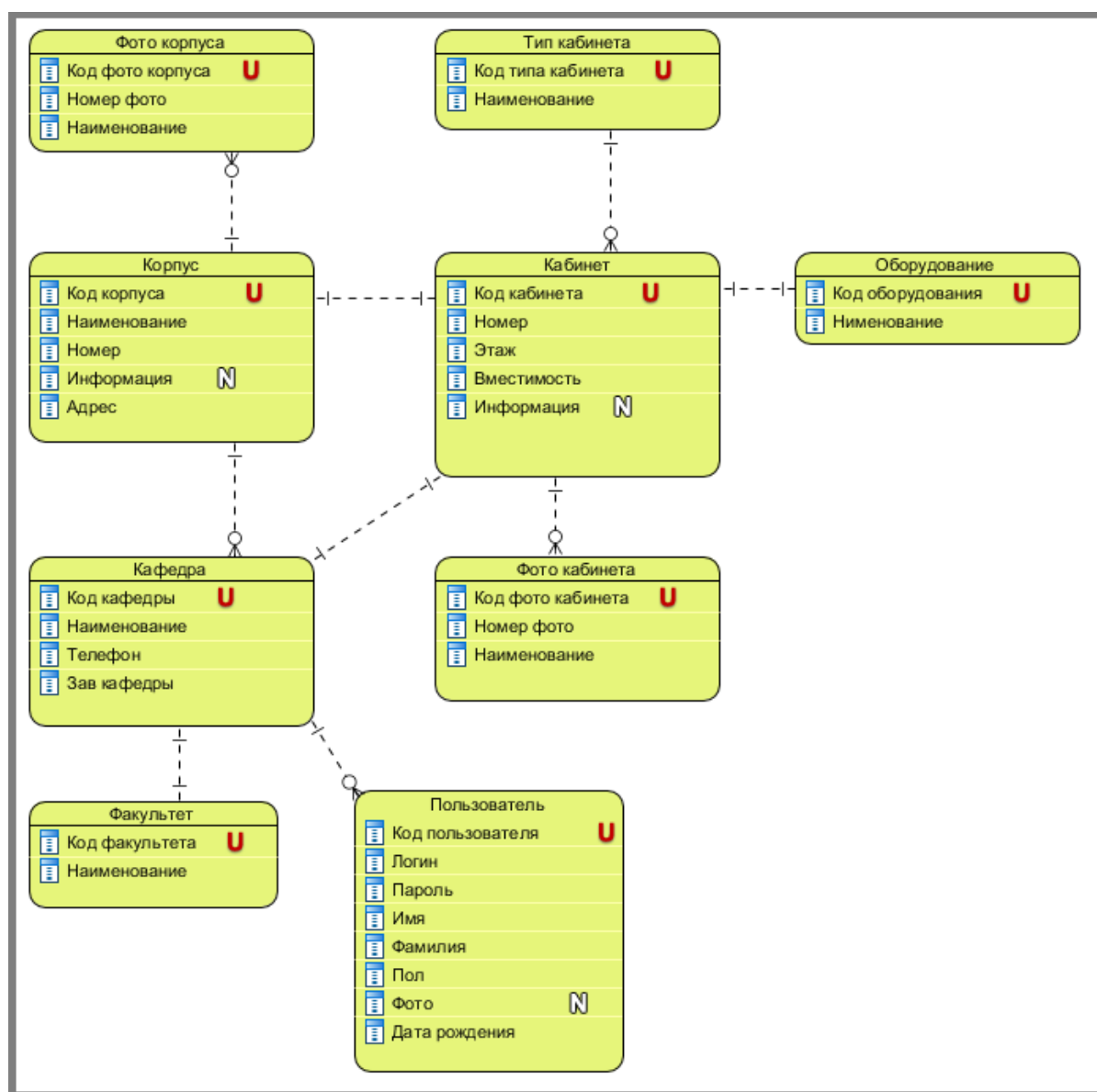


Рисунок 11 – Концептуальная модель предметной области

Описание спецификаций сущностей разрабатываемого приложения приведено в таблице 9.

Таблица 9 – Описание сущностей предметной области концептуальной модели

№ п/п	Наименование	Описание	Категория	Количество экземпляров	Режим обновления
1	Корпус	Университетское здание	Стержневая	19	5% в год
2	Фото корпуса	У корпусов имеется некоторое количество фотографий	Характеристическая	150	5% в год
3	Кабинет	Аудитория или помещение в здании корпуса	Стержневая	50	30% в год
4	Фото кабинета	У кабинетов имеется некоторое количество фотографий	Характеристическая	60	5% в год
5	Оборудование	Применяемое оборудование в кабинете	Характеристическая	5	30% в год
6	Факультет	Факультет в университете	Стержневая	8	20% в год
7	Кафедра	Кафедра в университете	Стержневая	91	40% в год
8	Пользователь	Зарегистрированный пользователь	Стержневая	20	70% в месяц
9	Тип кабинета	Тип кабинета	Характеристическая	10	5% в год

Описание связей между сущностями представлено в приложении Г (таблица Г.1). Описание атрибутов всех сущностей приведено в таблицах приложения Г (таблица Г.2-Г.10)

Полученные результаты являются исходными данными для моделирования базы данных и разработки приложения.

2.3 Разработка программной архитектуры приложения

При разработке приложения для ВКР был выбран клиент-серверный вид архитектуры по причине ее распространенности, удобстве в разработки и поддержке.

Людям потребовалось разделить общую логику работы на два независимых уровня: на уровень клиента и уровень сервера. Все данные хранятся на одном компьютере, а другие пользователи могут видеть все изменения в режиме реального времени.

Разработчик работает с программой на своем компьютере, клиент отвечает только за отображение данных. Он не может напрямую изменять данные, но может попросить сервер сделать это. Клиент передает запрос серверу на изменение данных и передает эти данные, а сервер в свою очередь занимается всей бизнес-логикой: сохраняет данные, осуществляет проверки, уведомляет клиента и т. д.

Один и тот же компьютер в сети может быть и клиентом, и сервером. Только поведение устройство определяет его роль.

Для определения клиента и сервера и предотвращения путаницы между ними существуют простые описания для каждого вида. Если программа запрашивает данные извне, отображает их, но не может изменить самостоятельно, то это программа клиент. Если программа может ждать запросы, менять или создавать данные, отправлять ответы, то это программа сервер.

Самое очевидное преимущество клиент – серверной архитектуры – данные хранятся в одном месте и всегда актуальны. Есть и другой, не менее важный плюс. Раньше всю вычислительную логику старались выносить на сервер, потому что сервера были в разы мощнее персональных компьютеров и справлялись с задачами быстрее клиентских машин. Но сегодня любой компьютер может справиться с большинством задач по расчёту и подготовке данных. Это в свою очередь позволило программистам решать, какую логику

исполнять на стороне клиента, а какую на стороне сервера. При балансе нагрузки распределения между клиентом и сервером можно добиться увеличения производительности и отказоустойчивости системы.

Есть и недостатки и это зависимость всей цепочки от сервера. При отказе работы сервера по разным причинам, программа у клиентов не заработает.

На рисунке 12 отображена диаграмма развертывания приложения, которая описывает простой вид отображения клиент-серверной архитектуры.

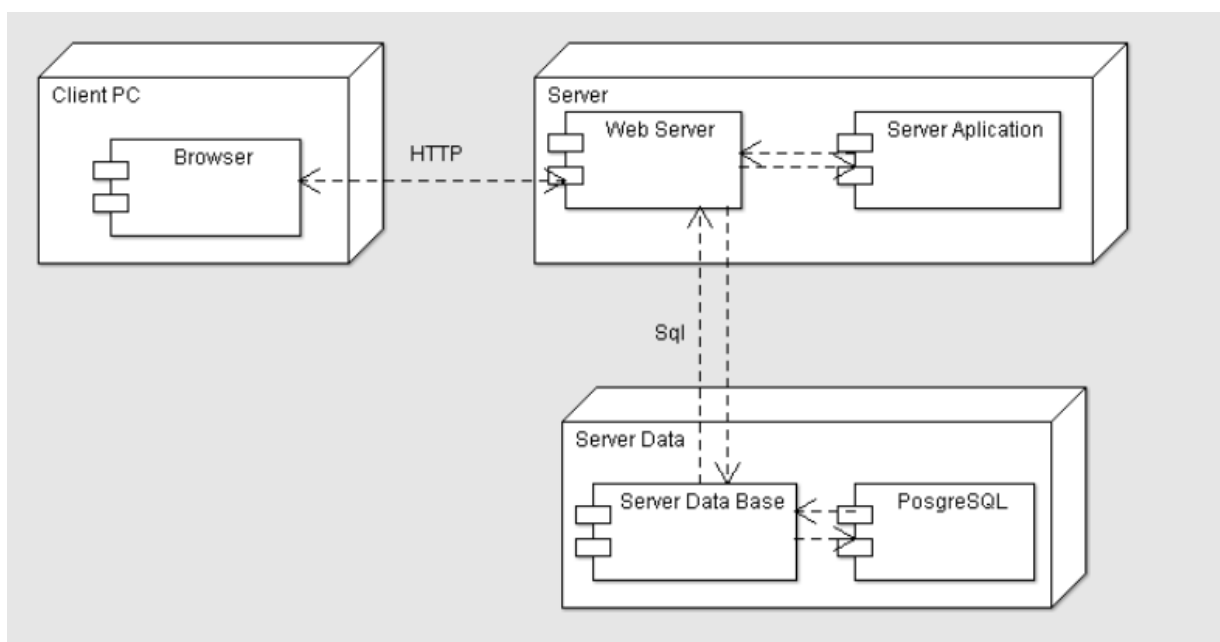


Рисунок 12 – Диаграмма развертывания приложения аудиторного гида

При такой организации работы приложения управление данными централизовано, эффективно распределяется нагрузка приложения между клиентом и сервером, что позволяет сделать приложение гибким и масштабируемым.

2.4 Разработка прототипа пользовательского интерфейса

Под интерфейсом понимается средство взаимодействия между пользователем и компьютером. Пользовательский интерфейс позволяет эффективно выполнять определенные задачи или достигать определенной цели.

Существует два типа пользовательских интерфейса: графический и голосовой. При графическом интерфейсе пользователи взаимодействуют с

помощью визуальных элементов, которые представляют собой страницы и экраны, состоящие из статических форм, так и из интерактивных элементов. Таких как кнопки и элементы. При голосовом интерфейсе пользователи взаимодействуют с приложения только с помощью своего голоса. Это стало возможно благодаря достижениям обработки и естественного языка, технологии, основанной на машинном обучении, которое позволяет компьютеру понимать человеческий язык.

Важными качествами пользовательского интерфейса являются следующие параметры:

- простота;
- последовательность (согласованность);
- интуитивность;
- профилактика.

Под простой понимается достижение цели пользователя при помощи нескольких простых и понятных действий. Интерфейс должен иметь единый стиль, размер и шрифт – в этом заключается согласованность интерфейса.

Для пользователя использование интерфейса должно быть понятно, без получения определенных знаний или просмотра руководства оператора приложения.

Последний параметр интерфейса – профилактика. Подразумевается предусмотренное разработчиком предотвращение пользователям нежелательных действий.

В выпускной работе используется графический пользовательский интерфейс. Был сформирован прототип интерфейса, включающий в себя различные элементы экранного окна.

Главное окно, открывающееся при запуске приложения отображено на рисунке 13. Основные элементы окна представляют блоки с текстовой информацией, представленными в виде прокручивающегося списка. А также поля для записи значения поиска и кнопки для входа.

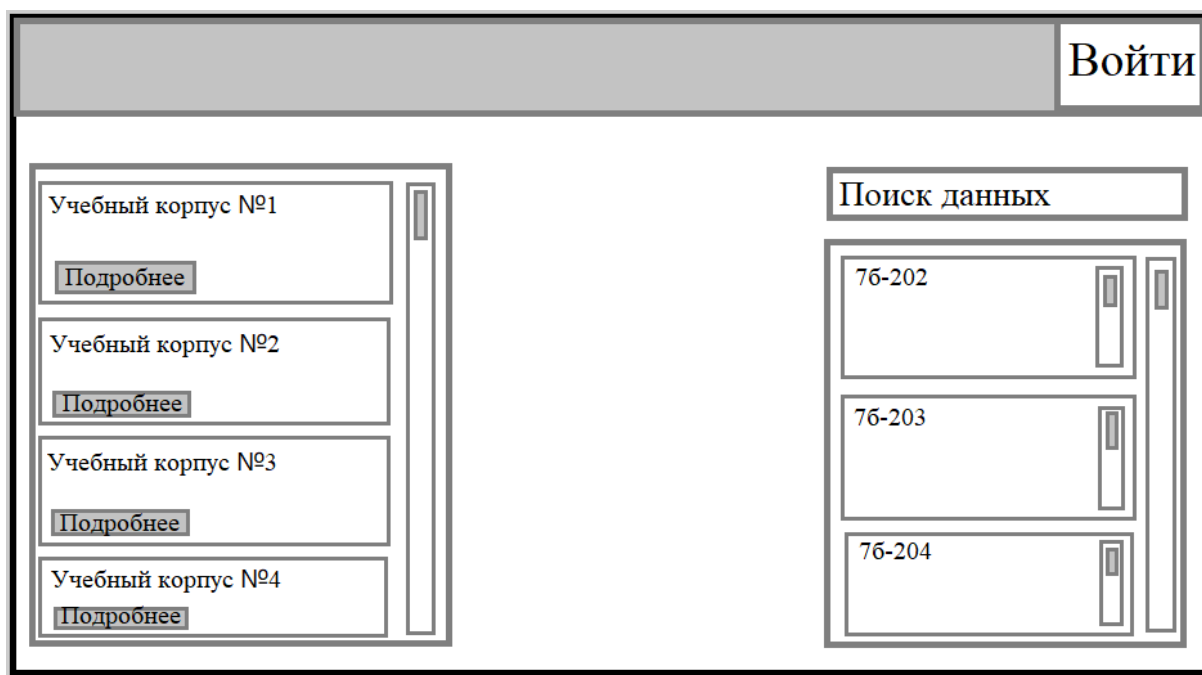


Рисунок 13 – Прототип главного окна

Всего было сформировано девять прототипов окон разрабатываемого приложения. Они представлены в приложении Д. Их список следующий:

- главное окно при запуске приложения (рисунок Д.1);
- окно просмотра информации о корпусах для незарегистрированного пользователя (рисунок Д.2);
- окно просмотра информации о кабинете (рисунок Д.3);
- окно входа в приложение (рисунок Д.4);
- окно регистрации пользователя (рисунок Д.5);
- окно поиска с фильтрами авторизованного пользователя (рисунок Д.6);
- окно просмотра расширенной информации о корпусах (рисунок Д.7);
- окно просмотра личного кабинета пользователя (рисунок Д.8);
- окно редактирования данных (рисунок Д.9).

На рисунке 14 можно проследить сценарий диалога пользователя, т. е. последовательность (иерархия) экранных окон, которые появляются в приложении. Стрелки показывают из какой страницы можно перейти и в каком направлении. Стрелки в обе стороны позволяют возвращаться пользователю к предыдущей странице.

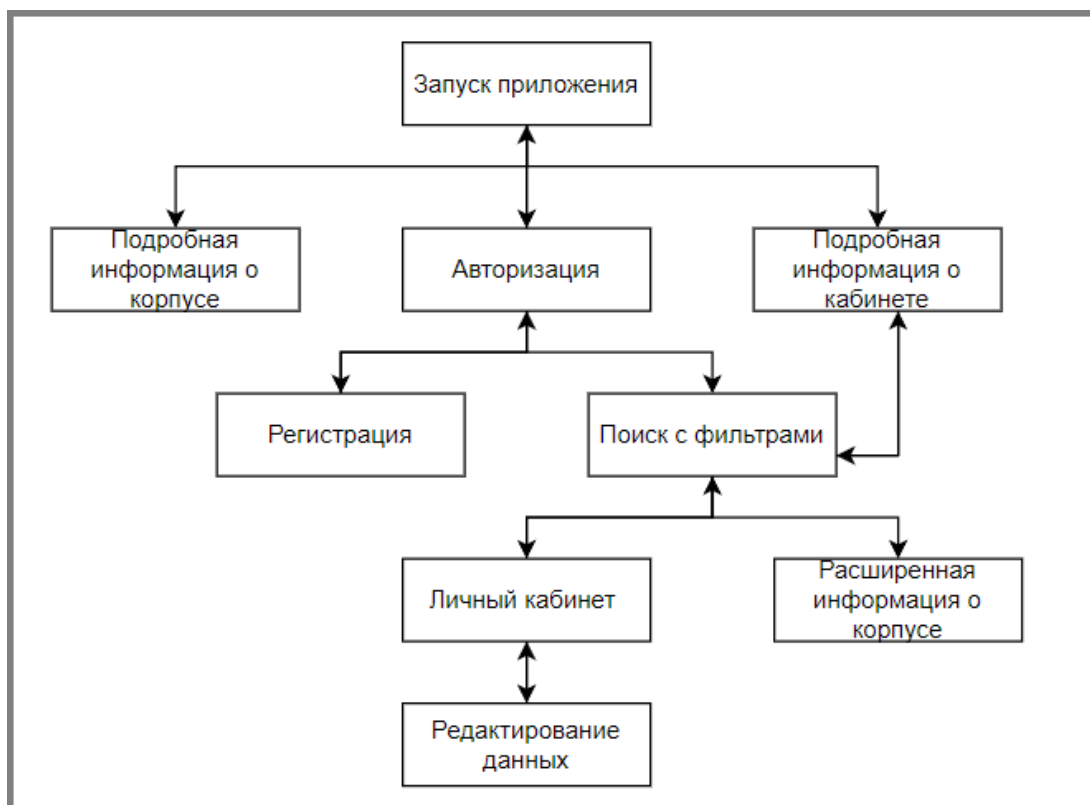


Рисунок 14 – Последовательность перехода страниц

Используя вышеописанные окна, пользователь может выполнять все функции приложения.

2.5 Разработка требований к аппаратно-программному окружению

Необходимо определить требования к программному и аппаратному обеспечению при разработке приложения в помощи навигации людей по университетскому кампусу.

Аппаратные требования были выявлены следующие:

- наличие процессора Intel Core i5 или эквивалентный;
- размер оперативной памяти не менее 8 ГБ;
- в составе устройства жесткий диск SSD 256 ГБ;
- монитор любого расширения.

Программные требования определены следующие:

- один из вариантов операционной системы: Windows 10, macOS или Linux;

- установка на сервере Java Development Kit (JDK) для компиляции и выполнения Java кода;
- установка и использование одной из среды разработки совместимой с JDK: Eclipse, IntelliJ IDEA;
- взаимодействие с установленным на сервере Tomcat или Jetty для обеспечения выполнения код приложения;
- для хранения информации использование реляционной базы данных, такой как MySQL, PostgreSQL или Oracle;
- работа с фреймворком Spring Boot для создания RESTful веб-сервисов веб-приложения;
- в сборке проекта и управлении зависимостями необходима работа инструмента сборки Maven или Gradle.

Также можно выделить сетевое требование – наличие доступа к Интернету.

При данном наборе необходимых требования к аппаратному и программному обеспечению можно выполнить разработку веб-приложения.

2.6 Обоснование выбора средств разработки

2.6.1 Обоснование выбора средств автоматизации

Для моделирования и проектирования программного обеспечения существует множество вариантов. Для любого уровня использования и удобство разработчика. Такие программы поддерживают унифицированные языки моделирования (UML) 2.4, SysML, ERD, DFD, BPMN 2.0, ArchiMate 2.0 и другие [7].

Список самых распространенных программ для моделирования UML представлен ниже:

- Visual Paradigm – обладает большим списком возможностей и поддерживает все типы языка;
- StartUML – в своем составе имеет одиннадцать типов диаграмм;

- Cасоо – помимо диаграмм позволяет создавать блок-схемы, может предоставить шаблоны проектов по бизнесу, дизайну и других видов;
- Gliffy – особенностью этого инструмента является рисование диаграмм от руки, как на доске;
- Microsoft Visio – является частью разработки Microsoft Office и может взаимодействовать и соединяться с другими их продуктами;
- ArgoUML – имеет открытый исходный код и поддерживается на многих языках, из-за чего обладает большой популярностью в использовании его [8].

Для работы с диаграммами были выбраны два доступных и распространенных инструмента: Visual Paradigm и ArgoUML.

Visual Paradigm понятен и удобен в использовании для любого уровня разработчика и аналитика, обладает большим выбором функций для работы с диаграммами. При работе на данной платформе доступна совместная работа над одним проектом, а также интеграция с программой IntelliJ IDEA.

В ArgoUML пользователи со многих стран создают диаграммы, так как программа поддерживает различные языки. Он подходит для начинающих разработчиков, в нем есть базовый набор функций, но нет расширенных возможностей. Функционал же возможно расширить с помощью плагинов.

Эти два инструмента удобны в их использовании. С помощью ArgoUML быстро и легко были созданы базовые диаграммы, такие как диаграмма развертывания, диаграмма вариантов использования. При работе с более сложными диаграммами использован инструмент Visual Paradigm. Они оба обладают преимуществами и подходят для достижения поставленных целей в дипломной работе.

2.6.2 Обоснование выбора инструментальных средств разработки программного обеспечения

Программа, написанная для приложения – это текст, состоящий из специальных терминов, символов, знаков. Первое время программы писали в текстовых редакторах. Сейчас же для продвинутых языков программирования и

даже для самых новых создаются инструментарии для написания программы. Для каждого языка существует большое количество интегрированных средств разработки (Integrated Development Environment), в которых программисты пишут, тестируют и запускают свои программы. Основным критерием выбора инструмента для работы программисты выбирают удобство.

Самые распространенные IDE – IntelliJ IDEA, Visual Studio и Eclipse.

Eclipse универсальный инструмент для разработки различных видов приложений, его можно адаптировать под цели проекта с помощью установленных плагинов.

От компании Microsoft представлена IDE – Visual Studio. Этот инструмент обладает большим количеством поддерживаемых фреймворков и технологий, включает в себя систему управления версиями, тестирование приложений. В ней работает достаточно широкий круг разработчиков.

IntelliJ IDEA – это самый распространенный вариант при выборе IDE. Все программисты, работающие с языком Java, пробовали работать в IDEA. Выбирают ее по многим причинам. Во-первых, все элементы, функции, инструменты расположены в удобном, понятном и логичном расположении в ее интерфейсе. Во-вторых, IntelliJ IDEA подходит как для опытных, так и для начинающих Java разработчиков, поскольку обладает большим количеством встроенных функций-помощников при написании кода. Программисту не нужно думать о верном расположении кода, в случае неверного написания IntelliJ IDEA подскажет и исправит проблему.

Выбор в качестве IDE IntelliJ IDEA основан на индивидуальных предпочтениях и возможности интеграции с другими инструментами разработки. Интерфейс и возможности IntelliJ IDEA понятны, а также опыт разработок в ней больше, чем в других программах.

2.6.3 Обоснование выбора языковых средств разработки

Любой язык программирования – это способ взаимодействия программиста с компьютером. У каждого языка своя концепция, правила,

лексика и реализация функций. Языков программирования огромное количество, исходя из цели разработки можно выбрать наиболее подходящий.

Выбор языка будет исходить от направления разработки, которая определена в цели дипломной работы. Для разработки веб приложения подходят такие языка, как Python, Java, Node.js, PHP.

Python – язык включает в себе большой набор библиотек и фреймворков, подходят не только для веб-разработок, но и для разработок игр и десктопных приложений. Однако, Python не подходит для масштабных и крупных проектов из-за его медленности выполнения.

Java является языком для высоконагруженных приложений, обладает высокой скоростью выполнения программ. Его тяжелее освоить, но выбирают язык за его преимущество в надежности кода.

Node.js позволяет писать серверные приложения, на нем пишут в совокупности с языком JavaScript на стороне сервера. Но обладает меньшей популярностью в использовании, чем Java или Python.

Одним из старейших языков является PHP. На данном языке написано множество сайтов разной сложности. Этот язык выбирают за его простоту и возможность работать с разными технологиями, включая базы данных и веб-серверы [9].

Для бэкенд части разработки был выбран язык Java из-за его надежности, высокой производительности, взаимодействия с выбранными фреймворками и программами для веб-приложения.

Для фронтенд части выбраны языки HTML и CSS. С помощью HTML языка написан интерфейс, включающий в себя расположение элементов на странице приложения. Команды на таком языке передают компьютеру информацию обо отображении содержимого. А CSS позволяет в разработке улучшать оформление с помощью подключаемых стилей, меняя размер заголовков, текстов, изменяя шрифты, цвета и размеры кнопок.

Удачный выбор языка программирования задаст верный старт для успеха задуманной разработки.

2.6.4 Обоснование выбора средств управления данными

Все данные разработки хранятся в базе данных. Под реляционной базой данных подразумевается упорядоченная информация, связанная между собой определенными отношениями. Внутри базы данных информация будет храниться в виде таблиц. Таблицы и заполнение этих таблиц хранятся в проекте в виде файлов с определенным форматом. Для управления изменения, удаления, добавления информации необходима специальная программа, которая могла бы создавать базу данных и управлять ею, иными словами, работать с файлами базы данных. Такой программой является СУБД – система управления базами данных.

Компании предоставляют свои СУБД в бесплатных и платных версиях. Среди всех СУБД по своей функциональности и популярности можно выделить следующие виды:

- Microsoft SQL Server;
- MySQL;
- Oracle;
- PostgreSQL.

Microsoft SQL Server – система управления базами данных от компании Microsoft. Она очень популярна в корпоративном секторе из-за чего расширенная версия доступная только платно. Однако, у SQL Server есть версия Express, которую можно использовать абсолютно бесплатно, например, для обучения или для разработки приложений, которые будут обрабатывать данные на небольших серверах размером до десяти гигабайт.

Oracle еще одна популярная СУБД. возможности Oracle database и Microsoft SQL Server сопоставимы, поэтому они являются серьезными конкурентами друг другу и стоимость их полнофункциональных версий очень высокая.

Также от компании Oracle создана бесплатная версия СУБД – MySQL. Ее доступность получила широкую распространенность среди программистов.

Большинство сайтов в интернете используют именно ее, как средство хранения данных.

PostgreSQL доступная версия для скачивания и работы с ней на любой операционной системе. Поддерживает разные типы и структуры данных, есть возможность создания собственных типов. PostgreSQL поддерживается большим активным сообществом пользователей и разработчиков [10].

Для работы с базой данных была выбрана СУБД PostgreSQL. За ее широкий спектр функциональных возможностей, высокую производительность, популярность и хорошие отзывы других программистов о работе в ней.

3 Разработка программного обеспечения приложения «Аудиторный гид»

3.1 Разработка базы данных

В среде Visual Paradigm был выполнен перенос модели в физическую модель данных (рисунок 15). У таблиц заданы имена атрибутов, типы данных, длины атрибутов, требуемые ограничения на допустимые значения, принимаемые по умолчанию значения атрибутов допустимость значения NULL.

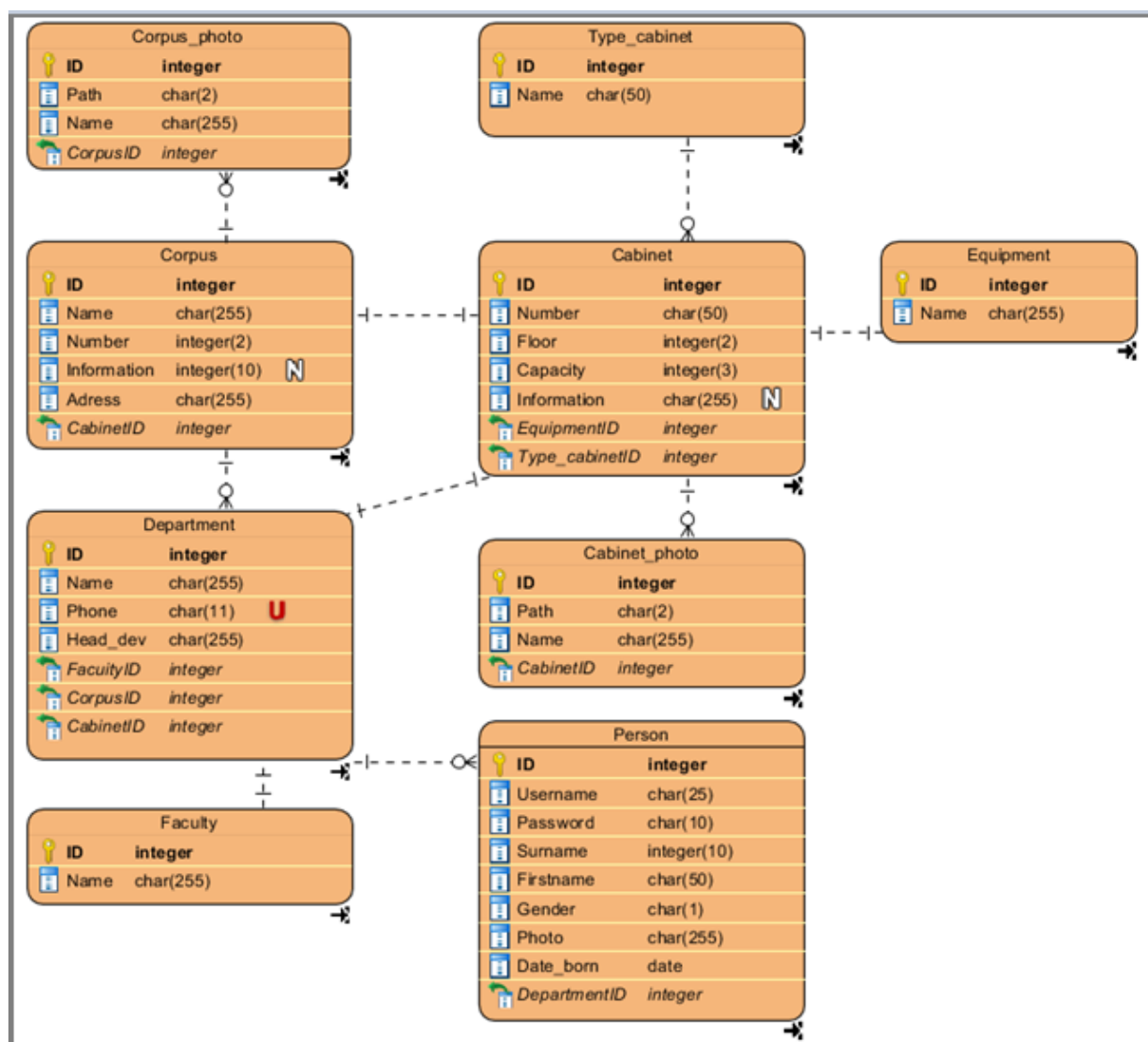


Рисунок 15 – Физическая модель данных предметной области

В выпускной квалификационной работе для работы с данными была выбрана база данных PostgreSQL.

Процесс создания базы данных начинается с создания самой базы данных. Это делается командой `CREATE DATABASE`. Далее создаются таблицы и расставляются связи между ними.

Команда `CREATE TABLE` позволяет создавать таблицы базы данных. В листинге 1 представлен фрагмент кода на создание таблицы «Корпус». Все остальные таблицы базы данных создаются аналогично и представлены в приложении Е (листинг Е.1).

Листинг 1 – Листинг для создания таблицы «Корпус»

```
CREATE TABLE IF NOT EXISTS Corpus
(
    id int primary key generated by default as identity,
    name varchar,
    number int,
    information varchar,
    address varchar
);
```

Результаты выполнения запросов на создание таблиц базы данных представлен на рисунке 16.

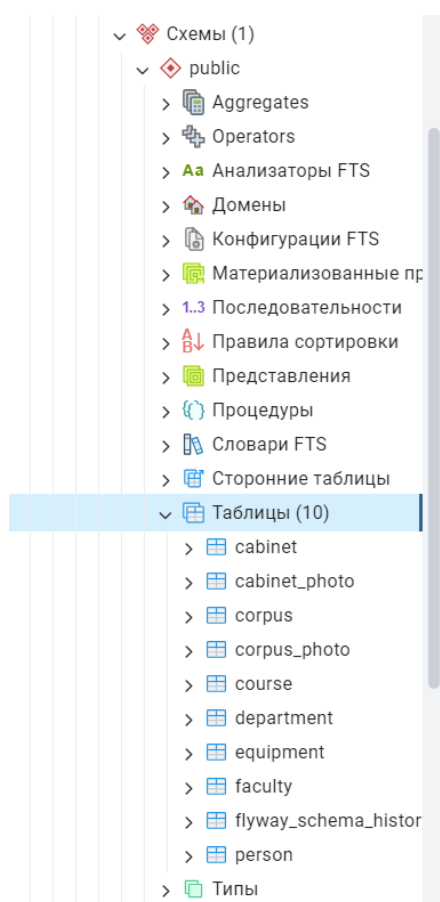


Рисунок 16 – Результат создания объектов базы данных

Процесс наполнения таблиц базы данных осуществлялся с помощью команды INSERT INTO. Заполнение каждой таблицы находится в соответствующем файле со скриптом. Тексты скриптов заполнения таблиц данными представлены в приложении Е (листинг Е.2 – листинг Е.9).

Результаты заполненных справочных таблиц «Корпус», «Фото корпуса» представлены на рисунке 17 и рисунке 18.

	id [PK] integer	name character varying	number integer	information character varying	address character varying
1	1	Учебный корпус №1	1	Главное здание ПГУ, где находится ректорат и другие службы	ул.Красная 40
2	2	Учебный корпус №2	2	В данном корпусе находится институт военного обучения	ул.Чкалова 57
3	3	Учебный корпус №3	3	В корпусе расположена обширная библиотека. В ней представлены книги по разным дисципли...	ул.Красная 46
4	4	Учебный корпус №4	4	Расположен корпус недалеко от лыжной базы и столовой ПГУ	ул.Красная 46
5	5	Учебный корпус №5	5	В этот корпус можно попасть по подземному переходу из первого корпуса	ул.Красная 40
6	6	Учебный корпус №6	6	Рядом с корпусом находится спортивная площадка	ул.Красная 40
7	7	Учебный корпус №7	7	В корпусе расположено студенческое пространство. Это неформальная площадка для студент...	ул.Лермонтова 26

Рисунок 17 – Таблица с заполненными данными «Корпус»

	id [PK] integer	path character varying	name character varying	corpus_id integer
1	1	/images/corpus/corpus_1.png	Фото корпуса 1	1
2	2	/images/corpus/corpus_2.png	Фото корпуса 2	2
3	3	/images/corpus/corpus_3.png	Фото корпуса 3	3
4	4	/images/corpus/corpus_4.png	Фото корпуса 4	4
5	5	/images/corpus/corpus_5.png	Фото корпуса 5	5
6	6	/images/corpus/corpus_6.png	Фото корпуса 6	6
7	7	/images/corpus/corpus_7.png	Фото корпуса 7	7
8	8	/images/corpus/corpus_8.png	Фото корпуса 8	8
9	9	/images/corpus/corpus_9.png	Фото корпуса 9	9
10	10	/images/corpus/corpus_10.png	Фото корпуса 10	10

Рисунок 18 – Таблица с заполненными данными «Фото корпуса»

Результаты заполненных справочных таблиц «Кафедра», «Оборудование» представлены на рисунке 19 и рисунке 20.

	id [PK] integer	name character varying	phone character varying (11)	head_dep character varying	faculty_id integer	corpus_id integer	cabinet_id integer
1	1	Сварочное, литейное производство и материаловеден...	88412666262	Розен Андрей Евгеньевич	1	3	2
2	2	Математическое обеспечение и применение ЭВМ	88412666592	Козлов Андрей Юрьевич	1	7	5
3	3	Техносферная безопасность	88412208402	Безборода Оксана Евгеньевна	1	8	1
4	4	Компьютерные технологии	88412666569	Горбаченко Владимир Иванович	3	7	3
5	5	Вычислительная техника	88412666589	Митрохин Максим Александрович	3	7	4
6	6	Анатомия человека	88412643972	Калмин Олег Витальевич	7	10	7
7	7	Хирургия	88412643964	Моисеева Инесса Яковлевна	7	10	6

Рисунок 19 – Таблица с заполненными данными «Кафедра»

	id [PK] integer	name character varying
1	1	Проектор
2	2	Компьютеры
3	3	Доска маркерная
4	4	Доска меловая
5	5	Лаборатория

Рисунок 20 – Таблица с заполненными данными «Оборудование»

Результаты заполненных справочных таблиц «Факультет», «Курс» представлены на рисунке 21 и рисунке 22.

	id [PK] integer	name character varying
1	1	Факультет промышленных технологий, электроэнергетики и транспор...
2	2	Факультет информационных технологий и электроники
3	3	Факультет вычислительной техники
4	4	Историко-фиологический факультет
5	5	Факультет физико-математических и естественных наук
6	6	Факультет педагогики, психологии и социальных наук
7	7	Лечебный факультет
8	8	Факультет стоматологии
9	9	Факультет физической культуры и спорта
10	10	Факультет международного сотрудничества

Рисунок 21 – Таблица с заполненными данными «Факультет»

	id [PK] integer	name character varying
1	1	Первый курс. Бакалавриат
2	2	Второй курс. Бакалавриат
3	3	Третий курс. Бакалавриат
4	4	Четвертый курс. Бакалавриат
5	5	Пятый курс. Бакалавриат
6	6	Первый курс. Специалитет
7	7	Второй курс. Специалитет
8	8	Третий курс. Специалитет
9	9	Четвертый курс. Специалитет
10	10	Пятый курс. Специалитет

Рисунок 22 – Таблица с заполненными данными «Курс»

Результаты заполненных справочных таблиц «Кабинет», «Фото кабинета» представлены на рисунке 23 и рисунке 24.

	id [PK] integer	number character varying (50)	floor integer	capacity integer	information character varying	type character varying	corpus_id integer	equipment_id integer
1	1	8-302a	3	6	Деканат ФПТЭиТ	Деканат	8	2
2	2	7a-510	5	7	Деканат КТ	Деканат	7	2
3	3	3-207	3	6	Деканат ФСЛПИМ	Деканат	3	2
4	4	7a-313	3	5	Деканат ВТ	Деканат	7	2
5	5	7a-323	3	6	Деканат МОиПЭВМ	Деканат	7	2
6	6	10-306	3	5	Деканат хирургии	Деканат	10	2
7	7	10-119	1	6	Деканат анатомии человека	Деканат	10	2
8	8	8-207	2	10	Приемная комиссия	Приемная комиссия	8	2
9	9	76-201	2	34	[null]	Лекционный	7	4
10	10	76-204	2	100	[null]	Лекционный	7	1

Рисунок 23 – Таблица с заполненными данными «Кабинет»

	id [PK] integer	path character varying	name character varying	cabinet_id integer
9	9	/images/cabinet/cabinet_9.png	Фото кабинета 9	9
10	10	/images/cabinet/cabinet_10.png	Фото кабинета 10	10
11	11	/images/cabinet/cabinet_11.png	Фото кабинета 11	11
12	12	/images/cabinet/cabinet_12.png	Фото кабинета 12	12
13	13	/images/cabinet/cabinet_1_evacuation.png	План эвакуации кабинета 1	1
14	14	/images/cabinet/cabinet_2_evacuation.png	План эвакуации кабинета 2	2
15	15	/images/cabinet/cabinet_3_evacuation.png	План эвакуации кабинета 3	3
16	16	/images/cabinet/cabinet_4_evacuation.png	План эвакуации кабинета 4	4
17	17	/images/cabinet/cabinet_5_evacuation.png	План эвакуации кабинета 5	5
18	18	/images/cabinet/cabinet_6_evacuation.png	План эвакуации кабинета 6	6
19	19	/images/cabinet/cabinet_7_evacuation.png	План эвакуации кабинета 7	7
20	20	/images/cabinet/cabinet_8_evacuation.png	План эвакуации кабинета 8	8

Рисунок 24 – Таблица с заполненными данными «Фото кабинета»

Наполненная база данных в дальнейшем будет использоваться для работы приложения «Аудиторный гид».

3.2 Разработка пользовательского интерфейса

Внешний вид приложения представлен в виде HTML страниц. Каждая страница имеет свой URL адрес. Стартовая страница имеет адрес <http://localhost:8080/main>. Стартовая страница имеет две области: 1 – область с информацией о корпусах, 2 – область осуществление поиска кабинета без фильтров (рисунок 25).

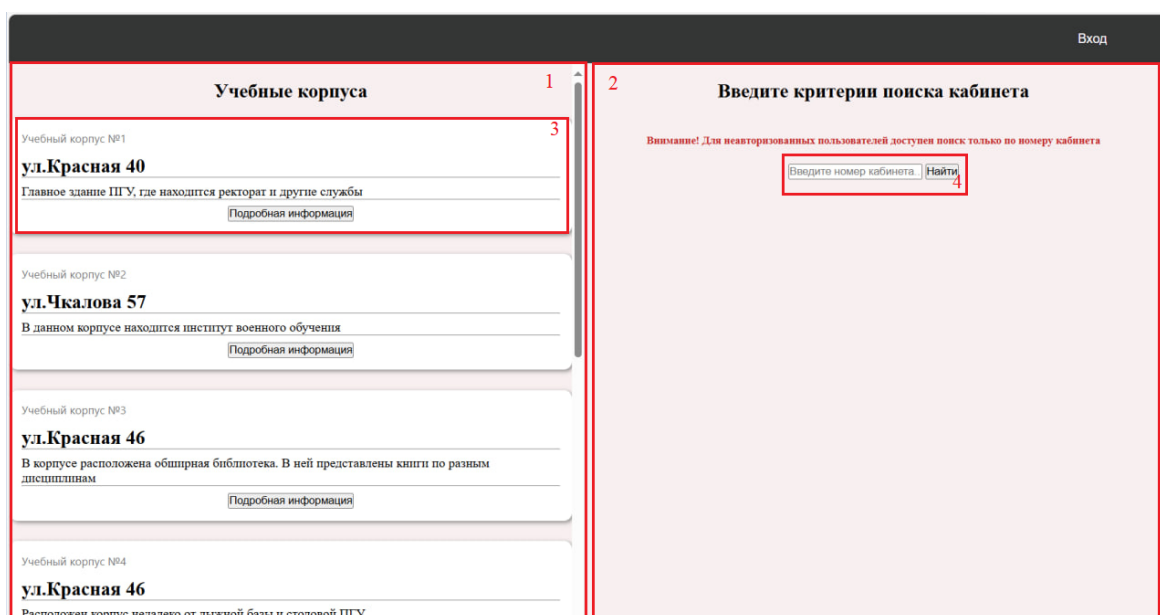


Рисунок 25 – Главная страница при запуске приложения

Элемент окна с наиболее важной информации о конкретном корпусе располагается под номером 3, а 4 – область ввода поиска кабинета по его номеру.

После выбора корпуса и нажатия на кнопку «Подробная информация» открывается страница с отображением подробной информации о корпусе. Основные элементы страницы: 1 – область отображения всей информации о выбранному корпусу, 2 – область с информацией о кафедрах, расположенных в выбранном корпусе, 3 – блок с контактной информации кафедры (рисунок 26).

Адрес страница имеет следующий: <http://localhost:8080/search/corpus/7..>

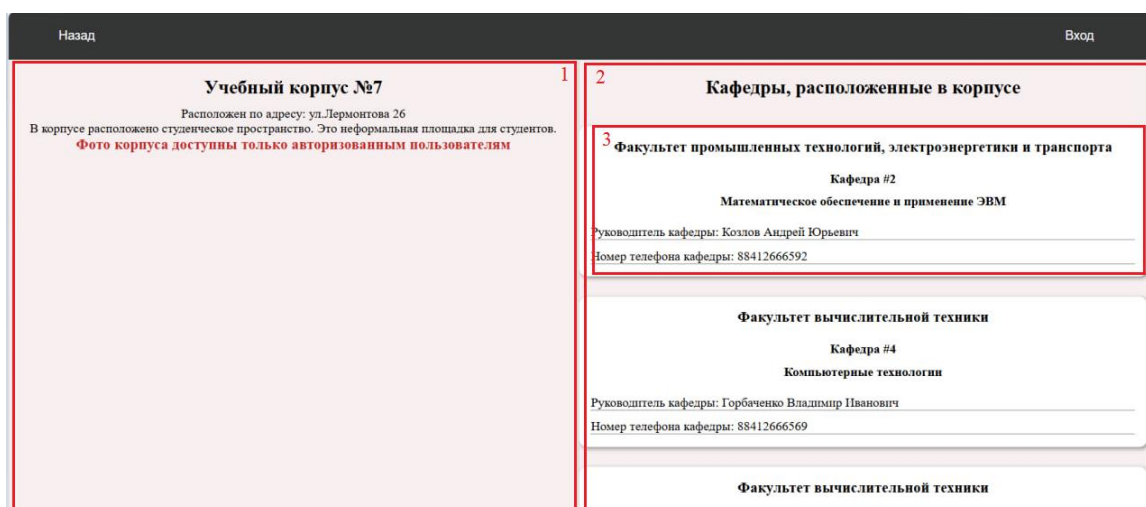


Рисунок 26 – Страница с информацией о корпусе

Для возвращения на главную страницу необходимо нажать кнопку «Назад», расположенную в шапке страницы. Для осуществления поиска необходимо ввести номер кабинета и нажать кнопку «Найти». Вместо пустой области отобразится блок с наиболее важной информацией найденного кабинета (рисунок 27).

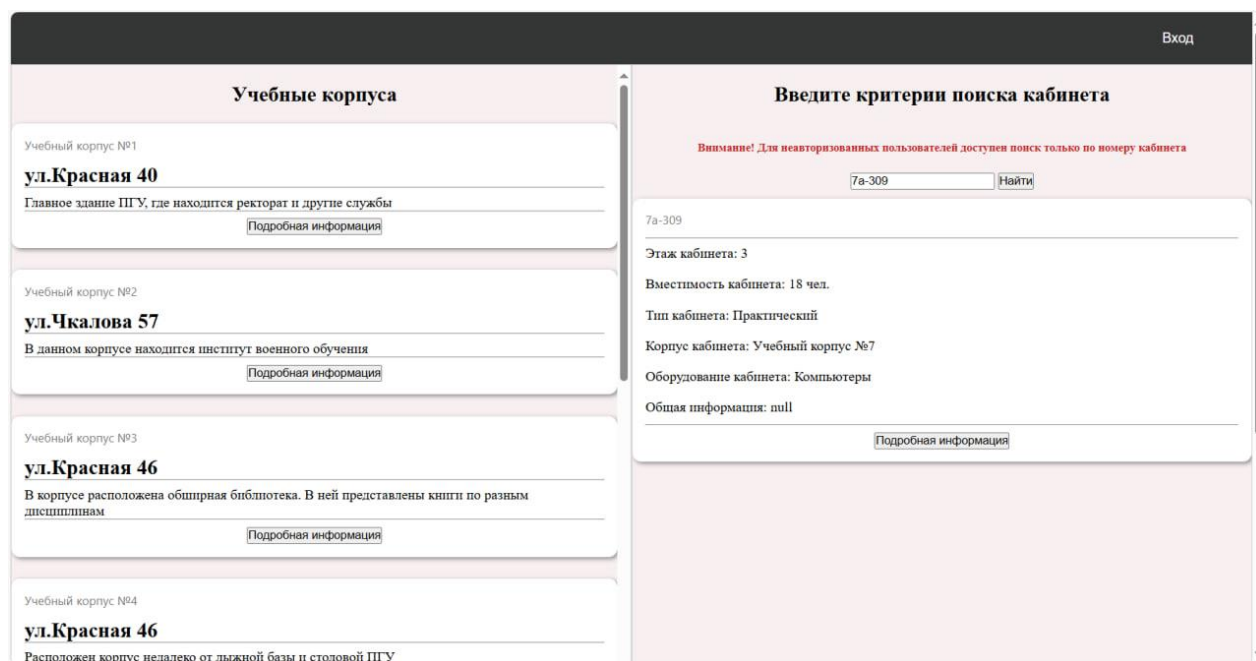


Рисунок 27 – Страница с поиском без фильтров

Красная надпись, как подсказка для пользователя, информирует о расширении возможностей при регистрации пользователя в системе приложения.

А по кнопке «Подробнее информация» пользователю доступен переход к странице информации о кабинете, где представлено больше данных о найденном кабинете.

Страница конкретного кабинета имеет URL <http://localhost:8080/search/cabinet/12?>. Окно состоит из текстовой справки о кабинете – область 1, и фотографий – область 2 (рисунок 28). Для возвращения на главную страницу необходимо нажать кнопку «Назад», расположенную в шапке страницы.

Назад
Вход

1

7а-309
Этаж кабинета: 3
Вместимость кабинета: 18 чел.
Тип кабинета: Практический
Корпус кабинета: Учебный корпус №7
Оборудование кабинета: Компьютеры
Общая информация

2

Фото кабинета 12

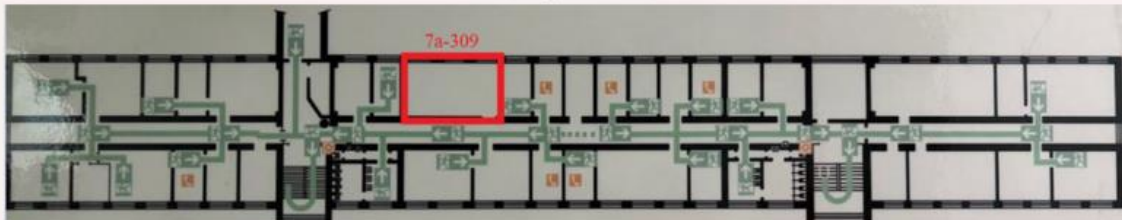

План эвакуации кабинета 12


Рисунок 28 – Страница с информацией о кабинете

Для доступа к расширенным функциям приложения пользователю необходимо зарегистрироваться (если не было регистрации пользователя) и авторизоваться в системе. Для перехода к регистрации необходимо нажать кнопку «Вход» в шапке страницы. Авторизация пользователя имеет <http://localhost:8080/auth/login> адрес и состоит из заполнения полей «Логина» и «Пароля» (рисунок 29).

Назад
Регистрация

Введите имя пользователя

Введите пароль

Войти

Рисунок 29 – Страница авторизации пользователя

Для перехода к регистрации пользователю нужно нажать кнопку «Регистрация» в шапке страницы. Страницы регистрации с URL адресом `http://localhost:8080/auth/registration` представляет из себя перечень полей для заполнения пользователем. Таких полей как логин, имя, фамилия, пол, дата рождения, пароль для входа (рисунок 30). Далее нажать кнопку «Зарегистрироваться».

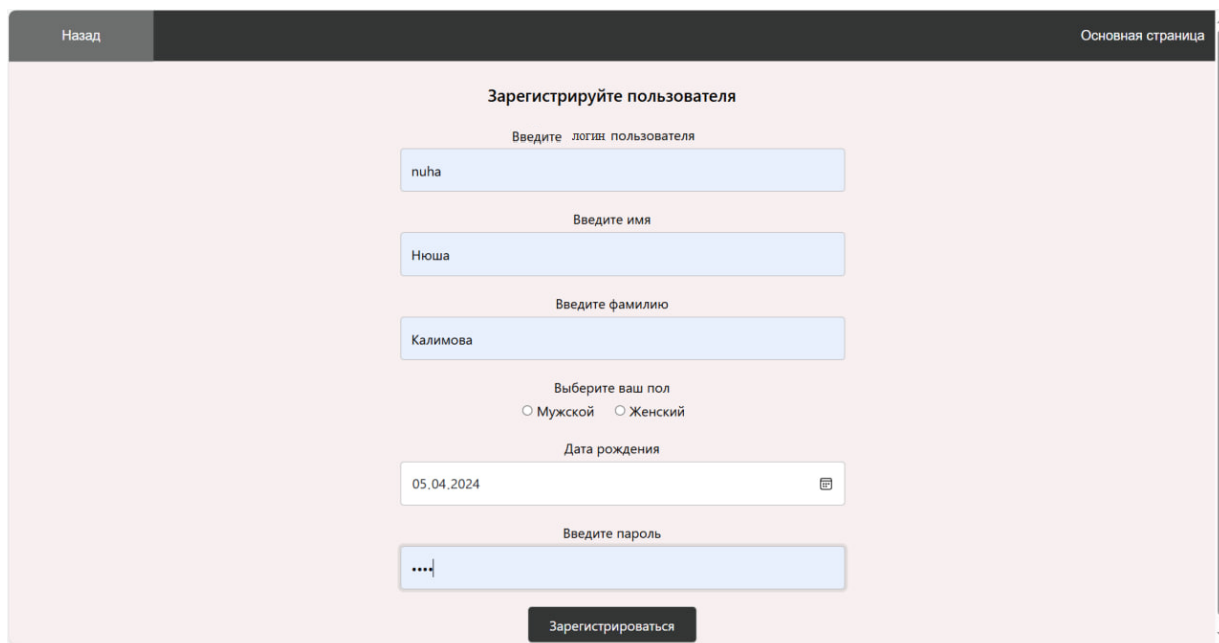


Рисунок 30 – Страница регистрации пользователя

После авторизации система отобразит окно авторизованного пользователя (рисунок 31).

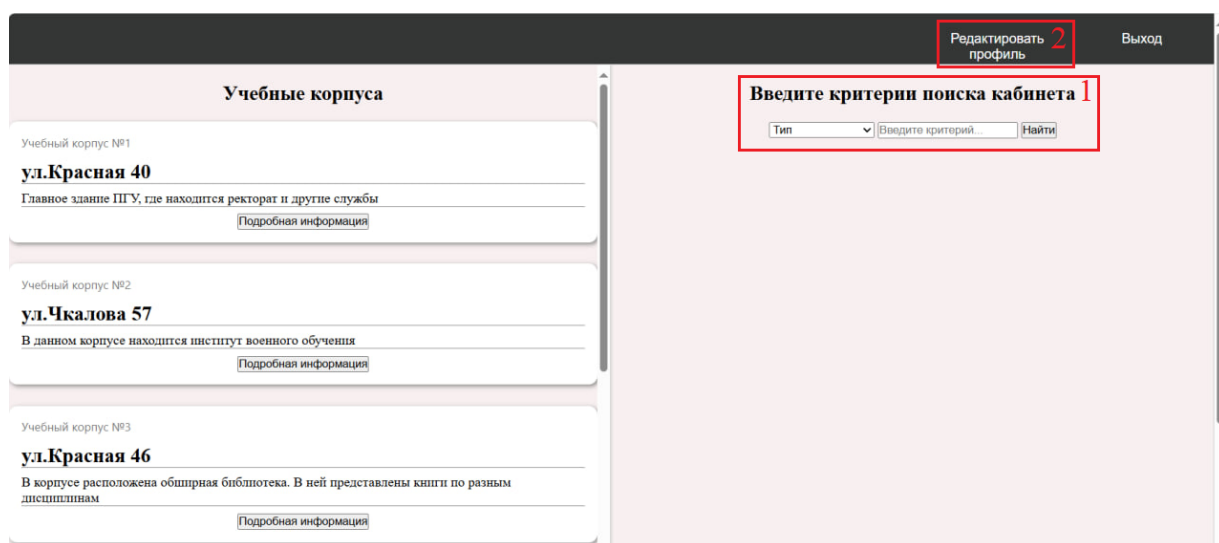


Рисунок 31 – Страница авторизованного пользователя

Страница авторизованного пользователя имеет схожую структуру с главной страницы, открывающейся при запуске программы. Однако имеет новые элементы 1 – область ввода поиска по фильтрам, 2 – кнопка для перехода в личный кабинет пользователя для редактирования своих данных. URL адрес – <http://localhost:8080/authorized/main>.

Для выполнения поиска необходимо выбрать фильтр поиска и ввести текст, по которому будет применен фильтр. Всего есть 6 типов фильтра. После выбора фильтра и ввода текста необходимо нажать кнопку «Найти» рядом с полем ввода текста (рисунок 32). После заполнения всех данных для поиска в зависимости от типа фильтра, будут отображаться подходящие варианты кабинетов.

Рисунок 32 – Выбор типа фильтра для поиска

При нажатии на кнопку «Подробная информация» о корпусе отобразится страница с предоставленными расширенными данными о корпусе (рисунок 33).

Рисунок 33 – Страница с информации о корпусе для авторизованного пользователя

А именно, отобразятся фотографии, относящиеся к конкретному корпусу по адресу страницы <http://localhost:8080/authorized/search/corpus/7?>.

Для осуществления функции редактирования личных данных необходимо нажать кнопку «Редактировать данные» в шапке чата. Страница редактирования состоит из таких личных данных как, фотография пользователя, логин, имя, фамилия, пол, дата рождения, пароль (рисунок 34). После нажатия кнопки «Сохранить» все введенные данные заменяются, база данных обновит личные данные пользователя. Страница редактирования профиля пользователя имеет следующий адрес – <http://localhost:8080/authorized/edit-profile/hanna>.

Рисунок 34 – Страница редактирования своих данных авторизованного пользователя

Используя вышеописанные HTML страницы, пользователь может выполнять все функции приложения.

3.3 Разработка программного обеспечения

Разработка программного обеспечения осуществлялась в среде разработки IntelliJ IDEA 2023.3.4 на языке программирования Java. Проект программы имеет сложную и вложенную структуру каталогов и файлов внутри них (рисунок 35).

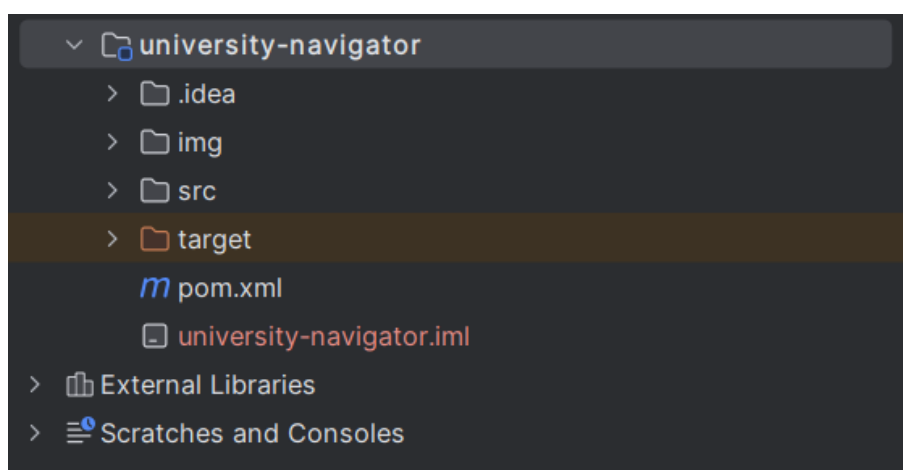


Рисунок 35 – Структура основных каталогов проекта

Каталог «.idea» хранит настройки проекта, такие как файлы .iml и workspace.xml. IntelliJ IDEA 2023.3.4 автоматически создает данную папку с файлами. vcs.xml отвечает за настройки системы контроля версий, modules.xml хранит расположения файлов «.iml» для каждого модуля, а workspace.xml некое хранилище с такими данными как, сведений о списке изменений, конфигурациях запуска, обновленных шаблонах конфигурации запуска, некоторых настройках Maven, точках останова и многом другом [11].

Каталог «img» создан для автоматической загрузки картинок, которые ставят пользователи себе на аватарку.

В каталог «target» IntelliJ IDEA обычно компилирует и собирает проект. Здесь находятся скомпилированные классы, а также собранные WAR файлы для развертывания. Данная папка создается автоматически при создании проекта.

Один из важных файлов проекта – pom.xml. Это файл конфигурации Maven проекта. В нем находятся данные о зависимостях, информация о сборке проекта, версии библиотек, настройки среды разработки. Это центральный файл Maven проекта, отвечающий за сборку, тестирование и управление приложением.

В каталоге «src» находятся исходные коды приложения. Он состоит из двух каталогов main, в котором расположены основные коды приложения, и resources, в котором находятся изображения проекта, файлы конфигурации и другие ресурсы приложения (рисунок 36).

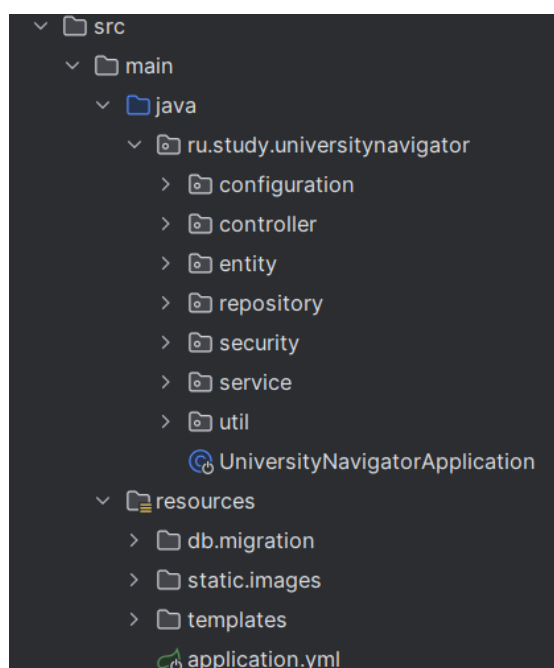


Рисунок 36 – Структура каталога src

Описание основных модулей каталога java приложения их назначения и ссылки на листинг представлены в таблице 10.

Таблица 10 – Назначение модулей каталога java

Название	Назначение	Ссылка на код модуля
Configuration	Модуль java, отвечающий за настройку базы данных, конфигурацию Spring Security, аутентификацию.	MvcConfig.java Приложение Ж (Листинг Ж.1) SecurityConfig.java Приложение Ж (Листинг Ж.2)
Controller	Модуль java, отвечающий за настройку страниц с действиями на них.	AuthController.java Приложение Ж (Листинг Ж.3) DataController.java Приложение Ж (Листинг Ж.4)
Entity	Модуль java, в котором описаны все поля таблицы базы данных. Каждая таблица описана в отдельном модуле. Его наличие позволяет работать с атрибутами таблиц базы данных.	Cabinet.java Приложение Ж (Листинг Ж.5) CabinetPhoto.java Приложение Ж (Листинг Ж.6) Corpus.java Приложение Ж (Листинг Ж.7) CorpusPhoto.java Приложение Ж (Листинг Ж.8) Course.java Приложение Ж (Листинг Ж.9) Department.java Приложение Ж (Листинг Ж.10) Equipment.java Приложение Ж (Листинг Ж.11) Faculty.java Приложение Ж (Листинг Ж.12) Person.java Приложение Ж (Листинг Ж.13)

Продолжение таблицы 10

Название	Назначение	Ссылка на код модуля
Repository	Модуль java, который необходим для взаимодействия с базой данных. Его наличие позволяет не писать запросы вручную.	CabinetPhotoRepository.java Приложение Ж (Листинг Ж.14) CabinetRepository.java Приложение Ж (Листинг Ж.15) CorpusPhotoRepository.java Приложение Ж (Листинг Ж.16) CorpusRepository.java Приложение Ж (Листинг Ж.17) DepartmentRepository.java Приложение Ж (Листинг Ж.18) EquipmentRepository.java Приложение Ж (Листинг Ж.19) PersonRepository.java Приложение Ж (Листинг Ж.20)
Security	Модуль java, отвечающий за получение данных аутентифицированного пользователя	PersonDetails.java Приложение Ж (Листинг Ж.21)
Service	Модуль java, который необходим для связи с репозиторием. В нем описана вся бизнес-логика для построения верной архитектуры.	CabinetPhotoService.java Приложение Ж (Листинг Ж.22) CabinetService.java Приложение Ж (Листинг Ж.23) CorpusPhotoService.java Приложение Ж (Листинг Ж.24) CorpusService.java Приложение Ж (Листинг Ж.25) DepartmentService.java Приложение Ж (Листинг Ж.26) PersonDetailsService.java Приложение Ж (Листинг Ж.27) RegistrationService.java Приложение Ж (Листинг Ж.28)
Utile	Модуль java, отвечающий за проверку существующего пользователя, а также для загрузки аватара пользователя	PersonValidator.java Приложение Ж (Листинг Ж.29) PhotoUploaderUtile.java Приложение Ж (Листинг Ж.30)
Templates	Модуль java, в котором описаны все html страницы приложения	cabinet.html Приложение Ж (Листинг Ж.31) corpus.html Приложение Ж (Листинг Ж.32) corpus-authorization.html Приложение Ж (Листинг Ж.33) edit-profile.html Приложение Ж (Листинг Ж.34) error.html Приложение Ж (Листинг Ж.35) login.html Приложение Ж (Листинг Ж.36) main.html Приложение Ж (Листинг Ж.37) main-authorization.html Приложение Ж (Листинг Ж.38) registration.html Приложение Ж (Листинг Ж.39)

Основной функцией приложения является поиск кабинета для получения подробной информации о нем и местонахождении в корпусе на этаже. Для наиболее наглядного представления алгоритма поиска при взаимодействии пользователя, программы и базы данных была использована диаграмма деятельности (рисунок 37).

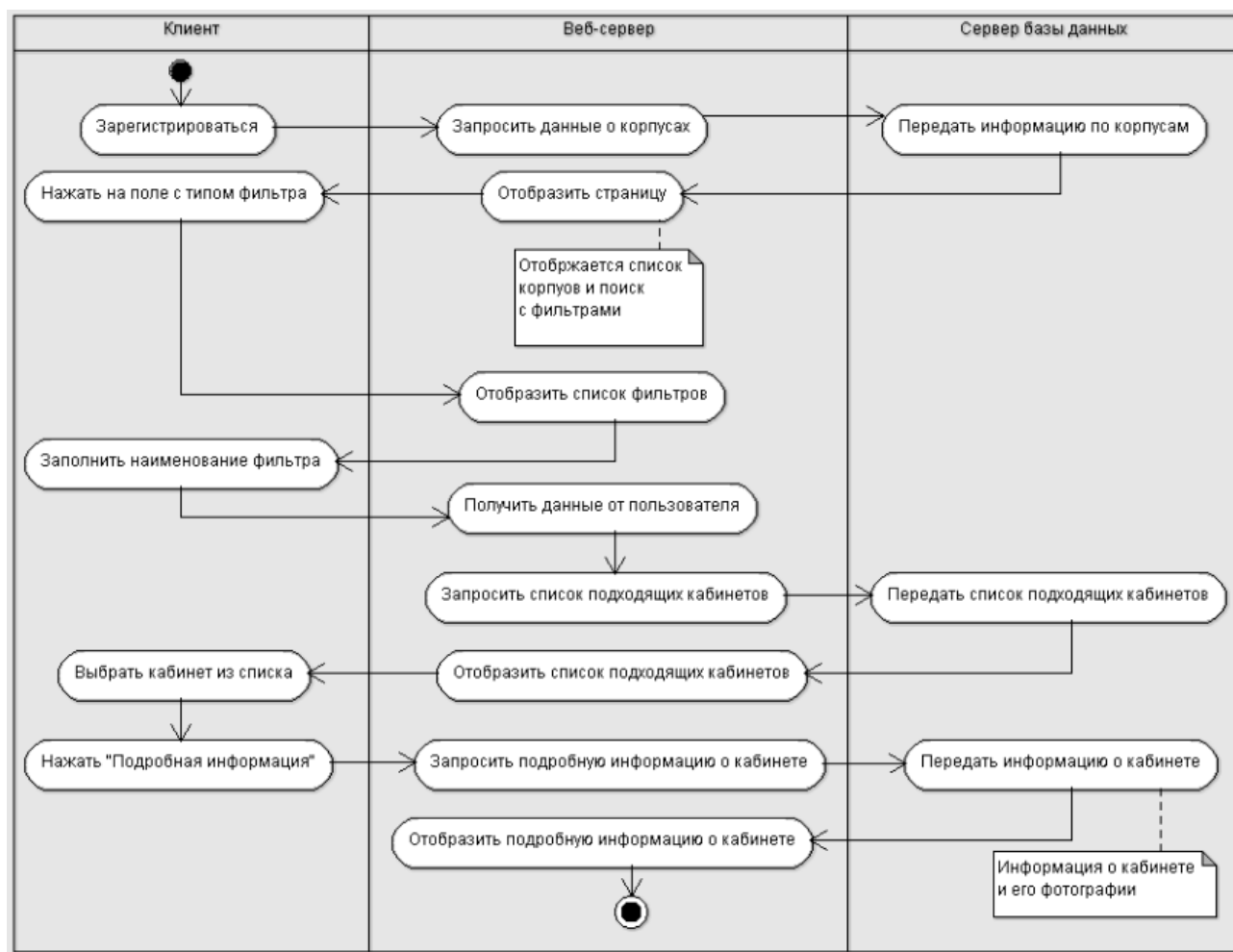


Рисунок 37 – Отображение функции поиска

Html страница, код которой представлен в приложении Ж (листинг Ж.38), отображает все действия пользователя и передает введенные пользователем данные. Полученные сервером данные от пользователя обрабатываются в коде DataController.java приложение Ж (листинг Ж.4) и запрашиваются у базы данных. Обработанный запрос базой данных передает серверу информацию, которая затем отображается той же самой html страницей.

Для получения, отображения и изменения данных из базы данных используются специализированные аннотации.

Аннотация `@PostMapping` используется для обработки HTTP Post-запросов.

В html коде страницы регистрации загружается страница с полями для регистрации пользователя (листинг 2).

Листинг 2 – Листинг для загрузки полей регистрации

```
<div id="block-content" style="max-width: 500px; margin-left: auto;
margin-right: auto; padding: 15px">
    <h1>Зарегистрируйте пользователя</h1>
    <form th:method="POST" th:action="@
        {/auth/registration}" th:object="${person}">
    <div class="input__wrapper">
    <label class="text-field__label" for="username">
        Введите имя пользователя</label>
    </div>
    ...
</form>
</div>
```

Код запроса добавления пользователя при регистрации находится в модуле `AuthController.java`. Фрагмент кода представлен в листинг 3.

Листинг 3 – Листинг post-запроса при регистрации пользователя

```
@PostMapping("/auth/registration")
public String performRegistration(@ModelAttribute("person") @Valid
    Person person, BindingResult bindingResult, Model model) {
    personValidator.validate(person, bindingResult);
    if (bindingResult.hasErrors()) {
        model.addAttribute("errors", bindingResult.
            getAllErrors());
        return "/error";
    }
    registrationService.register(person);
    return "login";
}
```

Данные со страницы передаются по аннотации в запрос на добавление данных в базу данных, с которой соединено приложение и предоставлен доступ по паролю и логину (листинг 4).

Листинг 4 – Листинг доступа к базе данных

```
spring:
    application:
        name: university-navigator
    datasource:
```

```
driverClassName: org.postgresql.Driver
url: jdbc:postgresql://localhost:5432/navigator_db
username: postgres
password: postgres
jpa:
  properties:
    hibernate:
      dialect: org.hibernate.dialect.PostgreSQLDialect
    hibernate:
      ddl-auto: none
      show-sql: true
upload:
  path: ${user.dir}\\img\\
```

Таким образом приложение «Аудиторный гид» был реализован с использованием языка программирования Java, CSS, HTML.

4 Контроль качества программного обеспечения приложения «Аудиторный гид»

4.1 Планирование процесса контроля качества программного обеспечения

4.1.1 Мероприятия по контролю качества программного обеспечения

Качество программного обеспечения – это совокупность характеристик, относящихся к способности удовлетворять ранее установленные потребности. То есть насколько система соответствует зафиксированным требованиям и ожиданиям пользователя или заказчика.

Контроль качества включает в себя тестирование и соответствие уровню качества продукта, который будут выпускать. Основная цель контроля качества предоставить обзор готовности продукта на разных этапах разработки. При положительной оценке контроля качества, можно утверждать, что требования соблюдены. В зависимости от специфики предметной области, можно выполнить следующие варианты по обеспечению контроля качества системы:

- тестирование документации;
- ревью кода на соответствие стандартам;
- внедрение методик по работе с качеством.

Мероприятия по контролю качества применялись на всех жизненных циклах программного обеспечения. Была проверена целостность базы данных, данные сначала были отобраны, а затем проведена проверка на согласованность информации между собой. Программный код просмотрен на наличие текстовых ошибок и верной пунктуации. Все этапы были задокументированы и сверены с поставленными требованиями и целями в начале проектирования приложения «Аудиторный гид» [12].

Контролирование качества программного обеспечения позволяет минимизировать появления дефектов и ошибок, повышает эффективность и

производительность системы за счет удовлетворения потребностей пользователя.

4.1.2 Обоснование выбора модели тестирования

Тестирование – это проверка системы перед тем, как будет пользоваться конечный пользователь. Тестирование выявляет ошибки, сбои и любые другие проблемы с функциями. Помимо визуальных ошибок, тестирование должно проверять, что система работает на различных платформах. Проверить работу с устройствами и убедиться в совместимости. Так, приложение будет обладать популярностью среди пользователей.

Тестирование проводится с помощью такого инструмента тестировщиков, как инструкция. Это документы, в которых описываются то, что нужно проверить, с какими условиями и действиями. Для каждой функции создается свой тест-кейсы, что позволяет удобно работать со всей информацией, видеть входные и выходные данные.

Существует два типа тестирования ручное и автоматизированное. В ручном тестировании тестировщик сам проходит по всем тест-кейсам и выполняет действия, описанные в них, а в автоматическом тестировщики пишут специальные программы, которые могут проверять функционал.

Видов тестирования очень много. Если классифицировать по целям, то существует функциональное и нефункциональное тестирование. В функциональном приложение проверяется на соответствие набора входных данных и ожидаемым выходным данным. При таком виде определяется правильная работа приложения всем ожидаемым бизнес-требованиям. При нефункциональном тестировании проверяется производительность, безопасность, удобство использования, масштабируемость [13].

Также существует различие в тестировании по сценариям. Позитивное тестирование проводится только с корректными данными для передачи в систему, негативное же с некорректными данными.

Наиболее распространенным типом функционального тестирования является тестирование белого ящика и черного ящика. Тестирование черного ящика основано на проверке ПО через интерфейсы. Белый ящик отвечает за тестирование внутренней работы программы. Ключом к успешному завершению тестирования является корректность данных [14].

При сравнении функционального и нефункционального вида тестирования можно выявить достоинства и недостатки каждого вида, исходя из которых будет сделан выбор типа тестирования приложения. Сравнение двух видов тестирования приведено в таблице 11.

Таблица 11 – Преимущества и недостатки функционального и нефункционального тестирования

Тип тестирования	Достоинства	Недостатки
Функциональное	<ol style="list-style-type: none"> 1. Ориентированность на конечного пользователя, так как проверяет соответствие работы программы ее заявленным функциональным требованиям. 2. Легкость проверки результатов и их анализа, что упрощает выявление дефектов и их исправление. 3. Простота автоматизации, что позволяет повторно и эффективно выполнять их на различных этапах разработки. 	<ol style="list-style-type: none"> 1. Возможность упущения логических ошибок в программном обеспечении. 2. Вероятность избыточного тестирования [15].
Нефункциональное	<ol style="list-style-type: none"> 1. Полнота покрытия позволяет учитывать нефункциональные аспекты, такие как производительность, надежность, безопасность и прочие нефункциональные требования. 2. Проверка качества позволяет обнаружить некоторые дефекты, связанные с нестандартными сценариями использования или внутренними механизмами программы. 	<ol style="list-style-type: none"> 1. Сложность проверки результатов. 2. Необходимость в привлечении дополнительных ресурсов и затраты на время.

Таким образом, для тестирования приложения было использовано тестирование функциональное тестирование. Его выбор обоснован качественной

проверкой функциональных требований, что определит ошибки и дефекты функциональности приложения. Такое тестирование не требует наличие дополнительных программ для проведения тестирования и позволит проверить основные действия, которые выполняет пользователь.

4.1.3 Обоснование выбора метрик программного кода

Анализ программного обеспечения можно проводить не только тестированием, но и, рассматривая те элементы, которые можно извлечь из ПО. Этими элементами являются метрики кода – это показатели качества программного кода. Метрики кода можно определить с помощью специальных инструментов, а можно посчитать вручную.

Один из частых показателей, которые используют разработчики для проверки кода – это цикломатическая сложность. Цикломатическая сложность представляет из себя числовое значение, представляющее сложность функции или конкретного метода. Для определения необходимо нарисовать граф потока управления и определить количество дуг, вершин и число компонентов связности. Дуги – это пути между вершинами, а вершины представляют узлы между дугами. При получившемся большом числе цикломатической сложности можно сделать вывод, что требуется провести больше тестов для достижения покрытия кода [16].

Показатель количества классов в пакете и количества строк исходного кода в каждом пакете тесно связаны между собой. Количество классов и строк дает хороший показатель с точки зрения размера. Сами по себе цифры не дают информацию, но отслеживая их в каждом коде и анализируя динамику развития, можно сделать вывод о повышении эффективности написания программного кода разработчиком.

Еще одной из распространенных метрик кода является глубина наследования. Для определения числового значения чаще всего используют граф наследования, где проще подсчитать общее количество наследуемых классов.

Высокое число глубины наследования говорит о высоком риске возникновении ошибок и сложности кода. Низкое число определяет меньшую сложность кода и меньшее использование наследования [17].

Для оценки качества кода с помощью метрик кода была выбрана комбинация из трех наиболее распространенных показателей:

- метрика SLOC (Source Lines of Code);
- метрика Холстеда;
- метрика Чепина.

Метрика SLOC подсчитывает объем кода, что является важным в разработке для оценки сложности и объема. Метрика Холстеда дает представление о понятности и сложности программы. А третий показатель – метрика Чепина помогает в выявлении потенциальных проблем, определяя степень структурированности кода.

Сочетание этих метрик помогает наиболее лучше представить разрабатываемый продукт, его характеристики и провести качественное тестирование приложения.

4.2 Тестирование программных средств

4.2.1 Тестирование черным ящиком

Для тестирования был выявлен класс эквивалентности с номером кабинета. Номер кабинета имеет сложную составную структура. Должен содержать 5-6 символов: цифры, русские буквы, дефис. Должен начинаться с цифры, обозначающей корпус, в котором находится кабинет. Цифра может быть двузначной или однозначной, однозначная цифра может иметь русскую букву, обозначающее крыло корпуса. Далее следует обязательный символ дефис. После него следует цифра, означающая этаж кабинета, и две последние цифры непосредственно номер кабинета на этаже.

Табличное представление классов эквивалентности представлено в таблице 12.

Таблица 12 – Табличное представление классов эквивалентности

	Правильные КЭ	Неправильные КЭ
Длина номера кабинета	5-6 (1)	0-4 (2) 7-∞ (3)
Структура	Начинается с цифры (4) Третий или второй символ дефис (5) 4,5,6 символы цифры (6)	Начинается не с цифры (7) Третий или второй символ любой кроме дефиса (8) 3,4,5 символы любые символы кроме цифр (9)
Содержание	Цифры (10) Русские буквы (11) Дефис (12)	Остальное (13)
Обязательность поля	Да (14)	Нет (15)

При рассмотрении графического представления классов эквивалентности можно рассмотреть длину номера кабинета (рисунок 38) и структуру кабинета (рисунок 39).



Рисунок 38 – Длина номера кабинета

Цифра			Цифра	Цифра	Цифра
-------	--	--	-------	-------	-------

Рисунок 39 – Структура номера кабинета

На рисунке 40 представлена неверная структура номера кабинета противоположная верной.

Не цифра			Не цифра	Не цифра	Не цифра
----------	--	--	----------	----------	----------

Рисунок 40 – Неверная структура номера кабинета

В виде кругов Эйлера отображено содержание номера кабинета, то есть из каких символов может быть представлено наименование кабинета в корпусе (рисунок 41).



Рисунок 41 – Содержание номера кабинета

Так как структура рассмотрена множеством, поэтому стоит предусмотреть тесты непересекающихся и пересекающихся множеств. Тест-кейсы представлены в таблице 13.

Таблица 13 – Тест-кейсы

№	КЭ	Входные данные	Выходные данные
0	Все правильные 1,4,5,6,10,11,12,14	76-324	Успешно
1	2, 4,5,6,10,11,12,14	76-3	Ошибка: номер кабинета содержит меньше 5 символов
2	3,4,5,6,10,11,12,14	76-3246	Ошибка: номер кабинета содержит больше 6 символов
3	1,7,5,6,10,11,12,14	66-324	Ошибка: номер кабинета начинается не с цифры
4	1,4,8,6,10,11,12,14	766324	Ошибка: номер кабинета не содержит символ дефис
5	1,4,5,9,10,11,12,14	76-666	Ошибка: последние символы не содержат цифр
6	1,4,5,6,13,14	Sw*qrn	Ошибка: неверный формат записи номера кабинета
7	1,4,5,6,10,11,12,15	-	Ошибка: заполните обязательное поле номера кабинета

Таким методом были протестированы поля логина и пароля при авторизации пользователя, а также заполняемое значение фильтра при выборе типа для поиска кабинета. Функциональное тестирование черным ящиком показало корректную работу системы.

4.2.2 Тестирование белым ящиком

С помощью тестирования белым ящиком были проверены такие функции, как поиск по наименованию, авторизация и регистрация пользователя, редактирование личных данных, просмотр информации о корпусах и кабинетах разными видами пользователя, а также поиск кабинета по фильтрам.

Для примера описания тестирования методом белого ящика был взят фрагмент кода с реализацией функцией подбора кабинета с помощью фильтров: по номеру корпуса, по типу кабинета, по этажу, оборудованию и по количеству мест (листинг 5).

Листинг 5 – Отрывки кода с подбором кабинета по фильтрам

```
//в интерфейсе
<div class="searchbar">
  <h2 class="searchbar-header">Введите критерии поиска кабинета</h2>
  <select name="criteria">
    <option value="type">Тип</option>
    <option value="number">Номер Кабинета</option>
    <option value="corpus">Корпус</option>
    <option value="floor">Этаж</option>
    <option value="equipment">Оборудование</option>
    <option value="capacity">Вместимость</option>
  </select>
  <input type="text" name="value" placeholder="Введите критерий...">
  <div th:if="${cabinets != null and not #lists.isEmpty(cabinets)}">
    <div th:each="cabinet : ${cabinets}">
      <div class="cabinets-box">
        ...
      </div>
    </div>
  </div>
</div>
//в функции
@GetMapping("/authorized/search")
public String searchByNumber(@RequestParam("criteria") String criteria,
    @RequestParam("value") String value, RedirectAttributes
    redirectAttributes) {
    List<Cabinet> cabinets = new ArrayList<>();
    switch (criteria) {
        case "type":
```

```

        cabinets = cabinetService.findByType(value.trim());
        break;
    case "capacity":
        cabinets = cabinetService.findByCapacity
            (Integer.parseInt(value.trim()));
        break;
    case "number":
        cabinets = cabinetService.findByNumber(value.trim());
        break;
    case "floor":
        cabinets = cabinetService.findByFloor
            (Integer.parseInt(value.trim()));
        break;
    case "equipment":
        cabinets = cabinetService.findByEquipment(value.trim());
        break;
    case "corpus":
        cabinets = cabinetService.findByCorpus(value.trim());
        break;
}
redirectAttributes.addFlashAttribute("cabinets", cabinets);
return "redirect:/authorized/main";

```

Блок схема показывает последовательность операций в функции поиска (рисунок 42).

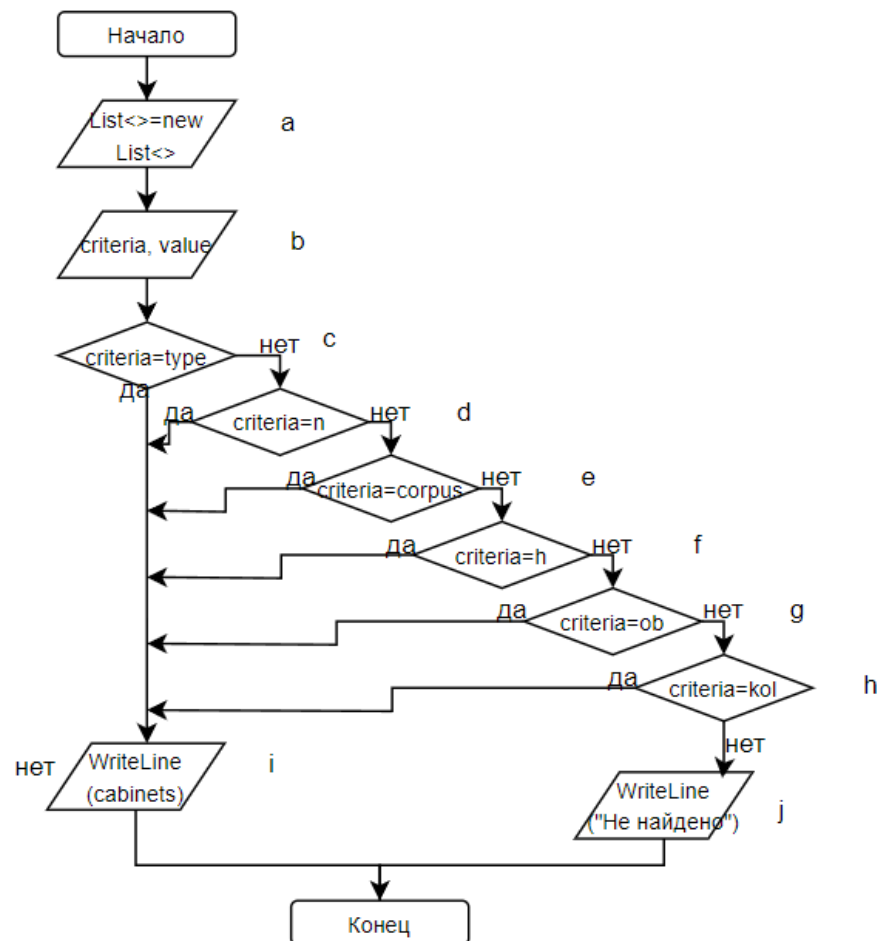


Рисунок 42 – Диаграмма функции подбора кабинета с помощью фильтров

Были рассмотрены несколько критериев и составлены соответствующие таблицы. Для критерия покрытия операторов подразумевается выполнение каждого оператора программы по крайней мере один раз (таблица 14).

Таблица 14 – Покрытие операторов

№	Операторы	Входные данные	Выходные данные
1	a,b,i,j	criteria = type n corpus h ob kol	Cabinets<cabinet1, cabinet2...cabinetn>
2	a,b,i,j	criteria != type n corpus h ob kol	Сообщение: «Не найдено»

По критерию покрытия решений необходимо составить такое число тестов, при которых каждое условие в программе примет как истинное значение, так и ложное значение (таблица 15).

Таблица 15 – Покрытие решений

№	Решения	Входные данные	Выходные данные
1	c – да d – да e – да f – да j – да h – да	criteria = type n corpus h ob kol	Cabinets<cabinet1, cabinet2...cabinetn>
2	c – нет d – нет e – нет f – нет j – нет h – нет	criteria != type n corpus h ob kol	Подходящих кабинетов не найден по данному набору фильтров.

В критерий покрытия условий записывается число тестов достаточное для того, чтобы все возможные результаты каждого условия в решении были выполнены по крайней мере один раз (таблица 16).

Таблица 16 – Покрытие условий

№	Условие	Входные данные	Выходные данные
1	c – да d – нет e – нет f – нет j – нет h – нет	criteria = type n corpus h ob kol	Cabinets<cabinet1, cabinet2...cabinetn>
2	c – нет d – да e – нет f – нет j – нет h – нет	criteria = type n corpus h ob kol	Cabinets<cabinet1, cabinet2...cabinetn>

Продолжение таблицы 16

№	Условие	Входные данные	Выходные данные
3	c – нет d – нет e – да f – нет j – нет h – нет	criteria = type n corpus h ob kol	Cabinets<cabinet1, cabinet2...cabinetn>
4	c – нет d – нет e – нет f – да j – нет h – нет	criteria = type n corpus h ob kol	Cabinets<cabinet1, cabinet2...cabinetn>
5	c – нет d – нет e – нет f – нет j – да h – нет	criteria = type n corpus h ob kol	Cabinets<cabinet1, cabinet2...cabinetn>
6	c – нет d – нет e – нет f – нет j – нет h – да	criteria = type n corpus h ob kol	Cabinets<cabinet1, cabinet2...cabinetn>
7	c – нет d – нет e – нет f – нет j – нет h – нет	criteria != type n corpus h ob kol	Подходящих кабинетов не найдено по данному набору фильтров.

Для покрытия условий и решений необходимо составить тесты так, чтобы результаты каждого условия выполнялись хотя бы один раз, результаты каждого решения так же выполнялись хотя бы один раз, и каждый оператор должен быть выполнен хотя бы один раз (таблица 17).

Таблица 17 – Покрытие условий и решений

№	Условие	Входные данные	Выходные данные
1	a, b, c(да), d(нет), e(нет), f(нет), g(нет), i(нет), j	criteria = type n corpus h ob kol	Cabinets<cabinet1, cabinet2...cabinetn>
2	a, b, c(нет), d(да), e(нет), f(нет), g(нет), i(нет), j	criteria = type n corpus h ob kol	Cabinets<cabinet1, cabinet2...cabinetn>
3	a, b, c(нет), d(нет), e(да), f(нет), g(нет), i(нет), j	criteria = type n corpus h ob kol	Cabinets<cabinet1, cabinet2...cabinetn>
4	a, b, c(нет), d(нет), e(нет), f(да), g(нет), i(нет), j	criteria = type n corpus h ob kol	Cabinets<cabinet1, cabinet2...cabinetn>
5	a, b, c(нет), d(нет), e(нет), f(нет), g(да), i(нет), j	criteria = type n corpus h ob kol	Cabinets<cabinet1, cabinet2...cabinetn>

Продолжение таблицы 17

№	Условие	Входные данные	Выходные данные
6	a, b, c(нет), d(нет), e(нет), f(нет), g(нет), i(да), j	criteria = type n corpus h ob kol	Cabinets<cabinet1, cabinet2...cabinetn>
7	a, b, c(нет), d(нет), e(нет), f(нет), g(нет), i(нет), j	criteria != type n corpus h ob kol	Подходящих кабинетов не найден по данному набору фильтров.

Для графического представления потока управлений был сформирован граф потока данных (рисунок 43).

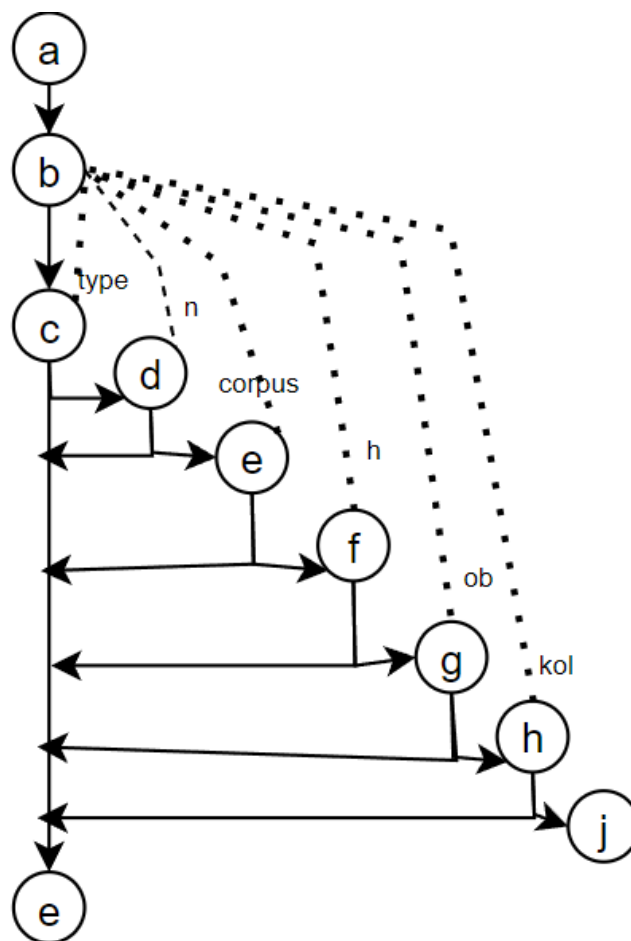


Рисунок 43 – Граф потока данных

Тестирование методами черного и белого ящика показало, что ошибок в работе программы не найдено.

4.3 Оценка качества программного кода на основе метрик

4.3.1 Метрика SLOC

Вычислив метрику SLOC, в рассматриваемом листинге 5 имеются:

- 50 физических строк кода;
- 39 логических строк кода;
- 2 строк комментариев.

Это нормальные значения для одной функции.

Для вычисления цикломатической сложности $V(G)$ применяется следующая формула:

$$V(G) = e - n + 2p,$$

где e – количество дуг,

n – количество вершин,

p – число компонент связности.

Граф потока управления содержит 10 вершин, 14 дуг и один компонент связности. Значение цикломатической сложности, вычисленной по формуле, равно 3 ($V(G) = 14 - 10 + 2 = 6$). Это значение не превышает 10, следовательно, функция не является структурно ложной.

4.3.2 Метрика Холстеда

Сложность и объем кода можно измерить по метрике Холстеда. В основе этого метода лежит система оценки характеристик программы. Операторы и операнды, используемые в программном коде, приведены в таблице 18 и таблице 19.

Таблица 18 – Операторы

Операторы	Число вхождений
=	23
If	1
new	1
!	1
.	15
List	1

Продолжение таблицы 18

Операторы	Число вхождений
:	7
;	15
{ }	4
«»	17
@	3
and not	1
n1 = 12	N1 = 89

Таблица 19 – Операнды

Операнды	Число вхождений
class	3
name	2
value	6
type	1
placeholder	1
cabinets	11
cabinet	2
searchByNumber	6
RequestParam	2
criteria	2
cabinetService	6
type	2
capacity	2
number	2
floor	2
equipment	2
corpus	2
n2 = 18	N2 = 54

Длина программы рассчитывается по формуле:

$$N = N1 + N2 = 89 + 54 = 143,$$

где N1 – число вхождений операторов,

N2 – число вхождений операндов.

Словарь программы рассчитывается по формуле:

$$n = n1 + n2 = 12 + 18 = 30,$$

где n1 – количество операторов,

n2 – количество операндов.

Объем программы рассчитывается по формуле:

$$HV = N * \log_2(n) = 143 * 5 = 715,$$

где N – длина программы,

n – словарь программы.

Сложность программы рассчитывается по формуле:

$$HD = (n1 / 2) * (N2 / n2) = (12 / 2) * (54 / 18) = 18,$$

где n1 – количество операторов,

N2 – число вхождений операндов,

n2 – количество операндов.

Уровень программы имеет значение ниже среднего по данным измерений из книги Холстеда.

4.3.3 Метрика Чепина

По метрике Чепина все переменные можно разделить на 4 функциональные группы:

- Р – вводимые переменные для расчетов и для обеспечения вывода;
- М – модифицируемые, или создаваемые внутри переменные;
- С – переменные, участвующие в управлении работой программного модуля (управляющие переменные);
- Т – не используемые в программе («паразитные») переменные.

По формуле измеряется значение метрики Чепина:

$$Q = a1 * P + a2 * M + a3 * C + a4 * T,$$

где a1, a2, a3, a4 – весовые коэффициенты.

В рассматриваемой функции $P = 3$, $M = 5$, $C = 6$, $T = 0$, весовые коэффициенты $a1 = 1$, $a2 = 2$, $a3 = 3$, $a4 = 0.5$, а уравнение принимает вид $Q = 3 + 2 * 5 + 3 * 6 + 0.5 * 0 = 31$.

Значение метрики Чепина равняется 11. Если в результате изменения рассматриваемых функций значение метрики значительно увеличиться, это будет означать низкое качество изменений.

В ходе тестирования приложения «Аудиторный гид» было выполнено функциональное тестирование белым и черным ящиком. Результаты различных видов тестирования показали, что программное обеспечение работает исправно. Оценка качества программного кода на основе метрик показало эффективную работу сервиса. Из всего сказанного следует, что контроль качества программного обеспечения прошел успешно.

Заключение

В ходе выполнения выпускной квалификационной работы, посвященной проектированию приложения для навигации людей по университетскому кампусу, было разработано приложение «Аудиторный гид». Приложение помогает в ориентации внутри корпусов и на территории кампуса, осуществляет поиск кабинета по фильтрам, предоставляет подробную информацию о кабинете и кафедрах, расположенных в корпусах.

Для разработки приложения была проанализирована предметная область, включающая в себя информацию о внутреннем устройстве университетов, о корпусах, о их наполнении, о территориальном наполнении университетского кампуса, об иерархии факультетов, кафедр, специальностей.

Найдены и сравнены приложения аналоги как российских университетов, так и зарубежных. Выявлены преимущества и недостатки, которые были учтены при разработке данного приложения.

Было произведено планирование системы для навигации людей, определены функциональные требования и требования к архитектуре. Для реализации были выбраны такие инструменты, как IntelliJ IDEA, PostgreSQL и язык программирования java. Также был спроектирован и продуман интерфейс пользователя.

Разработанное приложение протестировано с помощью функционального тестирования методами черного и белого ящика, а также с помощью метрик кода. В результате тесты показали безошибочную работу приложения для конечного пользователя.

Разработанное приложения будет актуально и полезно для использования людей, кто плохо знаком с расположением корпусов и кабинетов внутри них на территории университета. Предоставление структурированной информации поможет пользователям быстро находить нужную информацию.

Приложение можно будет развивать и расширять по функционалу. Планируется добавить маршрутизацию между корпусами, информацию о

кабинетах по другим корпусам, а также модернизировать внешний вид для еще более удобного использования. Приложение может быть добавлено в официальный сайт Пензенского Государственного университета, что повысит удобство пользования сайта и добавит новый функционал для использования.

Список используемой литературы

1. Сведения об образовательной организации [Электронный ресурс]. URL: <https://pnzgu.ru/sveden/struct> (дата обращения 20.02.2024)
2. Официальный сайт «Российской академии при Президенте Российской Федерации» [Электронный ресурс]. URL: <https://www.ranepa.ru/ob-akademii/infrastruktura/uchebnye-korpora/> (дата обращения 24.02.2024)
3. Официальный сайт «МГУ им. М.В. Ломоносова» [Электронный ресурс]. URL: <https://msu.ru/info/map/> (дата обращения 24.02.2024)
4. Навигация по университету «University of Cambridge» [Электронный ресурс]. URL: <https://map.cam.ac.uk> (дата обращения 24.02.2024)
5. Официальный сайт «Stanford University» [Электронный ресурс]. URL: <https://bgsha.ru/sveden/common/?ysclid=lvgkdvayj4953955489> (дата обращения 24.02.2024)
6. Фаулер, М. UML. Основы / 3 е издание. – Пер. с англ. – СПб: Символ Плюс, 2004 – 192 с.].
7. Аналоги Visual Paradigm [Электронный ресурс]. URL: <https://ruprogi.ru/software/visual-paradigm?ysclid=lvtizp9gqk34473823> (дата обращения 24.04.2024)
8. 30 лучших программ для моделирования диаграмм UML: [Электронный ресурс]. URL: <https://www.cybermedian.com/ru/top-30-best-uml-modeling-software-2022/> (дата обращения 24.04.2024)
9. Языки программирования для создания сайтов: [Электронный ресурс]. URL: <https://ru.hexlet.io/blog/posts/yazyki-programmirovaniya-dlya-sozdaniya-saytov?ysclid=lvuz6aq76b629876949> (дата обращения 25.04.2024)
10. СУБД PostgreSQL: почему её стоит выбрать для работы с данными: [Электронный ресурс]. URL: <https://practicum.yandex.ru/blog/chto-takoe-subd-postgresql/> (дата обращения 25.04.2024)
11. Maven проект на java [Электронный ресурс]. URL: https://translated.turbopages.org/proxy_u/en-ru.ru.143ac951-66598ae1-9d995f03-

74722d776562/https://www.baeldung.com/intellij-idea-directory (дата обращения 15.06.2024)

12. Контроль качества программного обеспечения [Электронный ресурс]. URL: https://translated.turbopages.org/proxy_u/en-ru.ru.18310981-665f0f4a-b92e4c39-74722d776562/https/en.wikipedia.org/wiki/Software_quality_Control (дата обращения 20.06.2024)

13. Функциональное и нефункциональное тестирование [Электронный ресурс]. URL: https://translated.turbopages.org/proxy_u/en-ru.ru.d82b0292-665f5e22-ab200912-74722d776562/https/www.baeldung.com/java-functional-vs-non-functional-testing (дата обращения 20.06.2024)

14. Тестирование методом черного ящика [Электронный ресурс]. URL: <https://coderlessons.com/tutorials/kachestvoprogrammnogoobespecheniia/ruchnoe-testirovanie/testirovanie-chernogo-iashchika-2> (дата обращения 20.06.2024)

15. Достоинства и недостатки современных видов тестирования программного обеспечения ящика [Электронный ресурс]. URL: <https://moluch.ru/archive/125/34699/?ysclid=lx0s782can64330216> (дата обращения 20.06.2024)

16. Значения метрик кода наследования [Электронный ресурс]. URL: <https://learn.microsoft.com/ru-ru/visualstudio/code-quality/code-metrics-values?view=vs-2022> (дата обращения 25.06.2024)

17. Метрики кода глубина наследования [Электронный ресурс]. URL: <https://learn.microsoft.com/ru-ru/visualstudio/code-quality/code-metrics-depth-of-inheritance?view=vs-2022> (дата обращения 25.06.2024)

Приложение А
(справочное)
Глоссарий

Университет – учебное заведение, обучающее и готовящее специалистов разных профессий по естественным, техническим, медицинским, гуманитарным и другим видам наук.

Университетский кампус – территориальное объединение строений разного типа, относящихся к образовательной организации.

Факультет – подразделение университета, осуществляющую обучение и подготовку студентов по схожим специальностям.

Кафедра – элемент факультета, объединяющий группы студентов по определенной специализации.

Специальность – профессия, которую получают студенты после окончания своего обучения в университете.

Кабинет – помещение, оборудованное под определенный вид деятельности.

Учебный корпус – это главное здание университета, где расположен ректорат и проректорат, административные офисы, службы университета, а также учебные кабинеты.

Студент – человек, поступивший в высшее учебное заведение.

Приложение – программа, написанная на определенном языке программирования, с целью выполнения заложенной в нее автоматизации.

База данных (БД) – система, которую используют для хранения данных приложения. Данные структурированы и связаны между собой отношениями.

Язык программирования – инструмент, который используют для связи программиста и компьютера между собой.

Интерфейс – результат выполнения программы, с помощью которой пользователь взаимодействует с системой.

Приложение Б
(рекомендуемое)
Рисунки ВКР

Центральный кампус

Учебные корпуса

Корпус 1

Корпус 2

Корпус 3

Корпус 5

Общежития

Гостиница

Медицинский центр

Военный учебный центр

Издательский дом «Дело»

Автошкола

Круглосуточные диспетчерские

РАНХиГС
РОССИЙСКАЯ АКАДЕМИЯ НАРОДНОГО ХОЗЯЙСТВА
И ГОСУДАРСТВЕННОЙ СЛУЖБЫ
ПРИ ПРЕЗИДЕНТЕ РОССИЙСКОЙ ФЕДЕРАЦИИ

1

1

КОРПУС

ЭТАЖ

План 1-го этажа

Рисунок Б.1– Отображение подробной информации о корпусе

Гостиница

Медицинский центр

Военный учебный центр

Издательский дом «Дело»

Автошкола

Круглосуточные диспетчерские

РАНХиГС
РОССИЙСКАЯ АКАДЕМИЯ НАРОДНОГО ХОЗЯЙСТВА
И ГОСУДАРСТВЕННОЙ СЛУЖБЫ
ПРИ ПРЕЗИДЕНТЕ РОССИЙСКОЙ ФЕДЕРАЦИИ

3

1

КОРПУС

ЭТАЖ

Узнайте больше на нашем сайте
www.gaterra.ru

Дополнительные вопросы по адресу
info@gaterra.ru

РАНХиГС
РОССИЙСКАЯ АКАДЕМИЯ НАРОДНОГО ХОЗЯЙСТВА
И ГОСУДАРСТВЕННОЙ СЛУЖБЫ
ПРИ ПРЕЗИДЕНТЕ РОССИЙСКОЙ ФЕДЕРАЦИИ

3

2

КОРПУС

ЭТАЖ

Рисунок Б.2 – Отображение подробной информации о корпусе

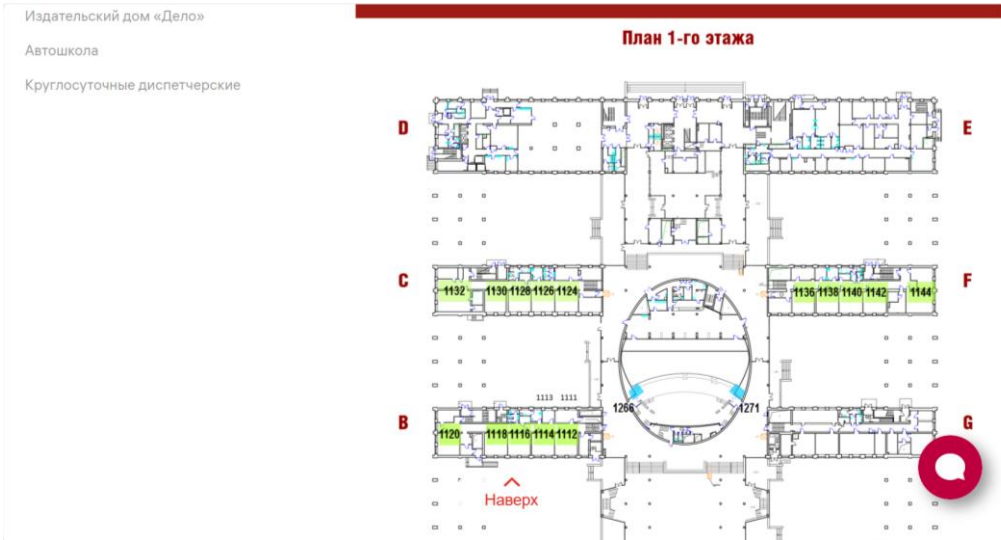


Рисунок Б.3 – Отображение кабинетов в плане этажа

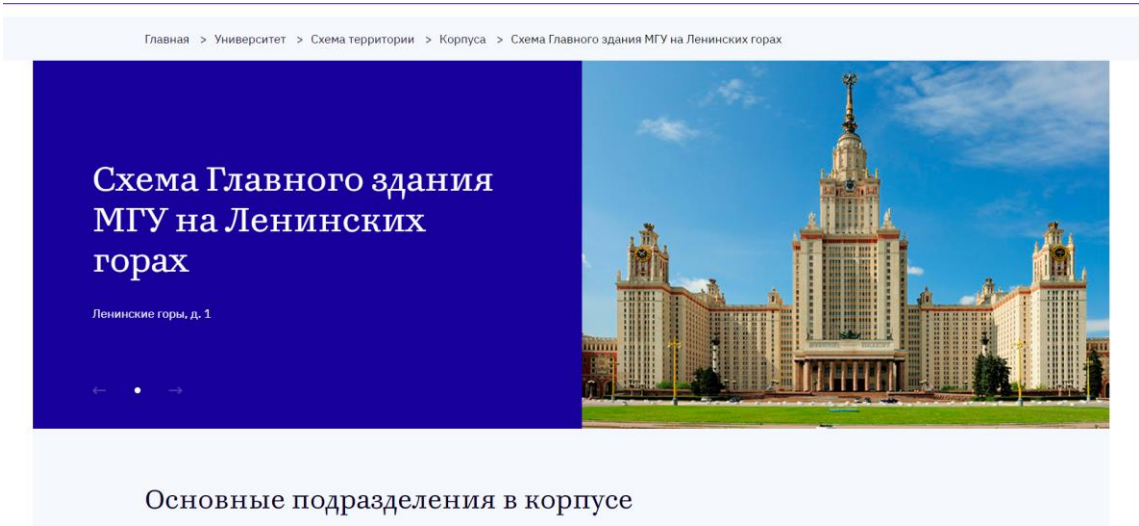


Рисунок Б.4 – Переход к подробной информации о выбранном корпусе

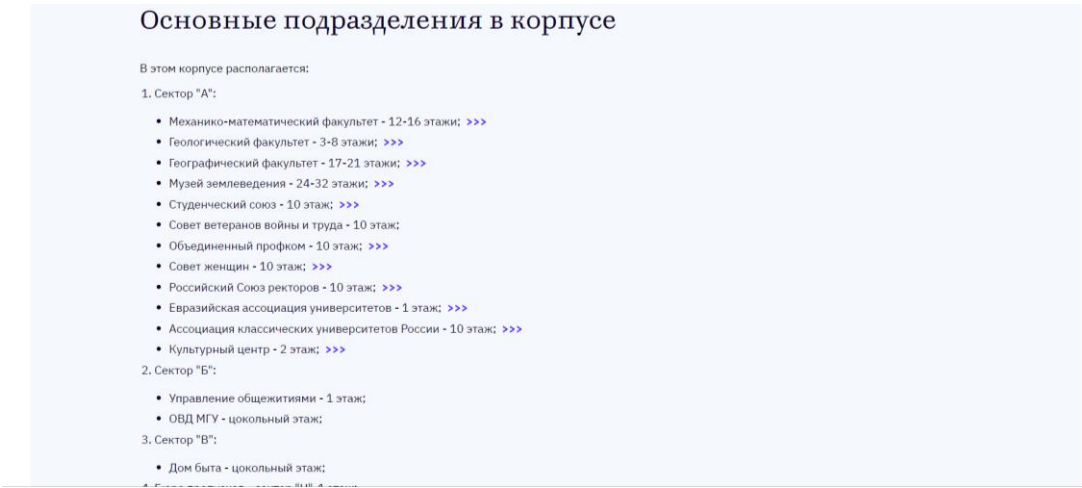
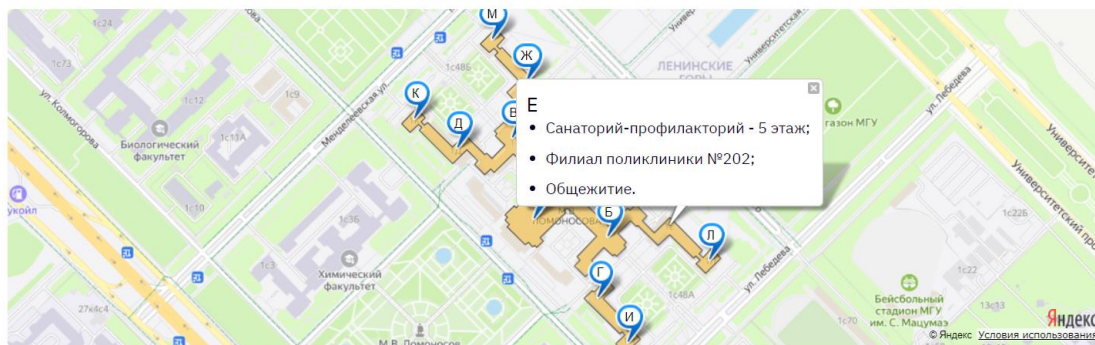


Рисунок Б.5 – Отображение списка подразделений в корпусе

Как добраться



Станция метро «Университет», далее авт. 1, 111, 113, 119, 464, м19 до остановки «Дом Культуры МГУ» или пешком. Станция метро «Ломоносовский проспект», далее авт. 1, 57, 113, 464, 715, м19 до остановки «Дом Культуры МГУ» или пешком.

Рисунок Б.6 – Расположение корпусов относительно других объектов

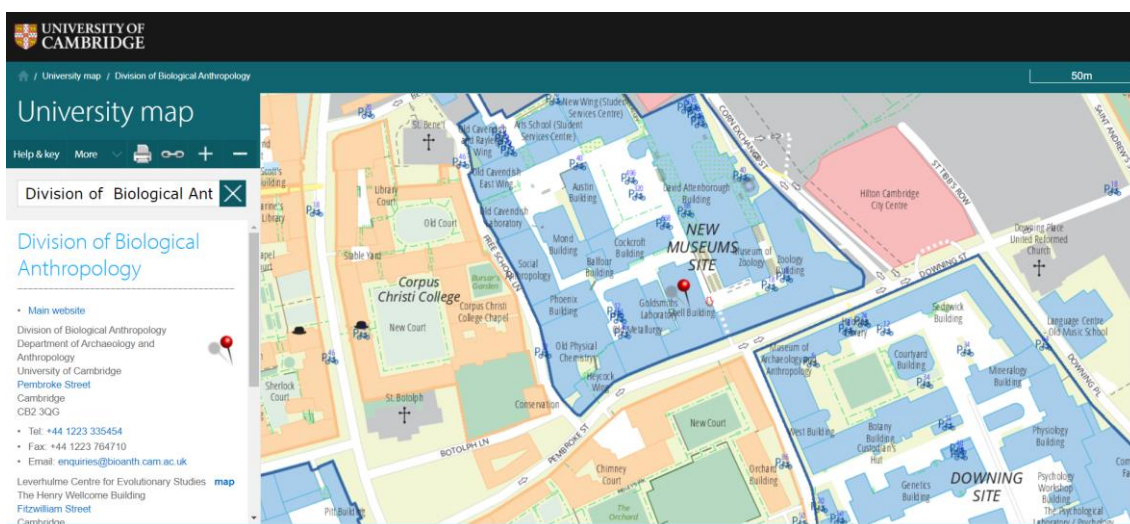


Рисунок Б.7 – Работа и отображение результатов поиска

Stanford University

Stanford | Login

SUNet ID

Password

Login

Important Security Information: Logging in lets you access other protected Stanford websites with this browser, not just the website you requested.

LOGIN HELP

TWO-STEP DEVICE UNAVAILABLE?

Use of this system is subject to Stanford University's rules and regulations. See the [Stanford Administrative Guide](#) for more information.

Stanford University

SU Home

Maps & Directions

Search Stanford

Terms of Use

Emergency Info

© Stanford University

Stanford, California 94305

Copyright Complaints

Trademark Notice

Рисунок Б.8 – Страница авторизации пользователя

Приложение В
(рекомендуемое)
Сценарии процессов

Таблица В.1 – Сценарий процесса «Поиска кабинета по наименованию и по фильтрам»

Поиск кабинета по наименованию и по фильтрам	
Id: 1	
Описание:	Система принимает запрос и загружает данные по выбранному пользователем объекту
Действующие лица:	Незарегистрированный пользователь, авторизованный пользователь
Предусловие:	Запрос о поиске кабинета
<p>Для незарегистрированного пользователя:</p> <ol style="list-style-type: none"> 1. Незарегистрированный пользователь вводит в строку поиска наименование кабинета. 2. Система формирует список подходящих кабинетов. 3. Незарегистрированный пользователь выбирает подходящий кабинет и нажимает кнопку Подробнее. 4. Система отображает страницу с подробными данными выбранного кабинета. <p>Для авторизованного пользователя:</p> <ol style="list-style-type: none"> 1. Авторизованный пользователь выбирает фильтр из списка. 2. Авторизованный пользователь вводит в строку поиска текст в соответствии с выбранным типом фильтра. 3. Система формирует список подходящих кабинетов. 4. Авторизованный пользователь выбирает подходящий кабинет и нажимает кнопку Подробнее. <p>Система отображает страницу с подробными данными выбранного кабинета.</p>	
Постусловие:	данные отображены системой для просмотра

Таблица В.2 – Сценарий процесса «Просмотр информации о корпусе»

Просмотр информации о корпусе	
Id: 2	
Описание:	Система принимает запрос и загружает данные по выбранному пользователем объекту
Действующие лица:	Незарегистрированный пользователь, авторизованный пользователь
Предусловие:	Запрос о просмотре информации о корпусе
<p>Для незарегистрированного пользователя:</p> <ol style="list-style-type: none"> 5. Незарегистрированный пользователь выбирает корпус из списка и нажимает кнопку Подробнее. 6. Система отображает страницу с подробными данными выбранного корпуса. 7. Незарегистрированный пользователь просматривает информацию о кафедрах, расположенных в выбранном корпусе. <p>Для авторизованного пользователя:</p> <ol style="list-style-type: none"> 1. Авторизованный пользователь выбирает корпус из списка и нажимает кнопку Подробнее. 2. Система отображает страницу с подробными данными выбранного корпуса. 3. Авторизованный пользователь просматривает информацию о кафедрах, расположенных в выбранном корпусе. 8. Авторизованный пользователь просматривает планы эвакуации каждого этажа выбранного корпуса 	
Постусловие:	данные отображены системой для просмотра

Таблица В.3 – Сценарий процесса «Регистрация пользователя»

Регистрация пользователя	
Id: 3	
Описание:	Система принимает и загружает данные, полученные от пользователя системой
Действующие лица:	Незарегистрированный пользователь
Предусловие:	Запрос о добавлении пользователя в систему
	<p>Для незарегистрированного пользователя:</p> <ol style="list-style-type: none"> 1. Пользователь выбирает кнопку Регистрации. 2. Система отображает страницу с регистрацией пользователя. 3. Пользователь вводит логи. 4. Пользователь вводит пароль. 5. Пользователь вводит фамилию. 6. Пользователь вводит имя. 7. Пользователь вводит отчество. 8. Пользователь вводит дату рождения. 9. Пользователь вводит пол. 10. Нажимает кнопку Зарегистрироваться. <p>Система добавляет данные о пользователе.</p>
Постусловие:	данные добавлены системой

Приложение Г
(рекомендуемое)
Атрибуты сущностей базы данных

Таблица Г.1 – Описание связей между сущностями концептуальной модели

№ п/п	Тип сущности 1	Тип сущности 2	Описание	Мощность	Обязательность	Ограничения
1	Корпус	Фото корпуса	Корпус <i>содержит</i> некоторое количество фотографий/ Фото корпуса <i>принадлежит</i> некоторому корпусу	1 : M	Обязательная (корпус обязательно должен содержать хотя бы одно фото)	Нет
2	Корпус	Кафедра	Корпус <i>содержит</i> кафедру/ Кафедра <i>принадлежит</i> некоторому корпусу	1 : M	Обязательная (кафедра прикрепляется в какому-то корпусу)	Нет
3	Кабинет	Корпус	Кабинет <i>относиться</i> к корпусу/ Корпус <i>имеет с своим составе</i> кабинет	1 : 1	Обязательная (кабинет без корпуса не может существовать)	Нет
4	Кабинет	Фото кабинета	Кабинет <i>содержит</i> некоторое количество фотографий / Фото кабинета <i>принадлежит</i> некоторому кабинету	1 : M	Обязательная (кабинет обязательно должен содержать хотя бы одно фото)	Нет
5	Кабинет	Оборудование	Кабинет <i>содержит</i> оборудование/ оборудование <i>принадлежит</i> кабинету	1 : 1	Обязательная (кабинет не может быть без оборудования)	Нет
6	Кафедра	Кабинет	Кафедра <i>относиться</i> к кабинету/ Кабинет <i>имеет</i> кафедру	1 : 1	Обязательная (у кафедры не может не быть кабинета)	Нет
7	Кафедра	Факультет	Кафедра <i>относиться</i> к факультету/ Факультет <i>имеет</i> кафедру	1 : 1	Обязательная (у кафедры не может не быть факультета)	Нет
8	Кафедра	Пользователь	Кафедра <i>содержит</i> пользователей/ Пользователь <i>принадлежит</i> кафедре	1 : M	Необязательная (у кафедры может не быть пользователей)	Нет
9	Тип кабинета	Кабинет	Тип кабинета <i>относиться</i> к кабинету/ Кабинет <i>имеет</i> тип кабинета	1 : M	Обязательная (у кабинета не может не быть типа кабинета)	Нет

Таблица Г.2 – Описание атрибутов сущности «Пользователь» концептуальной модели

№ п/п	Наименование	Описание	Тип данных	Размерность	Область допустимых значений	Null-значения	Роль	Пример
1	Код_пользователя	Код пользователя	целый	7	0000001-9999999	нет	РК	0000447
2	Логин	Логин пользователя при входе	символьный	50	'A-z', 0-9	нет		Kafka
3	Пароль	Пароль пользователя при входе	символьный	50	'A-z', 0-9	нет		kafka
4	Имя	Имя пользователя	символьный	250	'A-я'	нет		Павел
5	Фамилия	Фамилия пользователя	символьный	250	'A-я'	нет		Парамонов
6	Отчество	Отчество пользователя	символьный	250	'A-я'	нет		Семенович
7	Пол	Пол пользователя	символьный	1	м, ж	нет		м
8	Фото	Фото пользователя	символьный	100	'A-z', 01-99, _ ,	нет		Person1.png
9	Дата рождения	Дата рождения	Дата	10	01.01.1950-31.12.2050	нет		14.02.2002

Таблица Г.3 – Описание атрибутов сущности «Кабинет» концептуальной модели

№ п/п	Наименование	Описание	Тип данных	Размерность	Область допустимых значений	Null-значения	Роль	Пример
1	Код_кабинета	Код кабинета	целый	7	0000001-9999999	нет	РК	321293
2	Номер	Наименование кабинета	символьный	8	'A-я', 001-100, -	нет		7a-302
3	Этаж	Этаж, на котором расположен кабинет	целый	2	01-99	нет		3

Таблица Г.4 – Описание атрибутов сущности «Корпус» концептуальной модели

№ п/п	Наименование	Описание	Тип данных	Размерность	Область допустимых значений	Null-значения	Роль	Пример
1	Код_корпуса	Код корпуса	целый	7	0000001-9999999	нет	РК	321293
2	Наименование	Название корпуса	символьный	100	‘А-я’, 001-100, №	нет		Учебный корпус № 1
3	Номер	Номер корпуса	целый	2	01-99	нет		7
4	Информация	Дополнительная информация о корпусе	символьный	250	‘А-я’	да		Преимущественно учатся программисты
5	Адрес	Адрес расположения здания	символьный	250	‘А-я’	нет		ул.Красная 5

Таблица Г.5 – Описание атрибутов сущности «Оборудование» концептуальной модели

№ п/п	Наименование	Описание	Тип данных	Размерность	Область допустимых значений	Null-значения	Роль	Пример
1	Код_оборудования	Код оборудования	целый	3	001-999	нет	РК	002
2	Наименование	Название оборудования	символьный	100	‘А-я’	нет		проектор

Таблица Г.6 – Описание атрибутов сущности «Фото кабинета» концептуальной модели

№ п/п	Наименование	Описание	Тип данных	Размерность	Область допустимых значений	Null-значения	Роль	Пример
1	Код_фото_кабинета	Код фото кабинета	целый	7	0000001-9999999	нет	РК	321293
2	Номер_фото	Номер фото	целый	2	01-99	нет		1
3	Наименование	Название фото кабинета	символьный	100	‘А-z’, 01-99, _ , .	нет		Photo_2_cabinet.png

Таблица Г.7 – Описание атрибутов сущности «Тип кабинета» концептуальной модели

№ п/п	Наименование	Описание	Тип данных	Размерность	Область допустимых значений	Null-значения	Роль	Пример
1	Код_типа кабинета	Код типа кабинета	целый	5	00001- 99999	нет	РК	00007
2	Наименование	Название типа	символьный	250	‘А-я’	нет		лекционный

Таблица Г.8 – Описание атрибутов сущности «Фото корпуса» концептуальной модели

№ п/п	Наименование	Описание	Тип данных	Размерность	Область допустимых значений	Null-значения	Роль	Пример
1	Код_фото корпуса	Код фото корпуса	целый	7	0000001-9999999	нет	РК	321293
2	Номер_фото	Номер фото	целый	2	01-99	нет		1
3	Наименование	Название фото корпуса	символьный	100	‘А-з’, 01-99, _ , .	нет		Photo_2_corpuls.png
4	Вместимость	Количество человек, которое вмещается в аудиторию	целый	3	001-999	нет		100
5	Информация	Дополнительная информация о кабинете	символьный	250	‘А-я’	да		Кабинет относится к деканату

Таблица Г.9 – Описание атрибутов сущности «Факультет» концептуальной модели

№ п/п	Наименование	Описание	Тип данных	Размерность	Область допустимых значений	Null-значения	Роль	Пример
1	Код_факультета	Код факультета	целый	5	00001- 99999	нет	РК	00007
2	Наименование	Название факультета	символьный	250	‘А-я’	нет		Факультет вычислительной техники

Таблица Г.10 – Описание атрибутов сущности «Кафедра» концептуальной модели

№ п/п	Наименование	Описание	Тип данных	Размерность	Область допустимых значений	Null-значения	Роль	Пример
1	Код_кафедры	Код кафедры	целый	5	00001- 99999	нет	РК	00007
2	Наименование	Название кафедры	символьный	250	‘А-я’	нет		Кафедра Лечебное дело
3	Телефон	Номер телефона	символьный	11	0-9	нет		89623432222
4	Заф_кафедры	Заведующий кафедры	символьный	250	‘А-я’	нет		Антонов Павел Семенович

Приложение Д
(рекомендуемое)
Шаблоны пользовательского интерфейса

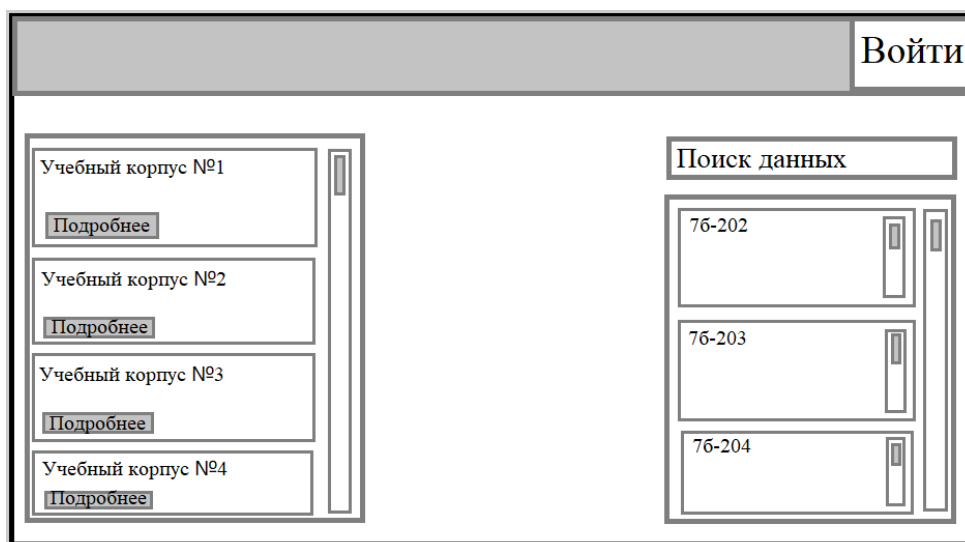


Рисунок Д.1 – Главное окно при запуске приложения

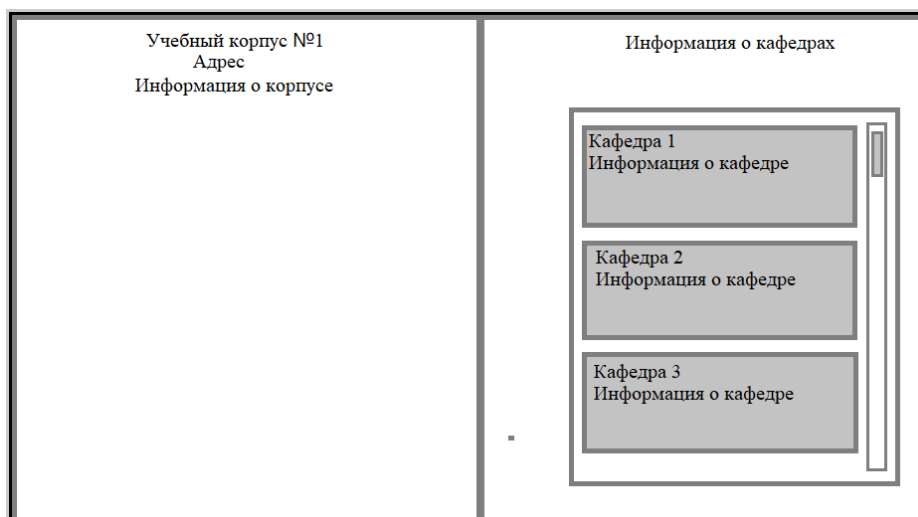


Рисунок Д.2 – Окно просмотра информации о корпусах для незарегистрированного пользователя

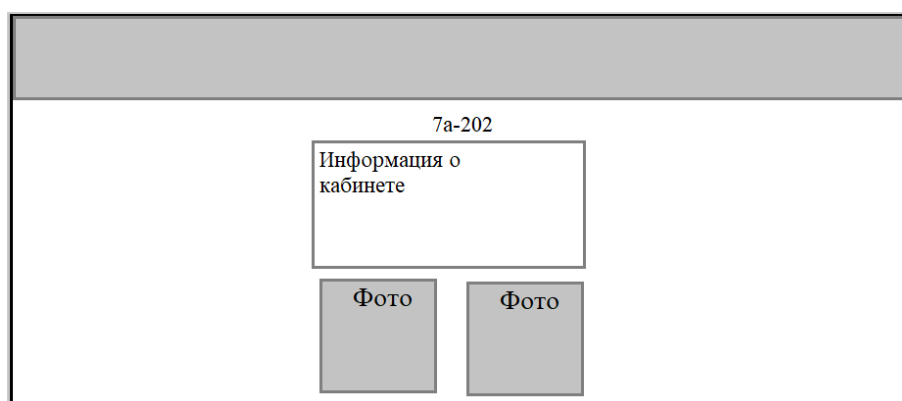


Рисунок Д.3 – Окно просмотра информации о кабинете

Регистрация

Логин

Пароль

Войти

Рисунок Д.4 – Окно входа в приложение

Регистрация пользователя

Фото

Данные

Данные

Данные

Данные

Сохранить

Рисунок Д.5 – Окно регистрации пользователя

Личный кабинет

Выход

Учебный корпус №1

Подробнее

Учебный корпус №2

Подробнее

Учебный корпус №3

Подробнее

Учебный корпус №4

Подробнее

Поиск данных с фильтром

76-202

76-203

76-204

Рисунок Д.6 – Окно поиска с фильтрами авторизованного пользователя

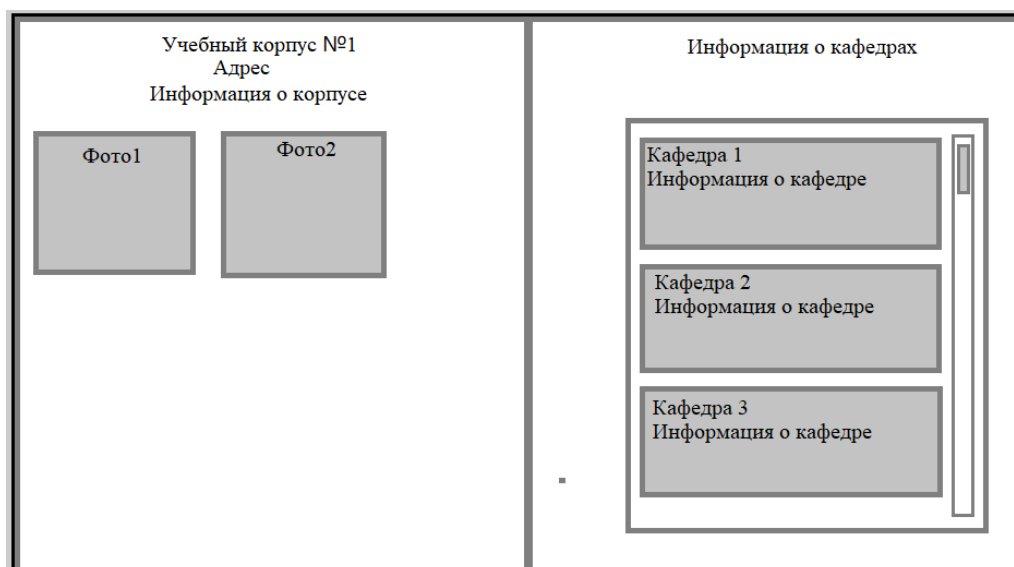


Рисунок Д.7 – Окно просмотра расширенной информации о корпусах

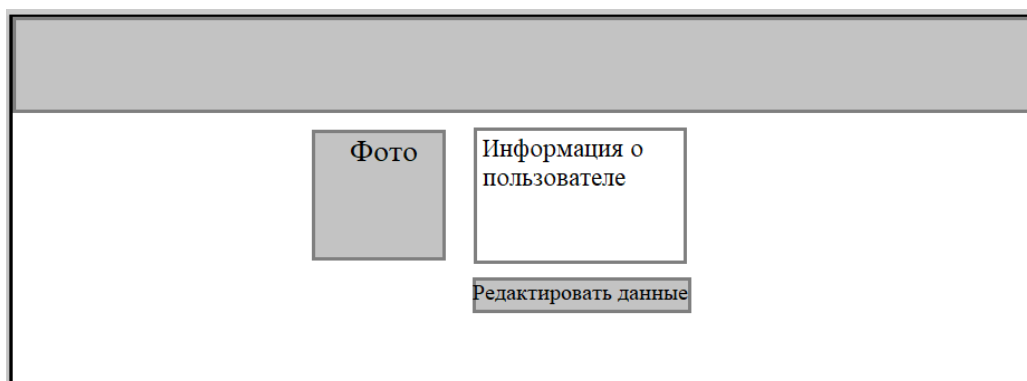


Рисунок Д.8 – Окно просмотра личного кабинета пользователя

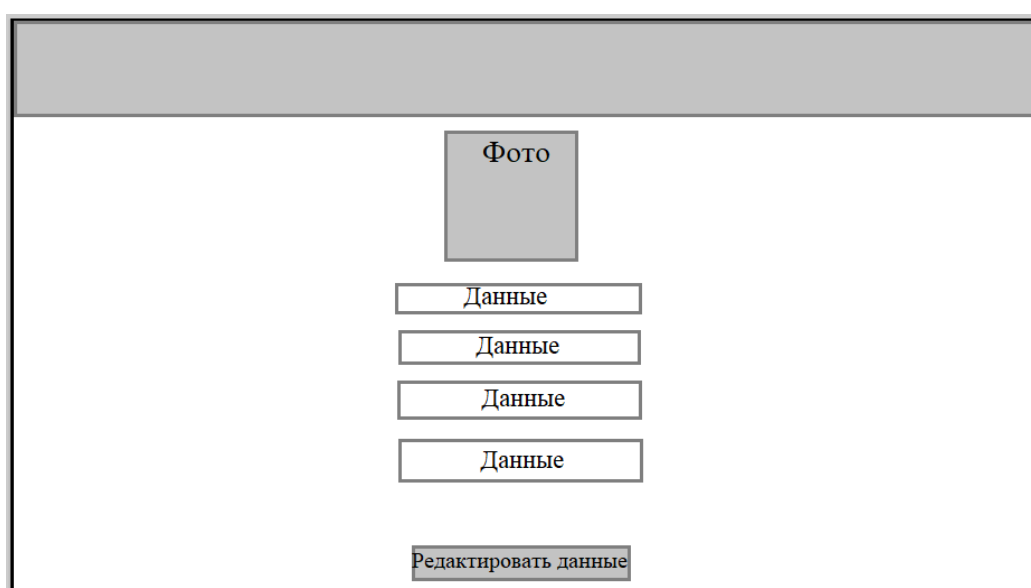


Рисунок Д.9 – Окно редактирования данных

Приложение Е
(рекомендуемое)
Код разработки базы данных

Листинг Е.1 – Скрипт для создания объектов базы данных

```

CREATE TABLE IF NOT EXISTS Faculty
(
    Id int primary key generated by default as identity,
    name varchar
);
CREATE TABLE IF NOT EXISTS Course
(
    id int primary key generated by default as identity,
    name varchar
);
CREATE TABLE IF NOT EXISTS Corpus
(
    id int primary key generated by default as identity,
    name varchar,
    number int,
    information varchar,
    address varchar
);
CREATE TABLE IF NOT EXISTS Equipment
(
    id int primary key generated by default as identity,
    name varchar
);
CREATE TABLE IF NOT EXISTS Cabinet
(
    id int primary key generated by default as identity,
    number varchar(50) not null,
    floor int not null,
    capacity int not null,
    information varchar,
    type varchar,
    corpus_id int references Corpus (id) on delete set null,
    equipment_id int references Equipment (id) on delete set null
);
CREATE TABLE IF NOT EXISTS Department
(
    id int primary key generated by default as identity,
    name varchar not null,
    phone varchar(11) not null,
    head_dep varchar,
    faculty_id int references Faculty (id) on delete set null,
    corpus_id int references Corpus (id) on delete set null,
    cabinet_id int references Cabinet (id) on delete set null
);
CREATE TABLE IF NOT EXISTS Person
(
    id int primary key generated by default as identity,
    username varchar(25) not null,
    password varchar not null,
    role varchar(25) not null,
    surname varchar(25) not null,
    firstname varchar(25) not null,
    gender int not null,
    photo varchar,
    date_of_birth date
);

```

```

CREATE TABLE IF NOT EXISTS Cabinet_photo
(
    id int primary key generated by default as identity,
    path varchar not null,
    name varchar not null,
    cabinet_id int references Cabinet (id) on delete set null
);
CREATE TABLE IF NOT EXISTS Corpus_photo
(
    id int primary key generated by default as identity,
    path varchar not null,
    name varchar not null,
    corpus_id int references Corpus (id) on delete set null
);

```

Листинг Е..2 – Скрипт заполнения таблицы «Факультет»

```

insert into Faculty (name) values ('Факультет промышленных
    технологий, электроэнергетики и транспорта');
insert into Faculty (name) values ('Факультет информационных
    технологий и электроники');
insert into Faculty (name) values ('Факультет вычислительной техники');
insert into Faculty (name) values ('Историко-фиологический факультет');
insert into Faculty (name) values ('Факультет физико-
    математических и естественных наук');
insert into Faculty (name) values ('Факультет педагогики,
    психологии и социальных наук');
insert into Faculty (name) values ('Лечебный факультет');
insert into Faculty (name) values ('Факультет стоматологии');
insert into Faculty (name) values ('Факультет физической культуры
    и спорта');
insert into Faculty (name) values ('Факультет международного
    сотрудничества');
insert into Faculty (name) values ('Факультет экономики и управления');
insert into Faculty (name) values ('Юридический факультет');

```

Листинг Е..3 – Скрипт заполнения таблицы «Курс»

```

insert into Course (name) values ('Первый курс. Бакалавриат');
insert into Course (name) values ('Второй курс. Бакалавриат');
insert into Course (name) values ('Третий курс. Бакалавриат');
insert into Course (name) values ('Четвертый курс. Бакалавриат');
insert into Course (name) values ('Пятый курс. Бакалавриат');
insert into Course (name) values ('Первый курс. Специалитет');
insert into Course (name) values ('Второй курс. Специалитет');
insert into Course (name) values ('Третий курс. Специалитет');
insert into Course (name) values ('Четвертый курс. Специалитет');
insert into Course (name) values ('Пятый курс. Специалитет');
insert into Course (name) values ('Первый курс. Магистратура');
insert into Course (name) values ('Второй курс. Магистратура');
insert into Course (name) values ('Первый курс. Аспирантура');
insert into Course (name) values ('Второй курс. Аспирантура');
insert into Course (name) values ('Третий курс. Аспирантура');

```

Листинг Е..4 – Скрипт заполнения таблицы «Корпус»

```

insert into Corpus (name, number, information, address) values
    ('Учебный корпус №1', 1, 'Главное здание ПГУ, где находится

```

```

ректорат и другие службы', 'ул.Красная 40');
insert into Corpus (name, number, information, address) values
('Учебный корпус №2', 2, 'В данном корпусе находится институт
военного обучения', 'ул.Чкалова 57');
insert into Corpus (name, number, information, address) values
('Учебный корпус №3', 3, 'В корпусе расположена обширная
библиотека. В ней представлены книги по разным
дисциплинам', 'ул.Красная 46');
insert into Corpus (name, number, information, address) values
('Учебный корпус №4', 4, 'Расположен корпус недалеко от
лыжной базы и столовой ПГУ', 'ул.Красная 46');
insert into Corpus (name, number, information, address) values
('Учебный корпус №5', 5, 'Преимущественно корпус юридического
направления, в этот корпус можно попасть по подземному
переходу из первого корпуса', 'ул.Красная 40');
insert into Corpus (name, number, information, address) values
('Учебный корпус №6', 6, 'Рядом с корпусом находится
спортивная площадка', 'ул.Красная 40');
insert into Corpus (name, number, information, address) values
('Учебный корпус №7', 7, 'В корпусе расположено студенческое
пространство. Это неформальная площадка для студентов,
созданная в целях проведения внеучебных мероприятий
и отдыха.', 'ул.Лермонтова 26');
insert into Corpus (name, number, information, address) values
('Учебный корпус №8', 8, 'Корпус соединен с первым и третьем
корпусами', 'ул.Красная 46');
insert into Corpus (name, number, information, address) values
('Учебный корпус №9', 9, 'Преимущественно экономическое
направление', 'ул.Чкалова 68');
insert into Corpus (name, number, information, address) values
('Учебный корпус №10', 10, 'Текст описания', 'ул.Лермонтова 3');
insert into Corpus (name, number, information, address) values
('Учебный корпус №11', 11, 'Текст описания', 'ул.Лермонтова 37');
insert into Corpus (name, number, information, address) values
('Учебный корпус №12', 12, 'Текст описания', 'ул.Лермонтова 37');
insert into Corpus (name, number, information, address) values
('Учебный корпус №13', 13, 'Текст описания', 'ул.Лермонтова 37');
insert into Corpus (name, number, information, address) values
('Учебный корпус №14', 14, 'Текст описания', 'ул.Лермонтова 37');
insert into Corpus (name, number, information, address) values
('Учебный корпус №15', 15, 'Текст описания', 'ул.Лермонтова 37');
insert into Corpus (name, number, information, address) values
('Учебный корпус №16', 16, 'Текст описания', 'ул.Карла Маркса 4');
insert into Corpus (name, number, information, address) values
('Учебный корпус №17', 17, 'Текст описания', 'ул.Чкалова 56');
insert into Corpus (name, number, information, address) values
('Учебный корпус №18', 18, 'Текст описания', 'ул.Лермонтова 37');
insert into Corpus (name, number, information, address) values
('Учебный корпус №19', 19, 'Текст описания', 'ул.Лермонтова 37');

```

Листинг Е..5 – Скрипт заполнения таблицы «Оборудование»

```

insert into Equipment (name) values ('Проектор');
insert into Equipment (name) values ('Компьютеры');
insert into Equipment (name) values ('Доска маркерная');
insert into Equipment (name) values ('Доска меловая');
insert into Equipment (name) values ('Лаборатория');

```

Листинг Е..6 – Скрипт заполнения таблицы «Кабинет»

```
INSERT INTO Cabinet (number, floor, capacity, type, information,
                    corpus_id, equipment_id)
VALUES
    (TRIM('8-302a'), 3, 6, TRIM('Деканат'), TRIM('Деканат
        ФПТЭиТ'), 8, 2),
    (TRIM('7a-510'), 5, 7, TRIM('Деканат'), TRIM('Деканат КТ'), 7, 2),
    (TRIM('3-207'), 3, 6, TRIM('Деканат'), TRIM('Деканат
        ФСЛПИМ'), 3, 2),
    (TRIM('7a-313'), 3, 5, TRIM('Деканат'), TRIM('Деканат ВТ'), 7, 2),
    (TRIM('7a-323'), 3, 6, TRIM('Деканат'), TRIM('Деканат
        МОиПЭВМ'), 7, 2),
    (TRIM('10-306'), 3, 5, TRIM('Деканат'), TRIM('Деканат
        хирургии'), 10, 2),
    (TRIM('10-119'), 1, 6, TRIM('Деканат'), TRIM('Деканат
        анатомии человека'), 10, 2),
    (TRIM('8-207'), 2, 10, TRIM('Приемная комиссия'),
    (TRIM('Приемная комиссия'), 8, 2),
    (TRIM('7б-201'), 2, 34, TRIM('Лекционный'), NULL, 7, 4),
    (TRIM('7б-204'), 2, 100, TRIM('Лекционный'), NULL, 7, 1),
    (TRIM('7a-324'), 3, 24, TRIM('Практический'), NULL, 7, 2),
    (TRIM('7a-309'), 3, 18, TRIM('Практический'), NULL, 7, 2);
```

Листинг Е..7 – Скрипт заполнения таблицы «Кафедра»

```
insert into Department (name, phone, head_dep, faculty_id,
                        corpus_id, cabinet_id) values ('Сварочное, литейное
                            производство и материаловедение', '88412666262', 'Розен
                            Андрей Евгеньевич', 1, 3, 2);
insert into Department (name, phone, head_dep, faculty_id,
                        corpus_id, cabinet_id) values ('Математическое обеспечение и
                            применение ЭВМ', '88412666592', 'Козлов Андрей Юрьевич', 1, 7, 5);
insert into Department (name, phone, head_dep, faculty_id,
                        corpus_id, cabinet_id) values ('Техносферная безопасность',
                            '88412208402', 'Везборода Оксана Евгеньевна', 1, 8, 1);
insert into Department (name, phone, head_dep, faculty_id,
                        corpus_id, cabinet_id) values ('Компьютерные технологии',
                            '88412666569', 'Горбаченко Владимир Иванович', 3, 7, 3);
insert into Department (name, phone, head_dep, faculty_id,
                        corpus_id, cabinet_id) values ('Вычислительная техника',
                            '88412666589', 'Митрохин Максим Александрович', 3, 7, 4);
insert into Department (name, phone, head_dep, faculty_id,
                        corpus_id, cabinet_id) values ('Анатомия человека',
                            '88412643972', 'Калмин Олег Витальевич', 7, 10, 7);
insert into Department (name, phone, head_dep, faculty_id,
                        corpus_id, cabinet_id) values ('Хирургия', '88412643964',
                            'Моисеева Инесса Яковлевна', 7, 10, 6);
```

Листинг Е..8 – Скрипт заполнения таблицы «Кабинет фото»

```
INSERT INTO Cabinet_photo (path, name, cabinet_id)
VALUES ('/images/cabinet/cabinet_1.png', 'Фото кабинета 1', 1),
    ('/images/cabinet/cabinet_2.png', 'Фото кабинета 2', 2),
    ('/images/cabinet/cabinet_3.png', 'Фото кабинета 3', 3),
    ('/images/cabinet/cabinet_4.png', 'Фото кабинета 4', 4),
    ('/images/cabinet/cabinet_5.png', 'Фото кабинета 5', 5),
    ('/images/cabinet/cabinet_6.png', 'Фото кабинета 6', 6),
```

```
( '/images/cabinet/cabinet_7.png', 'Фото кабинета 7', 7),
( '/images/cabinet/cabinet_8.png', 'Фото кабинета 8', 8),
( '/images/cabinet/cabinet_9.png', 'Фото кабинета 9', 9),
( '/images/cabinet/cabinet_10.png', 'Фото кабинета 10', 10),
( '/images/cabinet/cabinet_11.png', 'Фото кабинета 11', 11),
( '/images/cabinet/cabinet_12.png', 'Фото кабинета 12', 12),
( '/images/cabinet/cabinet_1_evacuation.png', 'План эвакуации
  кабинета 1', 1),
( '/images/cabinet/cabinet_2_evacuation.png', 'План эвакуации
  кабинета 2', 2),
( '/images/cabinet/cabinet_3_evacuation.png', 'План эвакуации
  кабинета 3', 3),
( '/images/cabinet/cabinet_4_evacuation.png', 'План эвакуации
  кабинета 4', 4),
( '/images/cabinet/cabinet_5_evacuation.png', 'План эвакуации
  кабинета 5', 5),
( '/images/cabinet/cabinet_6_evacuation.png', 'План эвакуации
  кабинета 6', 6),
( '/images/cabinet/cabinet_7_evacuation.png', 'План эвакуации
  кабинета 7', 7),
( '/images/cabinet/cabinet_8_evacuation.png', 'План эвакуации
  кабинета 8', 8),
( '/images/cabinet/cabinet_9_evacuation.png', 'План эвакуации
  кабинета 9', 9),
( '/images/cabinet/cabinet_10_evacuation.png', 'План эвакуации
  кабинета 10', 10),
( '/images/cabinet/cabinet_11_evacuation.png', 'План эвакуации
  кабинета 11', 11),
( '/images/cabinet/cabinet_12_evacuation.png', 'План эвакуации
  кабинета 12', 12);
```

Листинг Е.9 – Скрипт заполнения таблицы «Корпус фото»

```
INSERT INTO Corpus_photo (path, name, corpus_id)
VALUES ( '/images/corpus/corpus_1.png', 'Фото корпуса 1', 1),
( '/images/corpus/corpus_2.png', 'Фото корпуса 2', 2),
( '/images/corpus/corpus_3.png', 'Фото корпуса 3', 3),
( '/images/corpus/corpus_4.png', 'Фото корпуса 4', 4),
( '/images/corpus/corpus_5.png', 'Фото корпуса 5', 5),
( '/images/corpus/corpus_6.png', 'Фото корпуса 6', 6),
( '/images/corpus/corpus_7.png', 'Фото корпуса 7', 7),
( '/images/corpus/corpus_8.png', 'Фото корпуса 8', 8),
( '/images/corpus/corpus_9.png', 'Фото корпуса 9', 9),
( '/images/corpus/corpus_10.png', 'Фото корпуса 10', 10),
( '/images/corpus/corpus_11.png', 'Фото корпуса 11', 11),
( '/images/corpus/corpus_12.png', 'Фото корпуса 12', 12),
( '/images/corpus/corpus_13.png', 'Фото корпуса 13', 13),
( '/images/corpus/corpus_14.png', 'Фото корпуса 14', 14),
( '/images/corpus/corpus_15.png', 'Фото корпуса 15', 15),
( '/images/corpus/corpus_16.png', 'Фото корпуса 16', 16),
( '/images/corpus/corpus_17.png', 'Фото корпуса 17', 17),
( '/images/corpus/corpus_18.png', 'Фото корпуса 18', 18),
( '/images/corpus/corpus_19.png', 'Фото корпуса 19', 19),
( '/images/corpus/corpus_1_evacuation.png', 'План эвакуации корпуса
  1', 1),
( '/images/corpus/corpus_2_evacuation.png', 'План эвакуации корпуса
  2', 2),
( '/images/corpus/corpus_3_evacuation.png', 'План эвакуации корпуса
```

```

        3', 3),
('/images/corpus/corpus_4_evacuation.png', 'План эвакуации корпуса
    4', 4),
('/images/corpus/corpus_5_evacuation.png', 'План эвакуации корпуса
    5', 5),
('/images/corpus/corpus_6_evacuation.png', 'План эвакуации корпуса
    6', 6),
('/images/corpus/corpus_7_evacuation.png', 'План эвакуации корпуса
    7 1 этаж', 7),
('/images/corpus/corpus_7_evacuation2.png', 'План эвакуации
    корпуса 7а 2 этаж', 7),
('/images/corpus/corpus_7_evacuation2b.png', 'План эвакуации
    корпуса 7б 2 этаж', 7),
('/images/corpus/corpus_7_evacuation3.png', 'План эвакуации
    корпуса 7 3 этаж', 7),
('/images/corpus/corpus_7_evacuation5.png', 'План эвакуации
    корпуса 7 5 этаж', 7),
('/images/corpus/corpus_8_evacuation.png', 'План эвакуации корпуса
    8', 8),
('/images/corpus/corpus_9_evacuation.png', 'План эвакуации корпуса
    9', 9),
('/images/corpus/corpus_10_evacuation.png', 'План эвакуации
    корпуса 10', 10),
('/images/corpus/corpus_11_evacuation.png', 'План эвакуации
    корпуса 11', 11),
('/images/corpus/corpus_12_evacuation.png', 'План эвакуации
    корпуса 12', 12),
('/images/corpus/corpus_13_evacuation.png', 'План эвакуации
    корпуса 13', 13),
('/images/corpus/corpus_14_evacuation.png', 'План эвакуации
    корпуса 14', 14),
('/images/corpus/corpus_15_evacuation.png', 'План эвакуации
    корпуса 15', 15),
('/images/corpus/corpus_16_evacuation.png', 'План эвакуации
    корпуса 16', 16),
('/images/corpus/corpus_17_evacuation.png', 'План эвакуации
    корпуса 17', 17),
('/images/corpus/corpus_18_evacuation.png', 'План эвакуации
    корпуса 18', 18),
('/images/corpus/corpus_19_evacuation.png', 'План эвакуации
    корпуса 12', 19);

```

Приложение Ж
(рекомендуемое)
Листинг программы

Листинг Ж.1 – Код MvcConfig.java

```
package ru.study.universitynavigator.configuration;
import org.springframework.context.annotation.Configuration;
import org.springframework.web.servlet.config.annotation.
    ResourceHandlerRegistry;
import org.springframework.web.servlet.config.annotation.
    WebMvcConfigurer;
@Configuration
public class MvcConfig implements WebMvcConfigurer {
    //Редиректим работу с директорией img внутри static каталога
    //данный каталог не может использоваться для runtime взаимодействий
    @Override
    public void addResourceHandlers(ResourceHandlerRegistry registry) {
        registry.addResourceHandler("/img/**")
            .addResourceLocations("file:img/");
    }
}
```

Листинг Ж.2 – Код SecurityConfig.java

```
package ru.study.universitynavigator.configuration;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Bean;
import org.springframework.security.config.annotation.
    authentication.builders.AuthenticationManagerBuilder;
import org.springframework.security.config.annotation.method.
    configuration.EnableGlobalMethodSecurity;
import org.springframework.security.config.annotation.web.
    builders.HttpSecurity;
import org.springframework.security.config.annotation.web.
    configuration.EnableWebSecurity;
import org.springframework.security.config.annotation.web.
    configuration.WebSecurityConfigurerAdapter;
import org.springframework.security.crypto.bcrypt.
    BCryptPasswordEncoder;
import org.springframework.security.crypto.password.
    PasswordEncoder;
import org.springframework.security.web.util.matcher.
    AntPathRequestMatcher;
import ru.study.universitynavigator.service.PersonDetailsService;
@EnableWebSecurity
@EnableGlobalMethodSecurity(prePostEnabled = true)
public class SecurityConfig extends WebSecurityConfigurerAdapter {
    private final PersonDetailsService personDetailsService;
    @Autowired
    public SecurityConfig(PersonDetailsService personDetailsService) {
        this.personDetailsService = personDetailsService;
    }
    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http
            .authorizeRequests()
                .antMatchers("/main",
                    "/search",
                    "/search/corpus/{id}",
                    "/search/cabinet/{id}",
```

```

        "/auth/registration",
        "/auth/login").permitAll()
    .antMatchers("/authorized/main",
        "/authorized/search",
        "/authorized/search/corpus/{id}",
        "/authorized/search/cabinet/{id}",
        "/authorized/edit-profile/{username}",
        "/authorized/update-
        person/{id}").hasAnyRole("USER")
    .and()
    .formLogin()
    .loginPage("/auth/login")
    .loginProcessingUrl("/process_login")
    .defaultSuccessUrl("/authorized/main", true)
    .and()
    .logout()
    .logoutRequestMatcher(new
    AntPathRequestMatcher("/logout", "GET"))//
        URL, который инициирует выход из сессии
    .logoutSuccessUrl("/main"); // URL, на который
    пользователь будет
        перенаправлен после выхода
    }
// Настраиваем аутентификацию
@Override
protected void configure(AuthenticationManagerBuilder auth) throws
Exception {
    auth.userDetailsService(personDetailsService)
        .passwordEncoder(getPasswordEncoder());
}
@Bean
public PasswordEncoder getPasswordEncoder() {
    return new BCryptPasswordEncoder();
}
}

```

Листинг Ж.3 – Код AuthController.java

```

package ru.study.universitynavigator.controller;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PostMapping;
import ru.study.universitynavigator.entity.Person;
import ru.study.universitynavigator.service.RegistrationService;
import ru.study.universitynavigator.util.PersonValidator;
import javax.validation.Valid;
@Controller
public class AuthController {
    private final PersonValidator personValidator;
    private final RegistrationService registrationService;
    @Autowired
    public AuthController(PersonValidator personValidator,
        RegistrationService
            registrationService) {

```

```

        this.personValidator = personValidator;
        this.registrationService = registrationService;
    }
    @GetMapping("/auth/login")
    public String loginPage() {
        return "login";
    }
    @GetMapping("/auth/registration")
    public String registrationPage(@ModelAttribute("person") Person
    person) {
        return "registration";
    }
    @PostMapping("/auth/registration")
    public String performRegistration(@ModelAttribute("person") @Valid
    Person person,
        BindingResult bindingResult, Model model) {
        //Проверяем существует ли пользователь, чтобы не перезаписать
        данные уже
        существующего
        personValidator.validate(person, bindingResult);
        //При возникновении ошибки открываем форму ошибки
        if (bindingResult.hasErrors()) {
            model.addAttribute("errors",
                bindingResult.getAllErrors());
            return "/error";
        }
        registrationService.register(person);
        return "login";
    }
}

```

Листинг Ж.4 – Код DataController.java

```

package ru.study.universitynavigator.controller;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.core.Authentication;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.multipart.MultipartFile;
import org.springframework.web.servlet.mvc.support
    RedirectAttributes;
import org.thymeleaf.util.StringUtils;
import ru.study.universitynavigator.entity.*;
import ru.study.universitynavigator.repository.PersonRepository;
import ru.study.universitynavigator.service.*;
import ru.study.universitynavigator.util.PhotoUploaderUtil;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.List;
@Controller
public class DataController {
    private final CorpusService corpusService;

```

```

private final CabinetService cabinetService;
private final DepartmentService departmentService;
private final CabinetPhotoService cabinetPhotoService;
private final CorpusPhotoService corpusPhotoService;
private final PersonDetailsService personDetailsService;
private final PhotoUploaderUtil photoUploaderUtil;
@Autowired
public DataController(CorpusService corpusService,
    CabinetService cabinetService, DepartmentService
        departmentService, CabinetPhotoService
            cabinetPhotoService, CorpusPhotoService
                corpusPhotoService, PersonRepository
                    personRepository,
                        PersonDetailsService
                            personDetailsService,
                                PhotoUploaderUtil
                                    photoUploaderUtil)
{
    this.corpusService = corpusService;
    this.cabinetService = cabinetService;
    this.departmentService = departmentService;
    this.cabinetPhotoService = cabinetPhotoService;
    this.corpusPhotoService = corpusPhotoService;
    this.personDetailsService = personDetailsService;
    this.photoUploaderUtil = photoUploaderUtil;
}

@GetMapping("/main")
public String loginPage(Model model) {
    List<Corpus> corpuses = corpusService.findAll();
    model.addAttribute("corpuses", corpuses);
    return "main";
}

@GetMapping("/search")
public String searchByNumber(@RequestParam("number") String
    number, RedirectAttributes redirectAttributes){
    List<Cabinet> cabinets =
        cabinetService.findByNumber(number.trim());
    redirectAttributes.addFlashAttribute("cabinets",
        cabinets);
    return "redirect:/main";
}

@GetMapping("/search/corpus/{id}")
public String searchCorpusInfo(@PathVariable String id, Model
    model){
    Corpus corpus = corpusService.findById
        (Integer.parseInt(id));
    List<CorpusPhoto> photos = corpusPhotoService.
        getPhotoByCorpusId(Integer.parseInt(id));
    List<Department> departments = departmentService.
        findAllByCorpusId(Integer.parseInt(id));
    model.addAttribute("corpus", corpus);
    model.addAttribute("photos", photos);
    model.addAttribute("departments", departments);
    return "corpus";
}

@GetMapping("/search/cabinet/{id}")
public String searchCabinetInfo(@PathVariable String id,

```

```

        Model model){
    Cabinet cabinet = cabinetService.findById
        (Integer.parseInt(id));
    List<CabinetPhoto> photos = cabinetPhotoService.
        getPhotoByCabinetId(Integer.parseInt(id));
    model.addAttribute("cabinet", cabinet);
    model.addAttribute("photos", photos);
    return "cabinet";
}
@GetMapping("/authorized/main")
public String authorizedMain(Authentication authentication,
    Model model){
    if (authentication != null &&
        authentication.isAuthenticated()) {
        String username = authentication.getName();//Получаем имя
        model.addAttribute("username", username);//Передаем в модель
    }
    List<Corpus> corpuses = corpusService.findAll();
    model.addAttribute("corpuses", corpuses);
    return "main-authorized";
}
@GetMapping("/authorized/search")
public String searchByNumber(@RequestParam("criteria") String
    criteria, @RequestParam("value") String value,
    RedirectAttributes redirectAttributes){
    List<Cabinet> cabinets = new ArrayList<>();
    switch (criteria) {
        case "type":
            cabinets = cabinetService.findByType
                (value.trim());
            break;
        case "capacity":
            cabinets = cabinetService.findByCapacity
                (Integer.parseInt(value.trim()));
            break;
        case "number":
            cabinets = cabinetService.findByNumber
                (value.trim());
            break;
        case "floor":
            cabinets = cabinetService.findByFloor
                (Integer.parseInt(value.trim()));
            break;
        case "equipment":
            cabinets = cabinetService.findByEquipment
                (value.trim());
            break;
        case "corpus":
            cabinets = cabinetService.findByCorpus
                (value.trim());
            break;
    }
    redirectAttributes.addFlashAttribute("cabinets",
        cabinets);
    return "redirect:/authorized/main";
}
@GetMapping("/authorized/search/corpus/{id}")
public String searchCorpusInfoAfterAuthorize(@PathVariable

```

```

String id, Model model){
    Corpus corpus = corpusService.findById
        (Integer.parseInt(id));
    List<CorpusPhoto> photos = corpusPhotoService.
        getPhotoByCorpusId(Integer.parseInt(id));
    List<Department> departments = departmentService.
        findAllByCorpusId(Integer.parseInt(id));
    model.addAttribute("corpus", corpus);
    model.addAttribute("photos", photos);
    model.addAttribute("departments", departments);
    return "corpus-authorized";
}
@GetMapping("/authorized/search/cabinet/{id}")
public String searchCabinetInfoAfterAuthorize(@PathVariable
    String id, Model model){
    Cabinet cabinet =
        cabinetService.findById(Integer.parseInt(id));
    List<CabinetPhoto> photos = cabinetPhotoService.
        getPhotoByCabinetId(Integer.parseInt(id));
    model.addAttribute("cabinet", cabinet);
    model.addAttribute("photos", photos);
    return "cabinet-authorized";
}
@GetMapping("/authorized/edit-profile/{username}")
public String editPage(@PathVariable("username") String
    username, Model model) {
    Person person = personDetailsService.findByUsername
        (username);
    model.addAttribute("person", person);
    return "edit-profile";
}
@PostMapping("/authorized/update-person/{id}")
public String edit(@PathVariable("id") int id,@RequestParam
    ("person") Person person,@RequestParam String surname,
    @RequestParam String firstname,@RequestParam String
        dateOfBirth,@RequestParam int gender,
    @RequestParam String username,
    @RequestParam("photo") MultipartFile
        multipartFile,RedirectAttributes
        redirectAttributes) {
    if(username != null){
        person.setUsername(username);
    }
    if (surname != null) {
        person.setSurname(surname);
    }
    if (firstname != null) {
        person.setFirstname(firstname);
    }
    if (gender != 0) {
        person.setGender(gender);
    }
    if(dateOfBirth != null){
        SimpleDateFormat dateFormat = new SimpleDateFormat
            ("yyyy-MM-dd");
        try {
            Date date = dateFormat.parse(dateOfBirth);
            person.setDateOfBirth(date);

```

```

        } catch (ParseException e) {
            System.out.println("Some error when trying to
                                parse String to Date " + e.getMessage());
        }
    }
    String resultFilename = photoUploaderUtil.uploadPhoto
        (multipartFile);
    if (!StringUtils.isEmpty(resultFilename)) {
        person.setPhoto(resultFilename);
    }
    personDetailsService.update(id, person);
    redirectAttributes.addFlashAttribute("username",
        username);
    return "redirect:/authorized/edit-profile/" + username;
}
}

```

Листинг Ж.5 – Код Cabinet.java

```

package ru.study.universitynavigator.entity;
import lombok.Getter;
import lombok.Setter;
import javax.persistence.*;
@Entity
@Getter
@Setter
@Table(name = "Cabinet")
public class Cabinet {
    //Уникальный идентификатор кабинета
    @Id
    @Column(name = "id")
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;
    //Номер кабинета
    @Column(name = "number")
    private String number;
    //Этаж кабинета
    @Column(name = "floor")
    private int floor;
    //Вместимость кабинета
    @Column(name = "capacity")
    private int capacity;
    //Общая информация
    @Column(name = "information")
    private String information;
    //Тип кабинета
    @Column(name = "type")
    private String type;
    //Уникальный идентификатор корпуса, в котором находится кабинет
    @OneToOne(fetch = FetchType.EAGER)
    @JoinColumn(name = "corpus_id")
    private Corpus corpus;
    //Уникальный идентификатор оборудования, которое есть в кабинете
    @OneToOne(fetch = FetchType.EAGER)
    @JoinColumn(name = "equipment_id")
    private Equipment equipment;
}

```

Листинг Ж.6 – Код CabinetPhoto.java

```
package ru.study.universitynavigator.entity;
import lombok.Getter;
import lombok.Setter;
import javax.persistence.*;
@Entity
@Getter
@Setter
@Table(name = "Cabinet_photo")
public class CabinetPhoto {
    //Уникальный идентификатор картинки
    @Id
    @Column(name = "id")
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;
    //Путь до картинки кабинета
    @Column(name = "path")
    private String path;
    //Название картинки
    @Column(name = "name")
    private String name;
    //Уникальный идентификатор кабинета
    @Column(name = "cabinet_id")
    private int cabinetId;
}
```

Листинг Ж.7 – Код Corpus.java

```
package ru.study.universitynavigator.entity;
import lombok.Getter;
import lombok.Setter;
import javax.persistence.*;
@Entity
@Getter
@Setter
@Table(name = "Corpus")
public class Corpus {
    //Уникальный идентификатор корпуса
    @Id
    @Column(name = "id")
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;
    //Наименование корпуса
    @Column(name = "name")
    private String name;
    //Номер корпуса
    @Column(name = "number")
    private int number;
    //Общая информация
    @Column(name = "information")
    private String information;
    //Адрес корпуса
    @Column(name = "address")
    private String address;
}
```

Листинг Ж.8 – Код CorpusPhoto.java

```

package ru.study.universitynavigator.entity;
import lombok.Getter;
import lombok.Setter;
import javax.persistence.*;
@Entity
@Getter
@Setter
@Table(name = "Corpus_photo")
public class CorpusPhoto {
    //Уникальный идентификатор картинки
    @Id
    @Column(name = "id")
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;
    //Путь до картинки корпуса
    @Column(name = "path")
    private String path;
    //Название картинки
    @Column(name = "name")
    private String name;
    //Уникальный идентификатор корпуса
    @Column(name = "corpus_id")
    private int corpusId;
}

```

Листинг Ж.9 – Код Course.java

```

package ru.study.universitynavigator.entity;
import lombok.Getter;
import lombok.Setter;
import javax.persistence.*;
@Entity
@Getter
@Setter
@Table(name = "Course")
public class Course {
    //Уникальный идентификатор курса
    @Id
    @Column(name = "id")
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;
    //Наименование курса
    @Column(name = "name")
    private String name;
}

```

Листинг Ж.10 – Код Department.java

```

package ru.study.universitynavigator.entity;
import lombok.Getter;
import lombok.Setter;
import javax.persistence.*;
@Entity
@Getter
@Setter
@Table(name = "Department")
public class Department {
    //Уникальный идентификатор кафедры

```

```

@Id
@Column(name = "id")
@GeneratedValue(strategy = GenerationType.IDENTITY)
private int id;
//Наименование кафедры
@Column(name = "name")
private String name;
//Номер телефона кафедры
@Column(name = "phone")
private String phone;
//ФИО руководителя кафедры
@Column(name = "head_dep")
private String headDep;
//Уникальный идентификатор факультета, которому принадлежит кафедра
@OneToOne(fetch = FetchType.EAGER)
@JoinColumn(name = "faculty_id")
private Faculty faculty;
//Уникальный идентификатор корпуса, в котором базируется кафедра
@Column(name = "corpus_id")
private int corpusId;
//Уникальный идентификатор деканата, относящегося к кафедре
@Column(name = "cabinet_id")
private int cabinetId;
}

```

Листинг Ж.11 – Код Equipment.java

```

package ru.study.universitynavigator.entity;
import lombok.Getter;
import lombok.Setter;
import lombok.ToString;
import javax.persistence.*;
@Entity
@Getter
@Setter
@Table(name = "Equipment")
public class Equipment {
    //Уникальный идентификатор оборудования
    @Id
    @Column(name = "id")
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;
    //Наименование оборудования
    @Column(name = "name")
    private String name;
}

```

Листинг Ж.12 – Код Faculty.java

```

package ru.study.universitynavigator.entity;
import lombok.Getter;
import lombok.Setter;
import javax.persistence.*;
@Entity
@Getter
@Setter
@Table(name = "Faculty")
public class Faculty {

```

```

        //Уникальный идентификатор факультета
        @Id
        @Column(name = "id")
        @GeneratedValue(strategy = GenerationType.IDENTITY)
        private int id;
        //Наименование факультета
        @Column(name = "name")
        private String name;
    }

```

Листинг Ж.13 – Код Person.java

```

package ru.study.universitynavigator.entity;
import lombok.Getter;
import lombok.Setter;
import org.springframework.format.annotation.DateTimeFormat;
import org.springframework.security.core.Authentication;
import org.springframework.security.core.context.
    SecurityContextHolder;
import ru.study.universitynavigator.security.PersonDetails;
import javax.persistence.*;
import java.util.Date;
@Entity
@Getter
@Setter
@Table(name = "Person")
public class Person {
    //Уникальный идентификатор пользователя
    @Id
    @Column(name = "id")
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;
    //Логин пользователя
    @Column(name = "username")
    private String username;
    //Пароль пользователя
    @Column(name = "password")
    private String password;
    //Роль пользователя
    @Column(name = "role")
    private String role;
    //Фамилия пользователя
    @Column(name = "surname")
    private String surname;
    //Имя пользователя
    @Column(name = "firstname")
    private String firstname;
    //Дата рождения пользователя
    @Column(name = "date_of_birth")
    @Temporal(TemporalType.DATE)
    @DateTimeFormat(pattern = "yyyy-MM-dd")
    private Date dateOfBirth;
    //Путь до фото пользователя
    @Column(name = "photo")
    private String photo;
    //Пол пользователя
    @Column(name = "gender")
    private int gender;

```

```
}
```

Листинг Ж.14 – Код CabinetPhotoRepository.java

```
package ru.study.universitynavigator.repository;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;
import ru.study.universitynavigator.entity.CabinetPhoto;
import java.util.List;
@Repository
public interface CabinetPhotoRepository extends
    JpaRepository<CabinetPhoto, Integer> {
    List<CabinetPhoto> findAllByCabinetId(int id);
}
```

Листинг Ж.15 – CabinetRepository.java

```
package ru.study.universitynavigator.repository;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;
import ru.study.universitynavigator.entity.Cabinet;
import java.util.List;
@Repository
public interface CabinetRepository extends JpaRepository<Cabinet,
    Integer> {
    List<Cabinet> findAllByNumber(String number);
    List<Cabinet> findAllByType(String number);
    List<Cabinet> findAllByCapacity(int capacity);
    List<Cabinet> findAllByFloor(int floor);
    List<Cabinet> findAllByEquipmentId(int equipmentId);
    List<Cabinet> findAllByCorpusId(int corpusId);
}
```

Листинг Ж.16 – Код CorpusPhotoRepository.java

```
package ru.study.universitynavigator.repository;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;
import ru.study.universitynavigator.entity.CorpusPhoto;
import java.util.List;
@Repository
public interface CorpusPhotoRepository extends
    JpaRepository<CorpusPhoto, Integer> {
    List<CorpusPhoto> findAllByCorpusId(int id);
}
```

Листинг Ж.17 – Код CorpusRepository.java

```
package ru.study.universitynavigator.repository;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;
import ru.study.universitynavigator.entity.Corpus;
import ru.study.universitynavigator.entity.Equipment;
@Repository
public interface CorpusRepository extends JpaRepository<Corpus,
    Integer> {
    Corpus findByName(String name);
}
```

Листинг Ж.18 – Код DepartmentRepository.java

```
package ru.study.universitynavigator.repository;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;
import ru.study.universitynavigator.entity.Department;
import java.util.List;
@Repository
public interface DepartmentRepository extends
    JpaRepository<Department, Integer> {
    List<Department> findAllByCorpusId(int id);
}
```

Листинг Ж.19 – Код EquipmentRepository.java

```
package ru.study.universitynavigator.repository;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;
import ru.study.universitynavigator.entity.Equipment;
@Repository
public interface EquipmentRepository extends
    JpaRepository<Equipment, Integer> {
    Equipment findByName(String name);
}
```

Листинг Ж.20 – Код PersonRepository.java

```
package ru.study.universitynavigator.repository;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;
import ru.study.universitynavigator.entity.Person;
import java.util.Optional;
@Repository
public interface PersonRepository extends JpaRepository<Person,
    Integer> {
    Optional<Person> findByUsername(String username);
}
```

Листинг Ж.21 – Код PersonDetails.java

```
package ru.study.universitynavigator.security;
import org.springframework.security.core.GrantedAuthority;
import org.springframework.security.core.authority.
    SimpleGrantedAuthority;
import org.springframework.security.core.userdetails.UserDetails;
import ru.study.universitynavigator.entity.Person;
import java.util.Collection;
import java.util.Collections;
public class PersonDetails implements UserDetails {
    private final Person person;
    public PersonDetails(Person person) {
        this.person = person;
    }
    @Override
    public Collection<? extends GrantedAuthority> getAuthorities
        () {
        // SHOW_ACCOUNT, WITHDRAW_MONEY, SEND_MONEY
        // ROLE_ADMIN, ROLE_USER роли
    }
```

```

        return Collections.singletonList(new
            SimpleGrantedAuthority(person.getRole()));
    }
    @Override
    public String getPassword() {
        return this.person.getPassword();
    }
    @Override
    public String getUsername() {
        return this.person.getUsername();
    }
    @Override
    public boolean isAccountNonExpired() {
        return true;
    }
    @Override
    public boolean isAccountNonLocked() {
        return true;
    }
    @Override
    public boolean isCredentialsNonExpired() {
        return true;
    }
    @Override
    public boolean isEnabled() {
        return true;
    }
    //Нужно, чтобы получать данные аутентифицирован пользователя
    public Person getPerson() {
        return this.person;
    }
}

```

Листинг Ж.22 – Код CabinetPhotoService.java

```

package ru.study.universitynavigator.service;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import ru.study.universitynavigator.entity.CabinetPhoto;
import ru.study.universitynavigator.repository.
    CabinetPhotoRepository;
import java.util.List;
@Service
public class CabinetPhotoService {
    private final CabinetPhotoRepository cabinetPhotoRepository;
    @Autowired
    public CabinetPhotoService(CabinetPhotoRepository
        cabinetPhotoRepository) {
        this.cabinetPhotoRepository = cabinetPhotoRepository;
    }
    public List<CabinetPhoto> getPhotoByCabinetId(int id){
        return cabinetPhotoRepository.findAllByCabinetId(id);
    }
}

```

Листинг Ж.23 – Код CabinetService.java

```

package ru.study.universitynavigator.service;

```

```

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import ru.study.universitynavigator.entity.Cabinet;
import ru.study.universitynavigator.entity.Campus;
import ru.study.universitynavigator.entity.Equipment;
import ru.study.universitynavigator.repository.CabinetRepository;
import ru.study.universitynavigator.repository.CampusRepository;
import ru.study.universitynavigator.repository.
    EquipmentRepository;
import java.util.List;
@Service
public class CabinetService {
    private final CabinetRepository cabinetRepository;
    private final CampusRepository campusRepository;
    private final EquipmentRepository equipmentRepository;
    @Autowired
    public CabinetService(CabinetRepository cabinetRepository,
        CampusRepository campusRepository, EquipmentRepository
            equipmentRepository) {
        this.cabinetRepository = cabinetRepository;
        this.campusRepository = campusRepository;
        this.equipmentRepository = equipmentRepository;
    }
    public List<Cabinet> findByNumber(String number) {
        return cabinetRepository.findAllByNumber(number);
    }
    public List<Cabinet> findByType(String type) {
        return cabinetRepository.findAllByType(type);
    }
    public List<Cabinet> findByFloor(int floor) {
        return cabinetRepository.findAllByFloor(floor);
    }
    public List<Cabinet> findByCapacity(int capacity) {
        return cabinetRepository.findAllByCapacity(capacity);
    }
    public List<Cabinet> findByEquipment(String equipment) {
        Equipment equipmentFromRepo = equipmentRepository.
            findByName(equipment);
        return cabinetRepository.findAllByEquipmentId
            (equipmentFromRepo.getId());
    }
    public List<Cabinet> findByCampus(String campus) {
        Campus campusFromRepo = campusRepository.
            findByName(campus);
        return cabinetRepository.findAllByCampusId
            (campusFromRepo.getId());
    }
    public Cabinet findById(int id) {
        return cabinetRepository.findById(id).orElse(null);
    }
}

```

Листинг Ж.24 – Код CorpusPhotoService.java

```

package ru.study.universitynavigator.service;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import ru.study.universitynavigator.entity.CampusPhoto;

```

```

import ru.study.universitynavigator.repository.
    CorpusPhotoRepository;
import java.util.List;
@Service
public class CorpusPhotoService {
    private final CorpusPhotoRepository corpusPhotoRepository;
    @Autowired
    public CorpusPhotoService(CorpusPhotoRepository
        corpusPhotoRepository) {
        this.corpusPhotoRepository = corpusPhotoRepository;
    }
    public List<CorpusPhoto> getPhotoByCorpusId(int id){
        return corpusPhotoRepository.findAllByCorpusId(id);
    }
}

```

Листинг Ж.25 – Код CorpusService.java

```

package ru.study.universitynavigator.service;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import ru.study.universitynavigator.entity.Corpus;
import ru.study.universitynavigator.repository.CorpusRepository;
import java.util.List;
@Service
public class CorpusService {
    private final CorpusRepository corpusRepository;
    @Autowired
    public CorpusService(CorpusRepository corpusRepository) {
        this.corpusRepository = corpusRepository;
    }
    public List<Corpus> findAll(){
        return corpusRepository.findAll();
    }
    public Corpus findById(int id){
        return corpusRepository.findById(id).orElse(null);
    }
}

```

Листинг Ж.26 – Код DepartmentService.java

```

package ru.study.universitynavigator.service;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import ru.study.universitynavigator.entity.Department;
import ru.study.universitynavigator.repository.
    DepartmentRepository;
import java.util.List;
@Service
public class DepartmentService {
    private final DepartmentRepository departmentRepository;
    @Autowired
    public DepartmentService(DepartmentRepository
        departmentRepository) {
        this.departmentRepository = departmentRepository;
    }
    public List<Department> findAllByCorpusId(int id){
        return departmentRepository.findAllByCorpusId(id);
    }
}

```

```

    }
}

```

Листинг Ж.27 – Код PersonDetailsService.java

```

package ru.study.universitynavigator.service;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.core.userdetails.
    UserDetailsService;
import org.springframework.security.core.userdetails.
    UsernameNotFoundException;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;
import ru.study.universitynavigator.entity.Person;
import ru.study.universitynavigator.repository.PersonRepository;
import ru.study.universitynavigator.security.PersonDetails
import java.util.List;
import java.util.Optional;
@Service
public class PersonDetailsService implements UserDetailsService {
    private final PersonRepository personRepository;
    @Autowired
    public PersonDetailsService(PersonRepository
        personRepository) {
        this.personRepository = personRepository;
    }
    @Override
    public UserDetails loadUserByUsername(String s) throws
        UsernameNotFoundException {
        Optional<Person> person = personRepository.
            findByUsername(s);
        if (person.isEmpty())
            throw new UsernameNotFoundException("User not
                found");
        return new PersonDetails(person.get());
    }
    public Person findByUsername(String username) {
        return personRepository.findByUsername(username)
            .orElse(null);
    }
    @Transactional
    public void update(int id, Person person) {
        Person personInBase = personRepository.findById(id)
            .orElse(null);
        if (personInBase != null) {
            person.setPhoto(personInBase.getPhoto());
            person.setFirstname(personInBase.getFirstname());
            person.setSurname(personInBase.getSurname());
            person.setUsername(personInBase.getUsername());
            person.setDateOfBirth(personInBase.getDateOfBirth());
            ;
        }
        person.setId(id);
        personRepository.save(person);
    }
}

```

Листинг Ж.28 – Код RegistrationService.java

```
package ru.study.universitynavigator.service;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.crypto.password.
    PasswordEncoder;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;
import ru.study.universitynavigator.entity.Person;
import ru.study.universitynavigator.repository.PersonRepository;
@Service
public class RegistrationService {
    private final PersonRepository personRepository;
    private final PasswordEncoder passwordEncoder;
    @Autowired
    public RegistrationService(PersonRepository personRepository,
        PasswordEncoder passwordEncoder) {
        this.personRepository = personRepository;
        this.passwordEncoder = passwordEncoder;
    }
    @Transactional
    public void register(Person person) {
        //Кодируем пароль для защиты данных пользователя
        person.setPassword(passwordEncoder.encode(person.getPassword()));
        //Проставляем дефолтную роль
        person.setRole("ROLE_USER");
        //При регист у всех пользован одинаковое дефолтное фото
        person.setPhoto("default.jpg");
        personRepository.save(person);
    }
}
```

Листинг Ж.29 – Код PersonValidator.java

```
package ru.study.universitynavigator.util;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.core.userdetails.
    UsernameNotFoundException;
import org.springframework.stereotype.Component;
import org.springframework.validation.Errors;
import org.springframework.validation.Validator;
import ru.study.universitynavigator.entity.Person;
import ru.study.universitynavigator.service.PersonDetailsService;
@Component
public class PersonValidator implements Validator {
    private final PersonDetailsService personDetailsService;
    @Autowired
    public PersonValidator(PersonDetailsService
        personDetailsService) {
        this.personDetailsService = personDetailsService;
    }
    @Override
    public boolean supports(Class<?> aClass) {
        return Person.class.equals(aClass);
    }
    //Проверяем существует ли такой пользователь в системе
    @Override
```

```

public void validate(Object o, Errors errors) {
    Person person = (Person) o;
    try {
        personDetailsService.loadUserByUsername(person.
            getUsername());
    } catch (UsernameNotFoundException ignored) {
        return; //все ок, пользователь не найден
    }
    errors.rejectValue("username", "", "Человек с таким
        именем пользователя уже существует");
}
}

```

Листинг Ж.30 – Код PhotoUploaderUtil.java

```

package ru.study.universitynavigator.util;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.stereotype.Component;
import org.springframework.web.multipart.MultipartFile;
import java.io.File;
import java.io.IOException;
import java.util.UUID;
@Component
public class PhotoUploaderUtil {
    @Value("${upload.path}")
    private String uploadPath;
    public String uploadPhoto(MultipartFile multipartFile) {
        String resultFilename = new String();
        if (multipartFile != null
            && !multipartFile.getOriginalFilename().isEmpty()) {
            File uploadDir = new File(uploadPath);
            if (!uploadDir.exists()) {
                uploadDir.mkdir();
            }
            String uuidFile = UUID.randomUUID().toString();
            resultFilename = uuidFile + "." +
                multipartFile.getOriginalFilename();
            try {
                multipartFile.transferTo(new File(uploadPath + "/" +
                    resultFilename));
            } catch (IOException ex) {
                System.out.println("Error when trying to upload file
                    " + ex.getLocalizedMessage());
            }
        }
        return resultFilename;
    }
}

```

Листинг Ж.31 – Код cabinet.html

```

<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>Страница кабинета</title>
    <style>
        *{

```

```

        margin: 0;
        padding: 0;
    }
    body {
        background-color: #f8f0f0;
    }
    header {
        width: 100%;
        height: 60px;
        background-color: #343636;
    }
    .top-menu {
        height: 60px;
        float: right;
    }
    .top-menu ul li {
        list-style: none;
        float: left;
        text-align: center;
    }
    .top-menu ul li a {
        font-family: "Helvetica Neue", sans-serif;
        font-size: 12pt;
        display: block;
        width: 171px;
        text-decoration: none;
        color: white;
        height: 40px;
        padding-top: 21px;
        cursor: pointer;
    }
    .top-menu ul li a:hover {
        transition: all 0.5s ease;
        -webkit-transition: all 0.5s ease;
        -o-transition: all 0.5s ease;
        -ms-transition: all 0.5s ease;
        -moz-transition: all 0.5s ease;
        background: #787a7a;
    }
    .cabinet {
        text-align: center;
        padding-top: 20px;
    }
    .photos {
        padding-top: 30px;
    }
    .photo {
        display: inline-block;
        padding-left: 20px;
    }
    .photo-name {
        text-align: center;
        padding-top: 15px;
    }
    img {
        padding-top: 10px;
    }
}
</style>

```

```

</head>
<body>
<header>
    <div class="top-menu">
        <ul>
            <li><a th:href="@{/auth/login}">Вход</a></li>
        </ul>
    </div>
    <div class="top-menu" style="float: left">
        <ul>
            <li><a th:href="@{/main}">Назад</a></li>
        </ul>
    </div>
</header>
<div class="cabinet">
    <h2 class="cabinet" th:text="${cabinet.number}"></h2>
    <p th:text="'Этаж кабинета: ' + ${cabinet.floor}"></p>
    <p th:text="'Вместимость кабинета: ' + ${cabinet.capacity} +
        ' чел.'"></p>
    <p th:text="'Тип кабинета: ' + ${cabinet.type}"></p>
    <p th:text="'Корпус кабинета: ' + ${cabinet.corpus.name}"></p>
    <p th:text="'Оборудование кабинета: ' +
        ${cabinet.equipment.name}"></p>
    <p th:text="'Общая информация: ' + ${cabinet.information}"></p>
    <p></p>
</div>
    <div class="photos" th:if="${photos != null and not
        #lists.isEmpty(photos)}">
        <div class="photo" th:each="photo : ${photos}">
            <p class="photo-name" th:text="${photo.name}"></p>
            
        </div>
    </div>
</div>
</body>
</html>

```

Листинг Ж.32 – Код corpus.html

```

<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
<meta charset="UTF-8">
<title>Страница учебного корпуса</title>
<style>
</style>
</head>
<body>
    <header>
        <div class="top-menu">
            <ul>
                <li><a th:href="@{/auth/login}">Вход</a></li>
            </ul>
        </div>
        <div class="top-menu" style="float: left">
            <ul>
                <li><a th:href="@{/main}">Назад</a></li>
            </ul>
        </div>
    </body>

```

```

</header>
<div class="corpus-box">
  <div class="corpus">
    <h2 class="corpus-name" th:text="${corpus.name}"></h2>
    <p th:text="'Расположен по адресу: ' + ${corpus.address}"></p>
    <p th:text="${corpus.information}"></p>
    <div class="information-message">
      <h3>Фото корпуса доступны только авторизованным
        пользователям</h3>
    </div>
  </div>
</div>
<div class="departments">
  <h2 class="departments-header">Кафедры, расположенные в
    корпусе</h2>
  <div th:if="${departments != null and not
    #lists.isEmpty(departments)}">
    <div th:each="department : ${departments}">
      <div class="departments-box">
        <h3 class="faculty-name"
          th:text="${department.faculty.name}"></h3>
        <h4 class="departments-header" th:text="'Кафедра #' +
          ${department.id}"></h4>
        <h4 class="departments-name" th:text="${department.name}"
          ></h4>
        <p class="department-information" th:text="'Руководитель
          кафедры: ' + ${department.headDep}"></p>
        <hr>
        <p class="department-information" th:text="'Номер
          телефона кафедры: ' + ${department.phone}"></p>
        <hr>
      </div>
    </div>
    <div th:unless="${departments != null and not
    #lists.isEmpty(departments)}">
      <div class="information-message">
        <h3>В указанном учебном корпусе не располагается ни
          одна из кафедр</h3>
      </div>
    </div>
  </div>
</div>
</body>
</html>

```

Листинг Ж.33 – Код corpus-authorization.html

```

<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
<meta charset="UTF-8">
<title>Страница учебного корпуса</title>
<style>
</style>
</head>
<body>
  <header>
    <div class="top-menu">

```

```

        <ul>
            <li><a th:href="@{/auth/login}">Вход</a></li>
        </ul>
    </div>
    <div class="top-menu" style="float: left">
        <ul>
            <li><a th:href="@{/main}">Назад</a></li>
        </ul>
    </div>
</header>
<div class="corpus-box">
    <div class="corpus">
        <h2 class="corpus-name" th:text="{corpus.name}"></h2>
        <p th:text="'Расположен по адресу: ' +
            {corpus.address}"></p>
        <p th:text="{corpus.information}"></p>
        <div class="photos" th:if="{photos != null and not
            #lists.isEmpty(photos)}">
            <div class="photo" th:each="photo : {photos}">
                <p class="photo-name" th:text="{photo.name}">
                    "></p>
                
            </div>
        </div>
    </div>
</div>
<div class="departments">
    <h2 class="departments-header">Кафедры, расположенные в
        корпусе</h2>
    <div th:if="{departments != null and not
        #lists.isEmpty(departments)}">
        <div th:each="department : {departments}">
            <div class="departments-box">
                <h3 class="faculty-name"
                    th:text="{department.faculty.name}"></h3>
                <h4 class="departments-header" th:text="'Кафедра #'
                    + {department.id}"></h4>
                <h4 class="departments-name"
                    th:text="{department.name}"></h4>
                <p class="department-information"
                    th:text="'Руководитель кафедры: ' +
                        {department.headDep}"></p>
                <hr>
                <p class="department-information"
                    th:text="'Номер телефона кафедры: ' +
                        {department.phone}"></p>
                <hr>
            </div>
        </div>
    </div>
    <div th:unless="{departments != null and not
        #lists.isEmpty(departments)}">
        <div class="departments-not-found">
            <h3>В указанном учебном корпусе не располагается ни
                одна из кафедр</h3>
        </div>
    </div>
</div>
</div>

```

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

Листинг Ж.34 – Код edit-profile.html

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>Редактирование профиля</title>
    <style>
    </style>
</head>
<body>
<div id="block-body">
    <header>
        <div class="top-menu" style="float: left">
            <ul>
                <li><a style="float: left" th:href="
                    @{/authorized/main}">Назад</a></li>
            </ul>
        </div>
        <div class="top-menu" style="float: right">
            <div class="top-menu">
                <ul>
                    <li><a th:href="@{/logout}">Выход</a></li>
                </ul>
            </div>
        </div>
    </header>
<div id="block-content">
    <div style="max-width: 500px; margin-left: auto; margin-right:
    auto; padding: 15px;">
        <form th:method="POST" th:action="@{/authorized/update-
            person/{id} (id=${person.getId()})}" th:object="$
                {person}" enctype="multipart/form-data">
            <div class="input-photo__wrapper">
                <div>
                    <label for="photo" class="text-
                        field__label">Фотография профиля: </label>
                </div>
                <div>
                    
                </div>
                <input style="padding-bottom: 10px" type="file"
                    name="photo" accept="image/*">
            </div>
            <div class="input__wrapper">
                <label class="text-field__label" for="username">Введите
                    логин пользователя (не больше
                    25 символов)</label>
                <input class="text-field__input" type="text"
                    th:field="*{username}" id="username" required
                    placeholder="Логин пользователя"/>
                <div style="color:red" th:if="${#fields.hasErrors
                    ('username')}" th:errors="*{username}">Username
                    Error
            </div>
        </form>
    </div>
</div>
```

```

        </div>
    </div>
    <div class="input__wrapper">
        <label class="text-field__label" for="firstname">
            Введите имя(не больше 25 символов)</label>
        <input class="text-field__input" type="text"
            th:field="*{firstname}" id="firstname"
            placeholder="Введите имя" required/>
        <div style="color:red" th:if="${#fields.hasErrors
            ('firstname')}" th:errors="*{firstname}">Firstna
            me
        Error
        </div>
    </div>
    <div class="input__wrapper">
        <label class="text-field__label" for="surname">Введите
        фамилию(не больше 25 символов)</label>
        <input class="text-field__input" type="text"
            th:field="*{surname}" id="surname"
            name="surname" required placeholder="
            Введите фамилию"/>
        <div style="color:red" th:if="${#fields.
            hasErrors('surname')}" th:errors="*{surname}"
            >Surname Error
        </div>
    </div>
    <div class="input__wrapper">
        <label class="text-field__label">Выберите ваш
            пол</label>
        <label class="text-field__label-radio">
        <input class="" th:value="${1}" th:field="*{gender}"
            type="radio">
            Мужской
        </label>
        <label class="text-field__label-radio">
        <input class="" th:value="${2}" th:field="*{gender}"
            type="radio">
            Женский
        </label>
    </div>
    <div class="input__wrapper">
        <label class="text-field__label" for="date_of_birth"
            >Дата рождения</label>
        <input class="text-field__input" type="date"
            th:field="*{dateOfBirth}" id="date_of_birth"
            required/>
    </div>
    <input type="hidden" th:value="${person.id}" id="person"
    name="person">
    <div style="margin-top: 15px">
        <input class="submit" type="submit" value=
            "Сохранить"/>
    </div>
</form>
</div>
</div>
</div>
</body>

```

```
</html>
```

Листинг Ж.35 – Код error.html

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
<meta charset="UTF-8">
<title>Ошибка</title>
    <style>
    </style>
</head>
<body>
<header>
    <div class="top-menu">
        <ul>
            <li><a th:href="@{/auth/registration}"
                >Регистрация</a></li>
        </ul>
    </div>
</header>
<div th:if="${errors}">
    <h2 class="error" th:each="error : ${errors}"
        th:text="${error.defaultMessage}"></h2>
</div>
</body>
</html>
```

Листинг Ж.36 – Код login.html

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
<meta charset="UTF-8">
<title>Авторизация</title>
    <style>
    </style>
</head>
<body>
<header>
    <div class="top-menu" style="float: right">
        <ul>
            <li><a th:href="@{/auth/registration}"
                ">Регистрация</a></li>
        </ul>
    </div>
    <div class="top-menu" style="float: left">
        <ul>
            <li><a th:href="@{/main}">Назад</a></li>
        </ul>
    </div>
</header>
<div id="block-content" style="max-width: 400px; margin-left: auto;
margin-right: auto; padding: 15px;">
    <form name="f" method="post" action="/process_login">
        <input type="hidden" th:name="${_csrf.parameterName}"
            th:value="${_csrf.token}">
        <div class="input__wrapper">
```

```

        <label class="text-field__label" for="username">
            Введите имя пользователя</label>
        <input class="text-field__input" type="text"
            name="username" id="username" placeholder="Имя
            пользователя"/>
    </div>
    <div class="input__wrapper">
        <label class="text-field__label" for="password">
            Введите пароль</label>
        <input class="text-field__input" type="password"
            name="password" id="password" placeholder=
            "Пароль"/>
    </div>
    <input class="submit" type="submit" value="Войти"/>
    <div th:if="${param.error}" style="color: red; text-
        align: center">
        Неправильные имя или пароль
    </div>
</form>
</div>
</body>
</html>

```

Листинг Ж.37 – Код main.html

```

<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
<meta charset="UTF-8">
<title>Основная страница</title>
    <style>
    </style>
</head>
<body>
<div id="block-body">
    <header>
        <div class="top-menu">
            <ul>
                <li><a th:href="@{/auth/login}">Вход</a></li>
            </ul>
        </div>
    </header>
    <div class="searchbar">
        <h2 class="searchbar-header">Введите критерии поиска
            кабинета</h2>
        <h5 class="searchbar-information-header">Внимание! Для
            неавторизованных пользователей доступен поиск только
            по номеру кабинета</h5>
        <form style="text-align: center; padding-bottom: 10px"
            action="/search" method="GET">
            <input type="text" name="number" placeholder="
                Введите номер кабинета...">
            <button type="submit">Найти</button>
        </form>
        <div th:if="${cabinets != null and not #lists.isEmpty
            (cabinets)}">
            <div th:each="cabinet : ${cabinets}">
                <div class="cabinets-box">

```

```

<p class="cabinet" th:text="\${cabinet.number}"></p>
<hr>
<p class="cabinet-information" th:text="'Этаж кабинета: ' + \${cabinet.floor}"></p>
<p class="cabinet-information" th:text="'Вместимость кабинета: ' + \${cabinet.capacity} + ' чел.'"></p>
<p class="cabinet-information" th:text="'Тип кабинета: ' + \${cabinet.type}"></p>
<p class="cabinet-information" th:text="'Корпус кабинета: ' + \${cabinet.corpus.name}"></p>
<p class="cabinet-information" th:text="'Оборудование кабинета: ' + \${cabinet.equipment.name}"></p>
<p class="cabinet-information" th:text="'Общая информация: ' + \${cabinet.information}"></p>
<hr>
<div class="more-information" style="text-align: center">
  <form th:action="@{'/search/cabinet/' + \${cabinet.getId()}}" method="GET">
    <button type="submit">Подробная информация
  </button>
</form>
</div>
</div>
</div>
</div>
<div class="corpuses">
  <h2 class="corpuses-header">Учебные корпуса</h2>
  <div th:each="corpus : \${corpuses}">
    <div class="corpuses-box">
      <p class="corpus" th:text="\${corpus.name}"></p>
      <h2 th:text="\${corpus.address}"></h2>
      <hr>
      <p class="corpus-information" th:text="\${corpus.information}"></p>
      <hr>
      <div class="more-information" style="text-align: center">
        <form th:action="@{'/search/corpus/' + \${corpus.getId()}}" method="GET">
          <button type="submit">Подробная информация
        </button>
      </form>
    </div>
  </div>
</div>
</div>
</div>
</body>
</html>

```

Листинг Ж.38 – Код main-authorization.html

```

<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
<meta charset="UTF-8">
<title>Основная страница</title>
    <style>
    </style>
</head>
<body>
<div id="block-body">
    <header>
        <div class="top-menu">
            <ul>
                <li><a th:href="@{/auth/login}">Вход</a></li>
            </ul>
        </div>
    </header>
    <div class="searchbar">
        <h2 class="searchbar-header">Введите критерии поиска кабинета</h2>
        <h5 class="searchbar-information-header">Внимание! Для неавторизованных пользователей доступен поиск только по номеру кабинета</h5>
        <form style="text-align: center; padding-bottom: 10px" action="/search" method="GET">
            <input type="text" name="number" placeholder="Введите номер кабинета...">
            <button type="submit">Найти</button>
        </form>
        <div th:if="${cabinets != null and not #lists.isEmpty(cabinets)}">
            <div th:each="cabinet : ${cabinets}">
                <div class="cabinets-box">
                    <p class="cabinet" th:text="${cabinet.number}"></p>
                    <hr>
                    <p class="cabinet-information" th:text="'Этаж кабинета: ' + ${cabinet.floor}"></p>
                    <p class="cabinet-information" th:text="'Вместимость кабинета: ' + ${cabinet.capacity} + ' чел.'"></p>
                    <p class="cabinet-information" th:text="'Тип кабинета: ' + ${cabinet.type}"></p>
                    <p class="cabinet-information" th:text="'Корпус кабинета: ' + ${cabinet.corpus.name}"></p>
                    <p class="cabinet-information" th:text="'Оборудование кабинета: ' + ${cabinet.equipment.name}"></p>
                    <p class="cabinet-information" th:text="'Общая информация: ' + ${cabinet.information}"></p>
                    <hr>
                    <div class="more-information" style="text-align: center">
                        <form th:action="@{/search/cabinet/' + ${cabinet.getId()}}" method="GET">
                            <button type="submit">Подробная информация</button>

```

```

                                </form>
                            </div>
                        </div>
                    </div>
                </div>
            </div>
</div>
<div class="corpuses">
    <h2 class="corpuses-header">Учебные корпуса</h2>
    <div th:each="corpus : ${corpuses}">
        <div class="corpuses-box">
            <p class="corpus" th:text="${corpus.name}"></p>
            <h2 th:text="${corpus.address}"></h2>
            <hr>
            <p class="corpus-information"
              th:text="${corpus.information}"></p>
            <hr>
            <div class="more-information" style="text-align:
              center">
                <form th:action="@{'/search/corpus/' +
                  ${corpus.getId()}}" method="GET">
                    <button type="submit">Подробная информация
                </button>
                </form>
            </div>
        </div>
    </div>
</div>
</div>
</body>
</html>

```

Листинг Ж.39 – Код registration.html

```

<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
<meta charset="UTF-8">
<title>Регистрация</title>
</head>
<body>
<div id="block-body">
    <header>
        <div class="top-menu" style="float: right">
            <ul>
                <li><a th:href="@{/main}">Основная
                  страница</a></li>
            </ul>
        </div>
        <div class="top-menu" style="float: left">
            <ul>
                <li><a th:href="@{/auth/login}">Назад</a></li>
            </ul>
        </div>
    </header>
    <div id="block-content" style="max-width: 500px; margin-left: auto;
      margin-right: auto; padding: 15px">
        <h1>Зарегистрируйте пользователя</h1>
    </div>

```

```

<form th:method="POST" th:action="@{/auth/registration}"
th:object="${person}">
    <div class="input__wrapper">
        <label class="text-field__label" for="username"
            ">Введите имя пользователя</label>
        <input class="text-field__input" type="text"
            th:field="*{username}" id="username" required
            placeholder="Имя пользователя"/>
        <div style="color:red" th:if="${#fields.
            hasErrors('username')}}" th:errors="*
                {username}">Username Error
        </div>
    </div>
</div>
    <div class="input__wrapper">
        <label class="text-field__label" for="firstname"
            ">Введите имя</label>
        <input class="text-field__input" type="text"
            th:field="*{firstname}" id="firstname"
            placeholder="Введите имя" required/>
        <div style="color:red" th:if="${#fields.
            hasErrors('firstname')}}" th:errors="*
                {firstname}">Firstname Error
        </div>
    </div>
</div>
    <div class="input__wrapper">
        <label class="text-field__label" for="surname"
            ">Введите фамилию</label>
        <input class="text-field__input" type="text"
            th:field="*{surname}" id="surname"
            name="surname" required
            placeholder="Введите фамилию"/>
        <div style="color:red" th:if="${#fields.
            hasErrors('surname')}}" th:errors="*
                {surname}">Surname Error
        </div>
    </div>
</div>
    <div class="input__wrapper">
        <label class="text-field__label">Выберите ваш
            пол</label>
        <label class="text-field__label-radio">
            <input class="" th:value="${1}" th:field="*{gender}"
                type="radio"> Мужской
            </label>
        <label class="text-field__label-radio">
            <input class="" th:value="${2}" th:field="*{gender}"
                type="radio"> Женский
            </label>
        </div>
</div>
    <div class="input__wrapper">
        <label class="text-field__label" for="date_of_birth"
            ">Дата рождения</label>
        <input class="text-field__input" type="date"
            th:field="*{dateOfBirth}" id="date_of_birth"
            required/>
        </div>
</div>
    <div class="input__wrapper">
        <label class="text-field__label"
            for="password">Введите пароль</label>

```

```
<input class="text-field__input" type="password"
      th:field="*{password}" required id="password"
      placeholder="Пароль"/>
<div style="color:red" th:if="${#fields.
      hasErrors('password')}" th:errors="*
      {password}">Password Error
    </div>
  </div>
  <input class="submit" type="submit"
    value="Зарегистрироваться"/>
</form>
</div>
</div>
</body>
</html>
```