

**МИНОБРНАУКИ РОССИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ ИМ. Р.Е. АЛЕКСЕЕВА»**

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

Липекина Алексея Игоревича
(фамилия, имя, отчество)

Институт ИРИТ

Кафедра ВСТ

Группа 18ВМв

Дата защиты « ____ » _____

Индекс

МИНОБРНАУКИ РОССИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ ИМ. Р.Е. АЛЕКСЕЕВА»
(НГТУ)

Институт радиоэлектроники и информационных технологий

Направление подготовки (специальность) 09.03.01 Информатика и вычислительная техника

Направленность (профиль) образовательной программы _____

(наименование)

Кафедра «Вычислительные системы и технологии» (ВСТ)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

Бакалавра

(бакалавра, магистра, специалиста)

Студента Липекина Алексея Игоревича группы 18 ВМв
(Ф.И.О.)

на тему «Автоматизированная система построения и анализа карты загруженности учебных аудиторий.»

(наименование темы работы)

СТУДЕНТ:

(подпись) Липекин А.И.
(фамилия, и., о.)

(дата)

КОНСУЛЬТАНТЫ:

1. По _____
(подпись) _____
(фамилия, и., о.)

(дата)

РУКОВОДИТЕЛЬ:

(подпись) Кулясов П.С.
(фамилия, и., о.)

(дата)

2. По _____
(подпись) _____
(фамилия, и., о.)

(дата)

РЕЦЕНЗЕНТ:

(подпись) _____
(фамилия, и., о.)

(дата)

3. По _____
(подпись) _____
(фамилия, и., о.)

(дата)

ЗАВЕДУЮЩИЙ КАФЕДРОЙ

(подпись) Жевнерчук Д.В.
(фамилия, и.о.)

(дата)

ВКР защищена _____
(дата)

протокол № _____

с оценкой _____

Кафедра «Вычислительные системы и технологии»

« » 20 г.

Country	Share of GDP
United States	1.0%
Germany	0.8%
France	0.7%
Italy	0.6%
Spain	0.5%
Japan	0.4%
China	0.3%

6. Консультанты по ВКР (с указанием относящихся к ним разделов)

Нормоконтроль

7. Дата выдачи задания 15.03.2023

Код и содержание Компетенции	Задание	Проектируемый результат	Отметка о выполнении
ОПК-1. Способен применять естественнонаучные и общинженерные знания, методы математического анализа и моделирования, теоретического и экспериментального исследования в профессиональной деятельности			
ОПК-2. Способен понимать принципы работы современных информационных технологий и программных средств, в том числе отечественного производства, и использовать их для решения задач профессиональной деятельности			
ОПК-3. Способен решать стандартные задачи профессиональной деятельности на основе информационной и библиографической культуры с применением информационно-коммуникационных технологий и с учетом основных требований информационной безопасности			
ОПК-4. Способен участвовать в разработке стандартов, норм и правил, а также технической документации, связанной с профессиональной деятельностью			
ОПК-5. Способен устанавливать программное и аппаратное обеспечение для информационных и автоматизированных систем			
ОПК-6. Способен разрабатывать бизнес-планы и технические задания на оснащение отделов, лабораторий, офисов			

компьютерным и сетевым оборудованием			
ОПК-7. Способен участвовать в настройке и наладке программно- аппаратных комплексов			
ОПК-8. Способен разрабатывать алгоритмы и программы, пригодные для практического применения			
ОПК-9. Способен осваивать методики использования программных средств для решения практических задач			
ПКС-1. Способен разрабатывать модели компонентов и алгоритмы функционирования вычислительной техники и автоматизированных систем			
ПКС-2. Способен сопрягать аппаратные и программные средства и обеспечивать их функционирование в составе вычислительных и автоматизированных систем			
ПКС-3. Способен участвовать в работах по обеспечению эффективного функционирования сетевых устройств, серверного программного обеспечения информационно- коммуникационных систем			
УК-1. Способен осуществлять поиск, критический анализ и синтез информации, применять системный подход для решения поставленных задач			
УК-2. Способен определять круг задач в рамках поставленной цели и выбирать оптимальные способы их решения, исходя из действующих правовых норм, имеющихся ресурсов и ограничений			
УК-3. Способен осуществлять социальное взаимодействие и реализовывать свою роль в команде			
УК-4. Способен осуществлять деловую коммуникацию в устной и письменной формах на государственном языке Российской Федерации и иностранным(ых) языке(ах)			
УК-5. Способен воспринимать межкультурное разнообразие			

общества в социально-историческом, этическом и философском контекстах			
УК-6. Способен управлять своим временем, выстраивать и реализовывать траекторию саморазвития на основе принципов образования в течение всей жизни			
УК-7. Способен поддерживать должный уровень физической подготовленности для обеспечения полноценной социальной и профессиональной деятельности			
УК-8. Способен создавать и поддерживать в повседневной жизни и в профессиональной деятельности безопасные условия жизнедеятельности для сохранения природной среды, обеспечения устойчивого развития общества, в том числе при угрозе и возникновении чрезвычайных ситуаций и военных конфликтов			
УК-9. Способен принимать обоснованные экономические решения в различных областях жизнедеятельности			
УК-10. Способен формировать нетерпимое отношение к коррупционному поведению			

Руководитель _____
(подпись)

Задание принял к исполнению _____
(дата)

Студент _____
(подпись)

Примечания:

1. Это задание прилагается к законченной работе и в составе пояснительной записки предоставляется в ГЭК.
2. До начала консультаций студент должен составить и утвердить у руководителя календарный график работы на весь период выполнения ВКР (с указанием сроков выполнения и трудоемкости отдельных этапов).

МИНОБРНАУКИ РОССИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ ИМ. Р.Е. АЛЕКСЕЕВА»
(НГТУ)

АННОТАЦИЯ

к выпускной квалификационной работе

по направлению подготовки (специальности) 09.03.01 Информатика и вычислительная техника студента Липекина Алексея Игоревича группы 18ВМв
(Ф.И.О.)

по теме «Автоматизированная система построения и анализа карты загруженности учебных аудиторий.»

Выпускная квалификационная работа выполнена на _____ страницах, содержит _____ диаграмм, _____ таблиц, библиографический список из _____ источников, _____ приложений.

Актуальность: Выбранная задача является актуальной и практически значимой, так как система управления и бронирования аудиторий в образовательных учреждениях всегда является сложным аспектом в работе и включает в себя составление расписаний занятий, разрешения возникающих конфликтов. Традиционные методы составления расписаний занятий часто предполагают ручное ведение электронных таблиц, это отнимает много времени и чревато ошибками.

Объект исследования: Система управления и бронирования аудиторий в образовательных учреждениях всегда является сложным аспектом в работе, так как включает в себя составление расписаний занятий, разрешения возникающих конфликтов. Традиционные методы составления расписаний занятий часто предполагают ручное ведение электронных таблиц, это отнимает много времени и чревато ошибками.

Предмет исследования: Разработка системы для преобразования расписания из электронных таблиц в веб-приложение с возможностью бронирования аудиторий.

Цель исследования: Создать эффективную, удобную и интерактивную систему для управления расписанием, с возможностью визуализировать, бронировать и получать дополнительную информацию о расписании.

Задачи исследования:

1. Анализ требований и потребностей системы управления у.
2. Разработка архитектуры системы.
3. Разработка интерфейса системы.
4. Реализация функционала системы.

Методы исследования: Анализ, моделирование системы, программирование.

Структура работы:

Во введении...

В 1 разделе «Название».

Во 2 разделе «Название»...

В 3 разделе «Название»...

В заключении...

Выводы:

1.

2.

Рекомендации:

1.

2.

подпись студента /расшифровка подписи

« ____ » _____ 20 ____ г.

Содержание

Введение.....	5
Глава 1 Постановка задачи.....	6
1.1 Общие сведения.....	6
1.2 Анализ существующих систем для управления процессами учебного заведения.	6
Глава 2 Изучение методов и подходов для создания веб-приложения	15
2.1 Сравнительный анализ языков программирования.....	16
2.2 Исследование фреймворков Python и их применение.....	19
2.3 Выбор базы данных и вспомогательных компонентов	21
2.4 Результаты анализа методов и средств для разработки	23
Глава 3 Реализация веб-приложения «Портал учебных аудиторий»	23
3.1 Анализ входных данных.....	23
3.2 Создание модели базы данных	24
3.2 Описание основных алгоритмов.....	36
3.4 Создание интерфейса и структуры веб-приложения.....	49
3.5 Тестирование	58
Заключение	58
Список литературы	59

					ВКР-НГТУ-09.03.01-18-ВМв-03-2023									
Изм.	Лист	Л	№ докум.	№	Подпись	Дата	Д							
Разраб.			Липекин А.И.					Лит.		Лит.	Лист	Лис	Листов	Листов
Провер.												4	57	
Реценз.								4						
Н. Контр. Н.			Шагалова П.А.											
Утверд.														

Введение

Область образования постоянно развивается, и с каждым годом становится более зависимой от технологий. В настоящее время существует достаточное количество решений для управления учебными процессами, но большинство этих решений упирается в необходимость больших денежных вложений и обеспечении постоянной технической поддержки для поддержания работоспособности. Одной из ключевых областей, где внедрение технологий может повысить эффективность - управление аудиториями.

Традиционные методы управления аудиториями в учебных заведениях зачастую неэффективны и могут приводить к недостаточному использованию ресурсов. Например, одни аудитории могут быть перегружены, в то время как другие остаются неиспользуемыми, поэтому учебные заведения могут столкнуться с проблемами в планировании расписаний, управлении ресурсами и обеспечении качества образования.

Данная работа направлена на разработку веб-приложения, которое поможет с анализом и представлением данных об использовании учебных аудиторий. Система предназначена для упрощения процесса управления аудиториями, позволяя загружать данные о расписании в формате .xlsx, которые затем анализируются и сохраняются в базе данных SQLite. Система генерирует графики загрузки аудиторий, отображает список занятых аудиторий, свободных аудиторий, с возможностью бронирования свободных аудиторий.

Таким образом, проект направлен на автоматизацию процесса управления аудиториями и повышение эффективности использования ресурсов в учебном заведении.

Глава 1 Постановка задачи

1.1 Общие сведения

Целью работы является создание веб-приложения для помощи в автоматизации и анализа управления учебными аудиториями для улучшения управления процессами учебного заведения.

Чтобы достичь поставленной цели необходимо последовательно решить следующие задачи:

- Первым шагом будет проведение анализа графика учебного расписания, для выявления ключевых показателей с последовательным извлечением данных и последующей обработки этих данных;

- Следующим этапом станет анализ имеющиеся на сегодняшний день инструментов для создания веб-приложений. Исходя из этого анализа, необходимо определить наиболее подходящие инструменты для достижения поставленной цели.

- После этого, необходимо спроектировать универсальную базу данных для выявления критериев загруженности аудиторий по данным, выгруженным из учебного расписания;

- Затем, следует перейдем к алгоритмам, которые будут использоваться в процессе разработки программного продукта. Описание каждого из используемых алгоритмов обеспечит прозрачность и понимание всего процесса;

- По завершению разработки, следует провести тестирование программного решения, чтобы убедиться, что оно работает так, как предполагалось, и отвечает всем установленным требованиям.

1.2 Анализ существующих систем для управления процессами учебного заведения.

Для того, чтобы эффективно управлять образовательными учреждениями, прежде всего, необходимо провести анализ уже существующих программных

продуктов в этой сфере. Особое внимание стоит уделить приложениям, которые способствуют автоматизации управления процессами обучения и координации хозяйственной деятельности. Ключевыми функциями таких систем являются контроль и управление процессами учебного плана, учащимися, преподавательским составом и ресурсами учебного заведения. Рассмотрим основные функции данных систем, построим сравнительную таблицу и выделим ключевые недостатки.

1.2.1 Moodle (Modular Object-Oriented Dynamic Learning Environment)

Moodle (Modular Object-Oriented Dynamic Learning Environment) - это бесплатная платформа для электронного обучения с открытым исходным кодом. Эта система была разработана Мартином Дугиамасом и впервые представлена в 2001 году. Moodle используется в образовательных учреждениях и организациях схожего характера для создания и управления курсами в онлайн-формате.

Основные функции и возможности Moodle:

- Создание контента курсов: Moodle предлагает широкий спектр инструментов для создания обучающего контента, включая тексты, изображения, видео, ссылки на внешние ресурсы и т.д. Преподаватели могут создавать разнообразные активности, такие как форумы, викторины, задания, опросы и многое другое.
- Управление курсами: можно создавать курсы, устанавливать правила доступа к ним, создавать группы студентов и управлять их работой.
- Оценка и отслеживание прогресса: Moodle имеет встроенные инструменты для оценки работы студентов и отслеживания их прогресса. Преподаватели могут настраивать различные методы оценки, включая ручную оценку и автоматическую оценку заданий и тестов.
- Коммуникация и взаимодействие: можно настроить форумы, чаты и сообщения для коммуникации между преподавателями и студентами, а также между самими студентами.

- Плагины и дополнения: платформа поддерживает расширение функционала при помощи плагинов, что позволяет преподавателям и администраторам настраивать систему под конкретные нужды образовательного процесса.

1.2.2 Blackboard Learn

Blackboard Learn - это коммерческая платформа для управления обучением (LMS), предлагающая интегрированные инструменты и сервисы для создания, распространения и управления онлайн-обучением.

Ключевые особенности Blackboard Learn:

- Создание и управление контентом: Blackboard предлагает простые и интуитивно понятные инструменты для создания курсов и контента, включая загрузку текстовых файлов, изображений, видео и других медиа-ресурсов. Также доступен функционал для создания интерактивных заданий, тестов и опросов.

- Управление курсами: платформа позволяет создавать и настраивать курсы, управлять студентами и их группами, а также создавать индивидуальные планы обучения.

- Коммуникация: Blackboard Learn предлагает инструменты для коммуникации, такие как электронная почта, чаты и форумы, а также инструменты для совместной работы, включая общие документы и вики-библиотеки.

- Отслеживание и аналитика: Blackboard предоставляет детализированные отчеты об успеваемости студентов и статистику использования курсов, что позволяет преподавателям и администраторам следить за прогрессом студентов и оптимизировать учебный процесс.

- Интеграция: Blackboard Learn может интегрироваться с различными системами, включая системы управления студентами, электронные библиотеки и другие образовательные платформы.

1.2.3 Галактика Управление вузом

«Галактика Управление вузом» - это автоматизированная информационная система, созданная российской компанией "Галактика" для комплексной автоматизации учебного процесса и управления в образовательных учреждениях. Это система электронного документооборота, организации учебного процесса и его административного управления.

Основные функциональные возможности системы:

- Организация учебного процесса: система обеспечивает автоматизацию процесса создания учебных планов, составления расписаний, проведения экзаменов и контроля успеваемости студентов.
- Управление документооборотом: платформа позволяет автоматизировать процесс обработки, хранения и передачи документов, связанных с учебным процессом.
- Управление финансами и ресурсами: «Галактика Управление вузом» включает функции для планирования и контроля бюджета, а также управления материально-техническими ресурсами учебного заведения.
- Отчетность и аналитика: система предоставляет функции для сбора и анализа данных об образовательной деятельности для создания различных отчетов и принятия управленческих решений.

1.2.4 Юнивуз-3.0

Юнивуз - это российская автоматизированная система управления учебной деятельностью. Она обеспечивает полный цикл управления образовательной организацией, включая организацию учебного процесса, управление документооборотом, отчетность и аналитику.

Основные функции:

- Управление учебным процессом: Юнивуз позволяет управлять учебными планами, расписаниями, проводить контроль знаний студентов, организовывать дистанционное обучение.

- Управление документооборотом: в системе реализован автоматизированный документооборот, который обеспечивает эффективную организацию работы с документами.

- Аналитика и отчетность: система обеспечивает сбор, хранение и обработку информации о деятельности образовательного учреждения для анализа и подготовки отчетности.

- Управление ресурсами: Юнивуз позволяет эффективно распределять учебные и административные ресурсы, учитывая текущие потребности и планы учебного заведения.

1.2.5 Сравнение ключевого функционала систем

Ниже представлена «Таблица 1», составленная по результатам сравнительного анализа по критериям, указанным в первом столбце таблицы.

	Moodle	Blackboard	Галактика Управление вузом	Юнивуз
Целевое использование	Универсальная система, доступная всем образовательным учреждениям.	Коммерческая система, часто используется в вузах и колледжах.	Автоматизированная система управления для вузов.	Автоматизированная система управления для вузов.
Управление учебным процессом	Управление курсами, создание и оценка заданий, форумы, встроенные инструменты для оценки и	Управление курсами, создание и оценка заданий, встроенные инструменты для оценки и	Автоматизация создания учебных планов, составления расписаний, проведения экзаменов и контроля	Управление учебными планами, расписаниями, контроль знаний студентов, организация дистанционного обучения.

	отслеживания прогресса.	отслеживания прогресса.	успеваемости студентов.	
Управление ресурсами	Создание виртуальных классов/аудиторий , управление доступом к учебным материалам.	Управление расписанием занятий, доступом к учебным материалам и цифровым ресурсам.	Планирование и контроль бюджета, управление материально- техническими ресурсами, планирование и распределение аудиторий.	Эффективное распределение учебных и административных ресурсов, управление распределением аудиторий.
Коммуникация	Форумы, чаты, сообщения, плагины для расширения коммуникационны х возможностей.	Электронная почта, чаты, форумы, общие документы и вики-библиотеки	Автоматизированны й документооборот.	Автоматизированны й документооборот.
Отчетность и аналитика	Отчеты об успеваемости студентов и статистика использования курсов.	Детализированны е отчеты об успеваемости студентов и статистика использования курсов.	Функции для сбора и анализа данных об образовательной деятельности для создания отчетов и принятия управленческих решений.	Сбор, хранение и обработка информации о деятельности образовательного учреждения для анализа и подготовки отчетности.
Управление физическими аудиториями	Не предусмотрено в базовой функциональност и	Не предусмотрено в базовой функциональност и	Планирование и распределение аудиторий в соответствии с расписанием и потребностями.	Планирование и распределение аудиторий в соответствии с расписанием и потребностями.

Управление виртуальными аудиториями	Создание и управление виртуальными классами, управление доступом к курсам и материалам.	Создание и управление виртуальными классами, управление доступом к курсам и материалам.	Не предусмотрено в базовой функциональности.	Не предусмотрено в базовой функциональности.
Управление загрузенностью	Отслеживание активности и загрузенности виртуальных аудиторий, отчеты о прогрессе студентов.	Отслеживание активности и загрузенности виртуальных классов, детальные отчеты о прогрессе студентов.	Управление загрузенностью аудиторий и распределение нагрузки сотрудников.	Управление загрузенностью аудиторий и распределение нагрузки сотрудников.

Таблица 1- сравнение функциональности

1.2.6 Основные недостатки систем

Среди недостатков вышеописанных систем управления учебными процессами можно отметить следующие:

Moodle:

- Требуется технических навыков для установки и настройки. Необходимость самостоятельного хостинга может представлять проблему для некоторых учебных заведений.

- Интерфейс может быть сложным и непривычным для пользователей, которые ранее не работали с подобными системами.

- Возможно появится необходимость в дополнительных плагинах для расширения функциональности, что может увеличивать сложность использования и поддержки системы.

Blackboard:

- Высокая стоимость лицензий и обслуживания.
- Некоторые владельцы системы считают интерфейс неинтуитивным и устаревшим.
- Могут возникнуть сложности при интеграции с другими уже внедренными системами.

Галактика Управление вузом:

- Высокая стоимость внедрения и поддержки.
- Сложность в освоении и использовании системы без специального обучения.
- Отсутствие или недостаточная поддержка виртуальных аудиторий в базовой функциональности.

Юнивуз:

- Высокая стоимость внедрения и поддержки.
- Сложность в освоении и использовании системы без специального обучения.
- Может быть сложность в интеграции с другими системами и сервисами.

Ниже приведена «Таблица 2» демонстрирующая результаты анализа существующих систем управления образовательными процессами. В процессе их сравнительного исследования, применяются условные оценочные показатели. Эти показатели представляют собой переменные, отражающиеся на числовой шкале [1]. Таким образом, числовым оценкам от 1 до 5 соответствует категория от «очень плохо» до «отлично».

	Moodle	Blackboard	Галактика Управление вузом	Юнивуз
Удобство использования	3	3	3	3
Гибкость и настраиваемость	4	3	3	3
Управление учебным процессом	4	4	4	4
Управление ресурсами	2	2	4	4
Отчетность и аналитика	3	4	4	4

После изучения различных программных решений, предназначенных для управления загруженностью учебных аудиторий, мы можем определить несколько основных требований к созданию конкурентоспособной системы:

- Процесс назначения и изменения расписания занятий в конкретных аудиториях должен быть простым и быстрым.
- Система должна предлагать интуитивно понятные и адаптивные шаблоны для визуализации использования аудиторий, облегчая планирование.
- Расписания и назначения аудиторий должны быть доступны по уникальным URL-адресам для удобства навигации и идентификации.
- Использование системы не должно повлечь к большим денежным затратам.
- Приоритет должен отдаваться простому и интуитивно понятному интерфейсу для управления расписанием, занятостью аудиторий.
- В системе должен быть предусмотрен удобный механизм загрузки и обновления расписаний и назначений аудиторий, возможно, с использованием единого файла, содержащего всю необходимую информацию.

Функционал?

Глава 2 Изучение методов и подходов для создания веб-приложения

Для создания веб-приложений доступно множество технологий. Самой простой среди этих технологий являются конструкторы сайтов и CMS-технологии. **ссылка** Однако эти решения обычно ограничивают функциональность, а последующее развитие сайта становится зависимым от финансовых затрат необходимых на расширение функционала. Кроме того, важно учитывать, что стоимость поддержания расширенного функционала может требовать регулярных вложений, что также является фактором, который может повлиять на выбор технологии для создания веб-приложения. Из-за этих проблем, более перспективным решением часто является создание веб-приложения "с нуля" с использованием специализированного фреймворка, что гарантирует большую гибкость и контроль над функционалом и стоимостью развития веб-приложения в долгосрочной перспективе.

В процессе разработки веб-приложения важно внимательно выбрать технологический стек, актуальный на момент создания проекта. Нужно начать с клиентской части приложения, где ошибочный выбор может привести к серии трудностей при разработке и поддержке приложения. Первоначально, необходимо определиться с языком программирования для клиентской и серверной части. Следующий этап - выбор фреймворка, подходящего под выбранный язык программирования. Далее, необходимо определить технологию или инструмент для редактирования стилей приложения. Что касается серверной части, то здесь требуется выбрать подходящий фреймворк, а также базу данных для создания, хранения и изменения информации обрабатываемой веб-приложением. Все эти решения формируют итоговый технологический стек, который в значительной степени определяет качество и удобство работы с веб-приложением. Для начала необходимо сделать выбор в пользу языка программирования, на котором лучше создать веб-приложение.

2.1 Сравнительный анализ языков программирования

Выбор языка программирования - это ключевой шаг в процессе разработки веб-приложения, так как от него зависит последующий выбор инструментов и методов реализации. Подходящий язык программирования предоставляет удобные инструменты для создания и управления хранилищем данных на сервере, а также реализации клиент-серверного взаимодействия. Перед выбором конкретного языка важно провести анализ существующих языков программирования и их сравнение по различным параметрам. Рассмотрим такие распространенные языки, как JavaScript, Python и PHP, которые часто используются в разработке веб-приложений, и проведем их сравнительный анализ.

PHP - это один из самых распространенных языков программирования для веб-разработки. Он был создан специально для работы с веб-приложениями, что отчасти объясняет его популярность.

Несколько ключевых преимуществ PHP:

- PHP работает на сервере, и результат его работы - это уже готовый HTML-код, который может быть прочитан любым браузером.
- PHP относительно легко изучить, особенно по сравнению с некоторыми другими языками программирования. Его синтаксис логичен и понятен, что делает его хорошим выбором для начинающих разработчиков.
- PHP имеет огромное сообщество разработчиков и большое количество ресурсов для изучения. Это обеспечивает быстрое решение проблем и помощь в устранении возникающих ошибок.
- PHP имеет встроенную поддержку для множества систем управления базами данных, включая MySQL, PostgreSQL и SQLite.
- PHP поддерживается почти всеми веб-серверами и большинством операционных систем. Это делает его очень гибким и универсальным инструментом.

- PHP, как правило, обрабатывает скрипты быстро, особенно по сравнению с некоторыми другими языками программирования. Это означает, что веб-сайты и приложения на PHP загружаются быстро, обеспечивая хорошее взаимодействие с пользователями.

JavaScript является еще одним важным языком программирования для веб-разработки. Изначально предназначенный для создания интерактивности на стороне клиента, JavaScript эволюционировал и сегодня используется на обеих сторонах - и на сервере (Node.js), и на клиенте.

Преимущества JavaScript:

- JavaScript позволяет создавать динамические и интерактивные веб-сайты и приложения, обрабатывая действия пользователя на стороне клиента без необходимости перезагрузки страницы.

- JavaScript работает в браузере пользователя, что уменьшает нагрузку на сервер и может улучшить производительность веб-сайта.

- Благодаря Node.js, JavaScript можно использовать не только на клиенте, но и на сервере. Это позволяет разработчикам использовать один и тот же язык программирования на всех уровнях приложения.

- JavaScript имеет одно из самых больших сообществ разработчиков, поэтому при возникновении проблем всегда можно найти помощь и ресурсы для обучения.

- Существует множество фреймворков и библиотек JavaScript (например, React, Angular, Vue.js), которые упрощают разработку и добавляют новые функции.

- Все современные браузеры поддерживают JavaScript без необходимости установки каких-либо дополнительных плагинов.

- JavaScript является языком, на котором основан формат JSON, широко используемый для обмена данными в веб-разработке.

Python - это универсальный язык программирования, который нашел широкое применение в веб-разработке. Он предлагает набор уникальных преимуществ, которые делают его привлекательным выбором для создания веб-приложений.

Основные преимущества Python:

- Python обладает чистым и простым синтаксисом, который делает его легким для чтения и обучения. Это также упрощает процесс отладки и сокращает время разработки.

- Python имеет огромную экосистему библиотек и фреймворков, которые упрощают создание веб-приложений. Django и Flask - два популярных фреймворка для веб-разработки на Python.

- Python поддерживает несколько парадигм программирования, включая процедурное, объектно-ориентированное и функциональное программирование.

- Python хорошо интегрируется с другими технологиями и платформами. Это делает его универсальным языком для веб-разработки.

- Python обычно требуется меньше кода для реализации тех же функций, что и многие другие языки. Это ускоряет процесс разработки и сокращает затраты на разработку.

- Python имеет одно из самых больших и активных сообществ разработчиков. Это означает, что разработчики всегда могут найти помощь и обучающие материалы.

- Python не только отлично подходит для веб-разработки, но и широко используется в таких областях как машинное обучение, наука о данных, автоматизация и других.

Ниже находится сравнительная «Таблица 3» языков Python, PHP, JavaScript по наиболее важным критериям, влияющим на разработку веб-приложений.

Критерий\Язык	Python	PHP	JavaScript
Уровень сложности обучения	Низкая	Средняя	Средняя
Поддержка объектно-ориентированного программирования	Полная	Полная	Прототипная
Автоматическая очистка неиспользуемых данных	Да	Да	Да
Множественное наследование	Нет (но есть миксины)	Нет	Нет
Наличие библиотек	Огромное количество	Большое количество	Большое количество
Поддержка многомерных массивов	Да	Да	Да
Наличие фреймворков	Огромное количество (Django, Flask)	Большое количество (Laravel, Symfony)	Большое количество (Angular, React, Vue)
Поддержка	Отличная	Хорошая	Хорошая

Таблица 3- сравнение языков

По итогу сравнительного анализа был выбран язык программирования Python по нескольким ключевым причинам: простота и читаемость, богатый набор библиотек и фреймворков, поддержка многопоточности и асинхронности, портативность и межплатформенность, большое и активное сообщество.

2.2 Исследование фреймворков Python и их применение

После выбора языка программирования, следующий важный шаг - это определение подходящего фреймворка для разработки веб-приложений. Фреймворки играют ключевую роль в том, каким образом пользователи

взаимодействуют с созданным приложением через интерфейс браузера. Они обеспечивают упрощение процесса разработки, позволяя эффективно интегрировать различные компоненты программной системы для построения качественного продукта.

Когда дело доходит до выбора подходящего фреймворка для веб-разработки на Python, рынок предлагает множество вариантов, включая такие как Django, Flask, Pyramid и другие. Стоит отметить, что наиболее используемыми веб-фреймворками являются - Django и Flask (см. Рисунок **). Именно эти два инструмента заслуживают особого внимания и детального анализа из-за своей высокой популярности среди разработчиков веб-приложений.

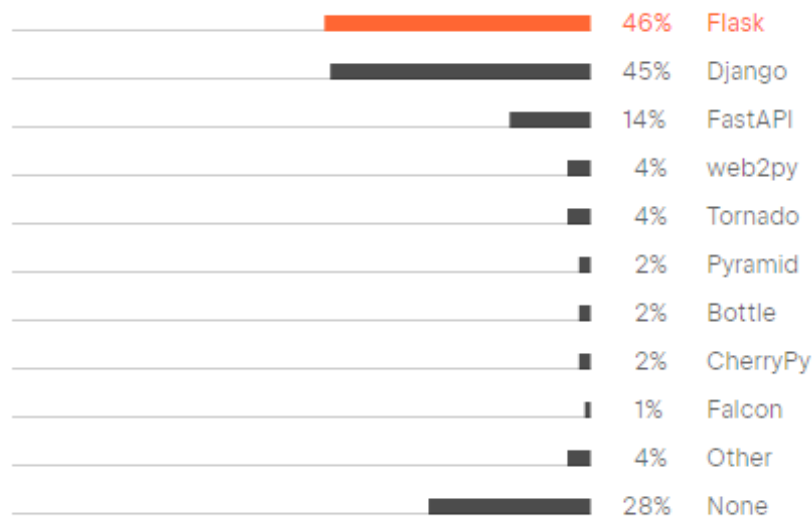


Рисунок ** - сравнение фреймворков Python за 2022 год по статистике сайта JetBrains

Фреймворк Flask был разработан в 2010 году австрийским программистом Ронахером А., а Django появился ещё раньше, в 2005 году, благодаря стараниям Django Software Foundation. Обе эти платформы обладают открытым исходным кодом и бесплатны для использования.

Когда дело доходит до различий между Flask и Django, можно выделить несколько основных моментов. Для начала, Django чаще применяется в разработке

больших и сложных веб-приложений, в то время как Flask обычно используют для более простых и маленьких проектов с ограниченным количеством функций.

Кроме того, документация Django более обширная и детальная, что делает процесс обучения этому фреймворку более понятным и простым. Flask также имеет документацию, но она менее объемная и детальная.

Третьим важным отличием является наличие встроенной ORM в Django, которая облегчает работу с различными базами данных. Flask, с другой стороны, не имеет встроенной ORM, хотя поддерживает внешние, например, SQLAlchemy.

Четвертый момент - при разработке приложений с Django автоматически создается строго структурированная система директорий, в то время как при использовании Flask структуру приложения приходится создавать самостоятельно. Это может показаться более простым решением, но для сложных проектов с большим количеством функций структурирование кода в Flask может стать сложнее из-за отсутствия встроенных инструментов для этого.

Один из наиболее мощных аспектов Django - это автоматически генерируемый интерфейс администратора для управления базой данных приложения.

С учетом всех этих аспектов, для разработки веб-приложения в данном случае более подходящим выбором становится Django.

2.3 Выбор базы данных и вспомогательных компонентов

Выбор системы управления базами данных (СУБД) [ссылка](#) является важным этапом при подготовке к разработке. Django, по умолчанию, использует SQLite в качестве такой системы. При использовании SQLite в приложении, обмен данными осуществляется через прямые и функциональные вызовы к файлу, хранящему данные, вместо стандартного интерфейса. Этот подход значительно упрощает обработку запросов сервером и делает SQLite быстрым и эффективным решением. Мощность этой СУБД обеспечивается с помощью основных технологий

библиотеки. Стоит отметить, что SQLite находится в списке десяти самых используемых систем управления базами данных, подтверждая свою популярность и надёжность (см. Рисунок **). [ссылка](#)

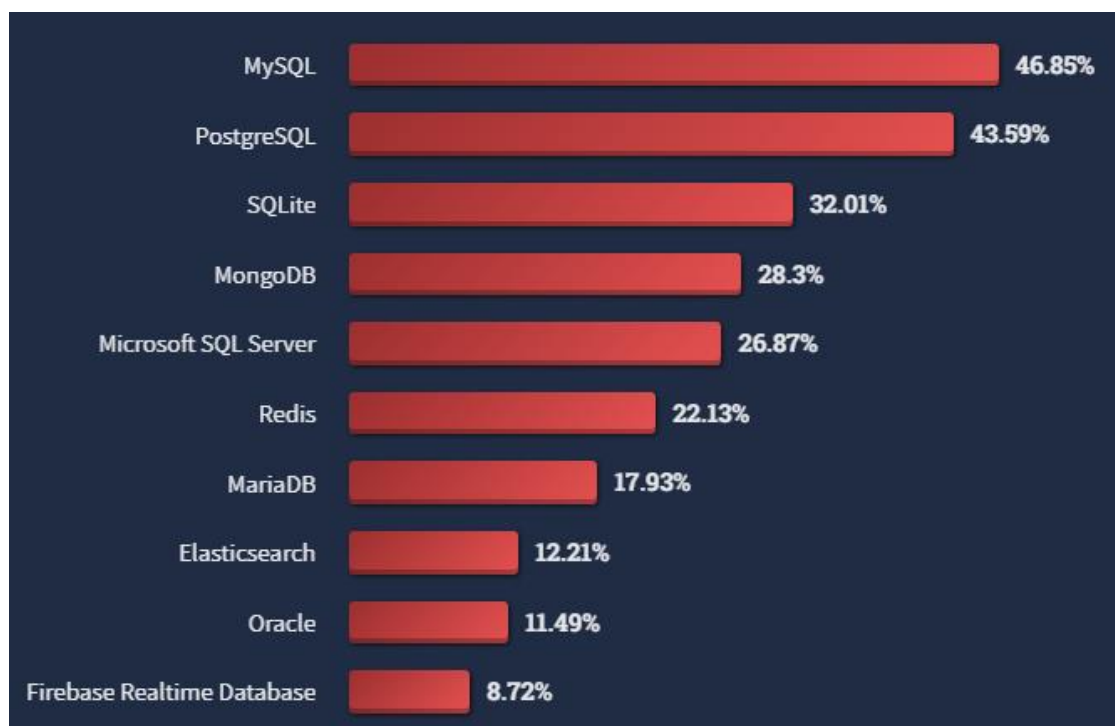


Рисунок ** - самые популярные базы данных за 2022 год по версии сайта stackoverflow

Инструментарием для разработки интерфейса сайта выбран Twitter Bootstrap (Bootstrap), и этот выбор обусловлен нынешним высоким уровнем развития данного фреймворка. Bootstrap обеспечивает возможность эффективного создания веб-интерфейсов для разнообразных сайтов. Bootstrap зарекомендовал себя как один из самых популярных инструментов для верстки, и это связано с его рядом значительных преимуществ. Основным из них является Less - динамический язык стилей, который расширяет стандартный функционал CSS. Less предоставляет разработчикам больше возможностей, позволяя им создавать переменные, вложенные колонки, эффективно управлять цветами и многое другое. Плюс ко всему, работа с Less не требует особых знаний и способностей, он легко осваивается.

2.4 Результаты анализа методов и средств для разработки

Исходя из проведенного анализа, для создания приложения определен следующий набор технологий: программирование на Python, использование фреймворка Django для Python, верстка на HTML, стилизация с помощью CSS Bootstrap, за управление базами данных отвечает SQLite.

Глава 3 Реализация веб-приложения «Портал учебных аудиторий»

3.1 Анализ входных данных

Данные для приложения предоставлены в формате электронных таблиц **xlsx**, содержащих расписание занятий для групп студентов в университете с информацией о дате, времени, месте (аудитории), предмете и преподавателе, а также информацией о типе занятия (лекция, практика, лабораторная работа). Скриншот таблицы ниже.

	2 сем.	22 ИВТ-4 (44)	22 ИВТ-2 (41)	22 ИВТ-3 (39)
Понедельник	1 сем.	а.6501 Философия пр. ин. - 1 п/тр. чк. - 2 п/тр. Михайлова Т.Л.	а.6350 Безопасность жизнедеятельности лаб. ин. - 1 п/тр. чк. - 1 п/тр. Ельин А.Б.	
	2 сем.	Иностранный язык (англ.) пр. 1, 2 п/тр. Борова А.В. а.6435 Беломоно Е.А. а.6519	а.6501 Философия пр. ин. - 1 п/тр. чк. - 2 п/тр. Михайлова Т.Л.	а.6350 Безопасность жизнедеятельности лаб. ин. - 2 п/тр. Ельин А.Б.
	3 сем.	а.6155 Математика пр. Юсупов С.А.	а.6435 Иностранный язык (англ.) пр. 1, 2 п/тр. Борова А.В. Беломоно Е.А. а.6519	а.6501 Философия пр. ин. - 1 п/тр. чк. - 2 п/тр. Михайлова Т.Л.
	4 сем.	а.6519 Иностранный язык (англ.) пр. 3 п/тр. Беломоно Е.А.	а.6152 Математика пр. Горохова И.В.	а.6347 Безопасность жизнедеятельности пр. Ельин А.Б.
	5 сем.		а.6519 Иностранный язык (англ.) пр. 3 п/тр. Беломоно Е.А.	а.6412 Иностранный язык (англ.) пр. 1 п/тр. Коргунова И.А.
	6 сем.			чк. а.6429 Час куратора 2, 6, 10, 14 недели
Вторник	1 сем.		нч. а.6347 Безопасность жизнедеятельности пр. Ельин А.Б.	нч. а.6433 Иностранный язык (англ.) пр. 2 п/тр. Голованова Л.Н.
	2 сем.	чк. а.6347 Безопасность жизнедеятельности пр. Ельин А.Б.	нч. а.6125 Философия лекция доц. Михайлова Т.Л.	
	3 сем.		чк. а.6125 Безопасность жизнедеятельности лекция доц. Ельин А.Б.	
	4 сем.		а.6125 Математика лекция доц. Багеев А.В.	
	5 сем.		а.6125 Программирование лекция ст. пр. Мартынов Д.С.	
	6 сем.		нч. а.6252 Программирование лаб. 3, 4 п/тр. Мартынов Д.С.	нч. а.6412 Иностранный язык (англ.) пр. 1 п/тр. Коргунова И.А.
				чк. а.6433 Иностранный язык (англ.) пр. 2 п/тр. Голованова Л.Н.

Рисунок 3.1 - содержание **xlsx** файла

В файле есть несколько листов, содержащих разные группы по разным специальностям специально.

Если рассмотреть типы данных в этом файле:

- Текстовые данные: дни недели, название предметов, имена преподавателей, типы занятий и т.д.

- Цифровые данные: номера аудиторий, номера подгрупп, время начала и окончания пар.

- Смешанные данные: информация, которая включает в себя и текст, и числа, например, формат времени (7:30/8:00) и обозначение номера группы и семестра (22 ИВТ-4).

Такое расписание можно представить в виде структурированного датасета с колонками: День, Пара, Аудитория, Предмет, Тип занятия, Неделя, Подгруппа, Преподаватель. Представление данных таким образом позволит проще обрабатывать и анализировать информацию, включая загруженность учебных аудиторий.

3.2 Создание модели базы данных

Приложение работает с реляционной базой данных SQLite, взаимодействуя с ней через фреймворк Django. Одной из особенностей Django является то, что дизайн таблиц осуществляется в коде, точнее, в модуле `models.py`, используя технологию ORM. Для создания новой таблицы в базе данных, в Django задается новый класс, который наследуется от `Django.models`. Здесь каждый атрибут класса представляет поле в таблице, и каждый такой класс соответствует отдельной таблице в базе данных. Django позволяет использовать всю мощь реляционных баз данных, связывая таблицы через отношения "один к одному", "один ко многим" и "многие ко многим", а также определяя характеристики каждого поля. Django предлагает разные типы полей для использования, включая:

- Charfield – для текстовых строк небольшого размера;

IntegerField – для целых чисел;

DecimalField – идеален для поля "цена";

FileField – предназначен для хранения файлов и может генерировать ссылки на них;


ImageField – в основном похож на FileField, но оптимизирован для работы с изображениями;

UUIDField – специальное поле для uuid;

FloatField – для чисел с плавающей точкой;

BooleanField – для булевых значений.

Это не полный список типов полей, предоставляемых Django. Существует также другие, включая те, которые поставляются со сторонними библиотеками.

Для наглядного представления структуры базы данных была разработана UML диаграмма классов (см. Рисунок  **).

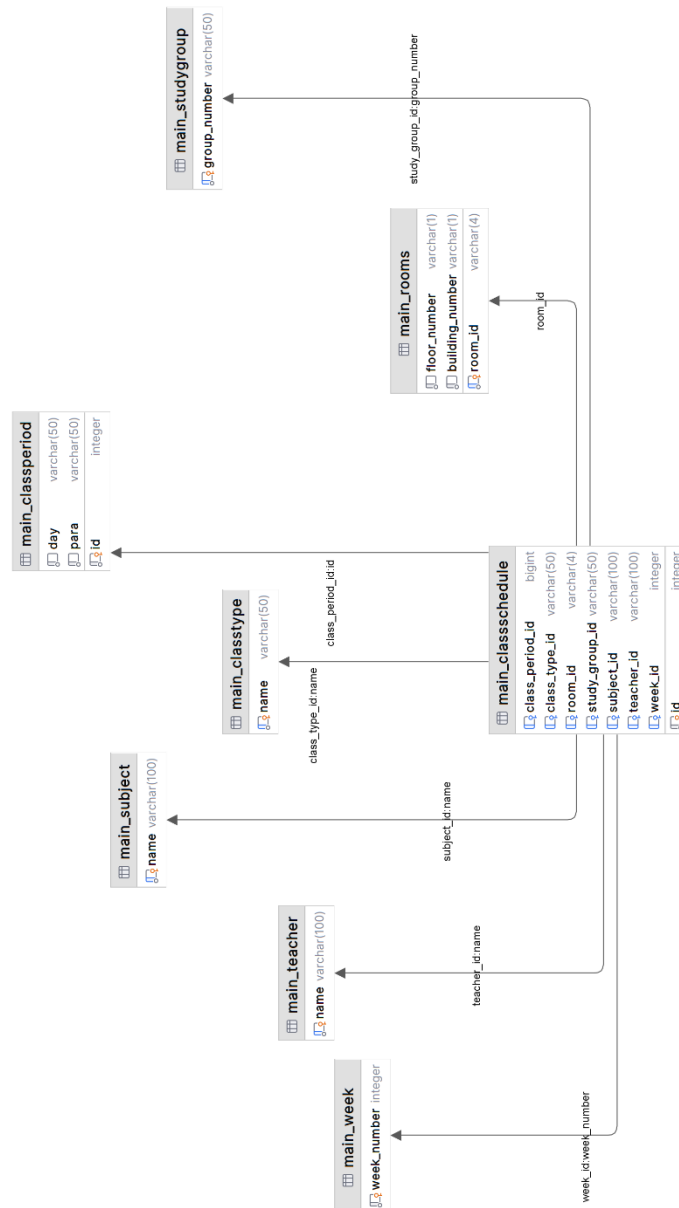


Рисунок 2.2 - диаграмма спроектированной базы данных

Основываясь на разработанной диаграмме классов, были созданы модели в Django для приложения. Следующие основные объекты были определены как ключевые элементы нашей предметной области:

- Rooms, представляет собой аудитории;
- Teacher, эта модель представляет преподавателей;
- Subject, это предмет, который преподается
- ClassType, это тип занятия;

- StudyGroup, это учебная группа, для которой проводится занятие;
- ClassPeriod, это время занятия;
- Week, это учебная неделя, когда проводится занятие;
- ClassSchedule, эта модель представляет конкретное расписание занятий.

В данной модели используются связи "один ко многим" (ForeignKey), которые представляют отношения между различными моделями.

- ClassSchedule и Rooms: Отношение "один ко многим" между аудиторией и расписанием занятий означает, что каждое занятие проводится в одной определенной аудитории, но каждая аудитория может быть использована для множества занятий.

- ClassSchedule и Teacher: Отношение "один ко многим" между преподавателем и расписанием занятий означает, что каждое занятие ведет один определенный преподаватель, но каждый преподаватель может вести несколько занятий.

- ClassSchedule и Subject: Отношение "один ко многим" между предметом и расписанием занятий означает, что каждое занятие соответствует одному предмету, но каждый предмет может быть преподаван на нескольких занятиях.

- ClassSchedule и ClassType: Отношение "один ко многим" между типом занятий и расписанием означает, что каждое занятие имеет один тип (например, лекция, семинар и т.д.), но каждый тип может быть использован на множестве занятий.

- ClassSchedule и StudyGroup: Отношение "один ко многим" между учебной группой и расписанием занятий означает, что каждое занятие предназначено для одной определенной группы, но каждая группа может иметь множество занятий.

- ClassSchedule и ClassPeriod: Отношение "один ко многим" между временем занятий и расписанием занятий означает, что каждое занятие соответствует одному временному периоду, но каждый временной период может быть использован для множества занятий.

ClassSchedule и Week: Отношение "один ко многим" между неделями и расписанием занятий означает, что каждое занятие проводится на определенной неделе, но каждая неделя может содержать множество занятий.

Таким образом модель ClassSchedule связывает все другие модели вместе, представляя конкретное занятие, которое проводится в определенной аудитории, с определенным преподавателем, на определенный предмет, определенного типа, для определенной группы, в определенное время и на определенной неделе. При внесении изменения в классы моделей обязательно необходимо проводить миграцию для применения этих изменений командами «python3 manage.py makemigrations» и «python3 manage.py migrate».

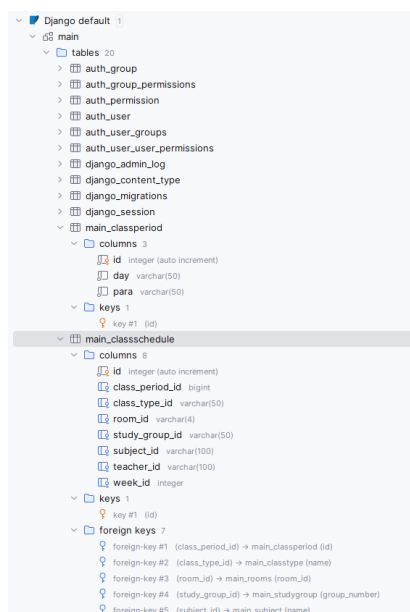


Рисунок 2.1 - база данных

В модель **Rooms** входит 3 поля:

- room_id: это уникальный идентификатор каждой аудитории, который также является первичным ключом.
- floor_number: это номер этажа, на котором находится аудитория.
- building_number: это номер здания, в котором находится аудитория.

	room_id	floor_number	building_n...
3	6410	4	6
4	6429	4	6
5	6505	5	6
6	6506	5	6
7	6347	3	6
8	6313	3	6
9	6128	1	6
10	5306	3	5
11	5303	3	5
12	6309	3	6
13	6151	1	6
14	6350	3	6

Рисунок 2.2 - модель Rooms

Листинг кода класса Rooms в файле models.py:

```
class Rooms(models.Model):
    room_id = models.CharField('Номер аудитории', max_length=4, primary_key=True)
    floor_number = models.CharField('Номер этажа', max_length=1)
    building_number = models.CharField('Номер корпуса', max_length=1)

    def __str__(self):
        return self.room_id

class Meta:
    verbose_name = 'Аудитория'
    verbose_name_plural = 'Аудитории'
```

В модель **Teacher** состоит из одного поля:

- name: это имя преподавателя, которое также является первичным ключом.

	name
1	Приблудова Е.Н.
2	Новоселова Н.А.
3	Крылова А.В.
4	Ерофеева А.В.
5	Белименко Е.А.
6	Раевская Ю.В.
7	Малахов В.А.
8	Шерстнева Л.В.
9	Проскурякова М.П.
10	Горохова И.В.
11	Конюхова Н.С.
12	Попова С.В.

Рисунок ** - модель Teacher

Листинг кода класса Teacher в файле models.py:

```
class Teacher(models.Model):
    name = models.CharField(max_length=100, primary_key=True)

    def __str__(self):
        return self.name

class Meta:
    verbose_name = 'Преподаватель'
    verbose_name_plural = 'Преподаватели'
```

Модель **Subject** состоит из одного поля:

- name: это название предмета, которое также является первичным ключом.

	name
1	Элективный курс по физической культуре и спорту
2	Информационные технологии
3	Физика
4	Великая Отечественная война без срока давности
5	Час куратора
6	Иностранный язык
7	Безопасность жизнедеятельности
8	История
9	Математика
10	Основы численных методов
11	Информатика
12	Электронные модели изделий электронных средств
13	Практикум по физике
14	Графические информационные технологии
15	Философия
16	Алгоритмы и структуры данных
17	Физические основы информационно-коммуникационных систем
18	Программирование на языках высокого уровня

Рисунок ** - модель Subject

Листинг кода класса Subject в файле models.py:

```
class Subject(models.Model):
    name = models.CharField(max_length=100, primary_key=True)

    def __str__(self):
        return self.name

class Meta:
    verbose_name = 'Предмет'
    verbose_name_plural = 'Предметы'
```

Модель **ClassType** также содержит одно поле:

- name: это название типа занятия, которое также является первичным ключом.

Листинг кода класса **ClassType** в файле models.py:

```
class ClassType(models.Model):
    name = models.CharField(max_length=50, primary_key=True)

    def __str__(self):
        return self.name
```

Модель **StudyGroup** содержит поле:

- group_number: это номер группы, который также является первичным ключом.

	group_number
1	22 РЭС (36)
2	22 Р 1 (19)
3	22 Р 2 (17)
4	22 КТЭС (26)
5	22ИТС (28)
6	22 ИСТ - 1 (41)
7	22 ИСТ - 2 (38)
8	22 ИСТ - 3 (35)
9	22 ИСТ - 4-1 (24)
10	22 ИСТ - 4-2 (24)
11	22 ИВТ-4 (44)
12	22 ИВТ-2 (41)
13	22 ИВТ-3 (39)
14	22 СИБ (29)
15	22 ПМ 1 (25)
16	22 ПМ 2 (23)

Рисунок 3.3- модель StudyGroup

Листинг кода класса **StudyGroup** в файле models.py:

```
class StudyGroup(models.Model):
    group_number = models.CharField(max_length=50, primary_key=True)

    def __str__(self):
        return self.group_number

class Meta:
    verbose_name = 'Группа'
    verbose_name_plural = 'Группы'
```

Модель **ClassPeriod** состоит из двух полей:

- day: это день недели, когда проводится занятие;
- para: это номер урока (или пары) в этот день.

	id	day	para
1	1	Понедельник	1 пара
2	2	Понедельник	2 пара
3	3	Понедельник	3 пара
4	4	Понедельник	4 пара
5	5	Понедельник	5 пара
6	6	Понедельник	6 пара
7	7	Вторник	1 пара
8	8	Вторник	2 пара
9	9	Вторник	3 пара
10	10	Вторник	4 пара
11	11	Вторник	5 пара
12	12	Вторник	6 пара

Рисунок 2.2 - модель ClassPeriod

Листинг кода класса **ClassPeriod** в файле models.py:

```
class ClassPeriod(models.Model):
    DAY_CHOICES = [
        ('Понедельник', 'Понедельник'),
        ('Вторник', 'Вторник'),
        ('Среда', 'Среда'),
        ('Четверг', 'Четверг'),
        ('Пятница', 'Пятница'),
        ('Суббота', 'Суббота'),
    ]
    PARA_CHOICES = [
```

```

('1 пара', '1 пара'),
('2 пара', '2 пара'),
('3 пара', '3 пара'),
('4 пара', '4 пара'),
('5 пара', '5 пара'),
('6 пара', '6 пара'),
]
day = models.CharField(max_length=50, choices=DAY_CHOICES)
para = models.CharField(max_length=50, choices=PARA_CHOICES)

def __str__(self):
    return f'{self.day}, {self.para}'

```

Модель **Week** состоит из двух полей:

- `week_number`: это номер недели в учебном году, который также является первичным ключом. Есть ограничение, что номер недели должен быть между 1 и 18.

🔍 week_number 🔍	
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	10
11	11
12	12
13	13

Рисунок 3.3 - модель `Week`

Листинг кода класса **Week** в файле `models.py`:

```

class Week(models.Model):
    week_number = models.IntegerField(primary_key=True)

    def __str__(self):
        return str(self.week_number)

class Meta:

```

```

verbose_name = 'Неделя'
verbose_name_plural = 'Недели'
constraints = [
    models.CheckConstraint(check=models.Q(week_number__gte=1, week_number__lte=18),
name='week_number_range'),
]

```

Модель **ClassSchedule** состоит из семи полей:

- room: это аудитория, в которой проводится занятие. Это внешний ключ к модели Rooms.
- teacher: это преподаватель, который ведет занятие. Это внешний ключ к модели Teacher.
- subject: это предмет, который преподается. Это внешний ключ к модели Subject.
- class_type: это тип занятия. Это внешний ключ к модели ClassType.
- study_group: это учебная группа, для которой проводится занятие. Это внешний ключ к модели StudyGroup.
- class_period: это время занятия. Это внешний ключ к модели ClassPeriod.
- week: это неделя учебного года, когда проводится занятие. Это внешний ключ к модели Week.

id	class_period_id	room_id	study_group_id	subject_id	teacher_id	week_id
1	1	7 ... 6339	22 РЭС (36)	Информационные технологии	Приблудова Е.Н.	1
2	2	7 ... 6339	22 РЭС (36)	Информационные технологии	Приблудова Е.Н.	2
3	3	7 ... 6339	22 РЭС (36)	Информационные технологии	Приблудова Е.Н.	3
4	4	7 ... 6339	22 РЭС (36)	Информационные технологии	Приблудова Е.Н.	4
5	5	7 ... 6339	22 РЭС (36)	Информационные технологии	Приблудова Е.Н.	5
6	6	7 ... 6339	22 РЭС (36)	Информационные технологии	Приблудова Е.Н.	6
7	7	7 ... 6339	22 РЭС (36)	Информационные технологии	Приблудова Е.Н.	7
8	8	7 ... 6339	22 РЭС (36)	Информационные технологии	Приблудова Е.Н.	8
9	9	7 ... 6339	22 РЭС (36)	Информационные технологии	Приблудова Е.Н.	9
10	10	7 ... 6339	22 РЭС (36)	Информационные технологии	Приблудова Е.Н.	10
11	11	7 ... 6339	22 РЭС (36)	Информационные технологии	Приблудова Е.Н.	11
12	12	7 ... 6339	22 РЭС (36)	Информационные технологии	Приблудова Е.Н.	12
13	13	7 ... 6339	22 РЭС (36)	Информационные технологии	Приблудова Е.Н.	13
14	14	7 ... 6339	22 РЭС (36)	Информационные технологии	Приблудова Е.Н.	14
15	15	7 ... 6339	22 РЭС (36)	Информационные технологии	Приблудова Е.Н.	15
16	16	7 ... 6339	22 РЭС (36)	Информационные технологии	Приблудова Е.Н.	16
17	17	7 ... 6339	22 РЭС (36)	Информационные технологии	Приблудова Е.Н.	17
18	18	7 ... 6339	22 РЭС (36)	Информационные технологии	Приблудова Е.Н.	18

Рисунок 4.4 - модель ClassSchedule

Листинг кода класса **ClassSchedule** в файле models.py:

```
class ClassSchedule(models.Model):
    room = models.ForeignKey(Rooms, on_delete=models.CASCADE, null=True,
verbose_name='Аудитория')
    teacher = models.ForeignKey(Teacher, on_delete=models.CASCADE, null=True,
verbose_name='Преподаватель')
    subject = models.ForeignKey(Subject, on_delete=models.CASCADE, null=True,
verbose_name='Предмет')
    class_type = models.ForeignKey(ClassType, on_delete=models.CASCADE, null=True)
    study_group = models.ForeignKey(StudyGroup, on_delete=models.CASCADE, null=True,
verbose_name='Группа')
    class_period = models.ForeignKey(ClassPeriod, on_delete=models.CASCADE, null=True)
    week = models.ForeignKey(Week, on_delete=models.CASCADE, null=True,
verbose_name='Неделя')

    def get_verbose_field_names(self):
        return {
            'room': self._meta.get_field('room').verbose_name,
            'study_group': self._meta.get_field('study_group').verbose_name,
            'teacher': self._meta.get_field('teacher').verbose_name,
            'subject': self._meta.get_field('subject').verbose_name,
        }
```

В базе данных также имеются табличные структуры, необходимые для функционирования Django. Присутствуют такие таблицы, как auth_permissions, django_migrations, dhango_sessions, django_site, socialaccount_socialaccount и ряд других. Таким выглядит полная модель базы данных приложения, на рисунке ниже.

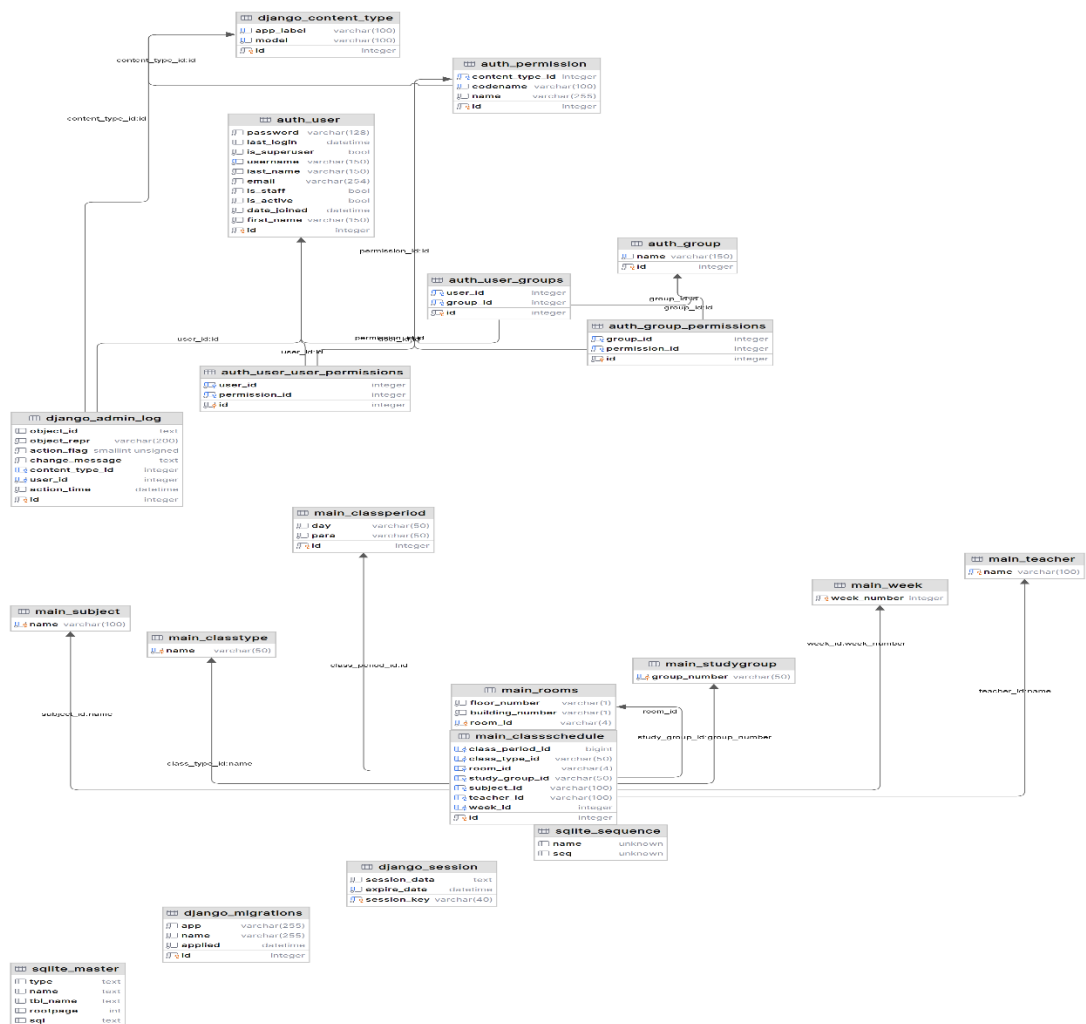


Рисунок 3.2 - база данных всего проекта Django

3.2 Описание основных алгоритмов

3.2.1 Алгоритм поиска

После анализа исходных данных необходимо приступить к извлечению данных из файла электронной таблицы `xlsx` для заполнения базы данных первоначальными базовыми значениями на основе которых будет строиться расписание `ClassSchedule`.

Необходимо из файла получить значения всех аудиторий, преподавателей, предметов, групп. Для этого был разработан алгоритм, показанный на блок схеме ниже.

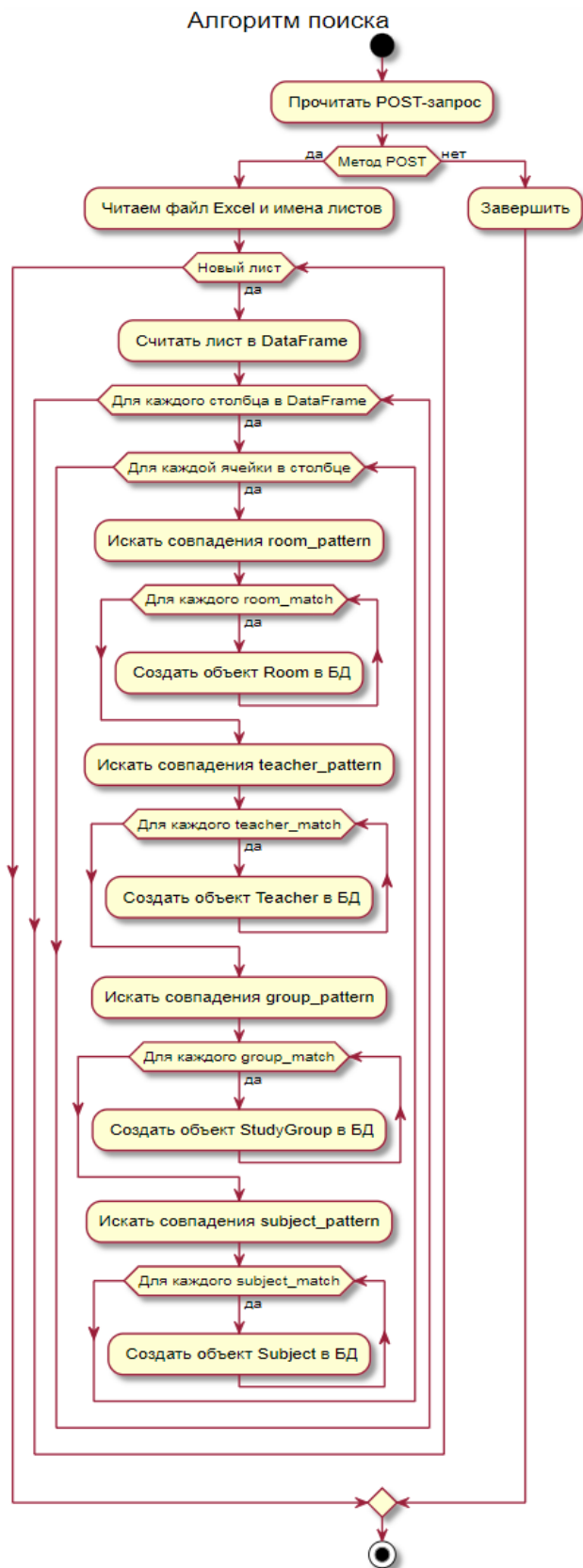


Рисунок 4.4 - алгоритм поиска

Листинг алгоритма в программе:

```
if self.request.method == 'POST':
    excel_file = self.request.FILES['excel_file']
    xls = pd.ExcelFile(excel_file)
    sheets = xls.sheet_names
    data_frame = pd.read_excel(excel_file)

    room_pattern = re.compile(r'b\d{4}\b\b\d\s\d{3}\b')
    teacher_pattern = re.compile(r'([А-Я][а-я]+(?:-[А-Я][а-я]+)?\s+[А-Я]\.[А-Я]\.)', re.UNICODE)
    group_pattern = re.compile(r'\d{2}s*[А-ЯЁа-яё]\s[-\d+]*\(\d+\)', re.UNICODE)
    subject_pattern = re.compile(
        r'([А-Я][А-Яа-я\s-]*?)(?=\s*\d+,|nr\.|лаб\.|лаб|лекция|спорткомплекс|проф\.|доц\.|(англ.)',
        re.UNICODE)

    for i in range(1, 19):
        Week.objects.get_or_create(week_number=i)

    for day_choice, _ in ClassPeriod.DAY_CHOICES:
        for para_choice, _ in ClassPeriod.PARA_CHOICES:
            ClassPeriod.objects.get_or_create(day=day_choice, para=para_choice)

    for sheet in sheets:
        data_frame = pd.read_excel(xls, sheet)
        for column in data_frame.columns:
            for item in data_frame[column]:
                item = str(item)
                room_matches = room_pattern.findall(item)
                teacher_matches = teacher_pattern.findall(item)
                group_matches = group_pattern.findall(item)
                subject_matches = subject_pattern.findall(item)

                for match in room_matches:
                    match = match.replace(' ', '')
                    building_number = match[0]
                    floor_number = match[1]
                    room, created = Rooms.objects.get_or_create(room_id=match,
                                                                defaults={'building_number': building_number,
                                                                 'floor_number': floor_number})

                for match in teacher_matches:
                    match = match.strip()
                    teacher, created = Teacher.objects.get_or_create(name=match)

                for match in group_matches:
                    match = match.strip()
```

```

group, created = StudyGroup.objects.get_or_create(group_number=match)

for match in subject_matches:
    match = match.strip()
    if match:
        subject, created = Subject.objects.get_or_create(name=match)

```

Также в этот алгоритм включено заполнение статических моделей Week и ClassPeriod базовыми значениями.

Описание шагов алгоритма:

1. Проверяется, является ли HTTP метод запроса POST. Если да, то алгоритм продолжается, иначе он заканчивается.
2. Из запроса извлекается файл 'excel_file'.
3. С помощью библиотеки pandas файл читается, и из него извлекаются имена всех листов.
4. Из всего файла создается pandas DataFrame.
5. Создаются шаблоны регулярных выражений для поиска информации о комнате, преподавателе, группе и предмете.
6. В базе данных создаются объекты для каждой недели от 1 до 18.
7. Создание объектов ClassPeriod: В базе данных создаются объекты для каждого возможного периода занятий.
8. Чтение каждого листа: Затем для каждого листа в Excel файле выполняются следующие действия:
 - Лист считывается в новый DataFrame.
 - Для каждого столбца в DataFrame:
 - Для каждого элемента в столбце:
 - Производится поиск совпадений с помощью регулярных выражений.
 - Для каждого совпадения:
 - Создается объект в базе данных, если он еще не существует.

Этот алгоритм проходит через каждую ячейку в каждом листе Excel файла, ищет совпадения с шаблонами регулярных выражений и создает объекты в базе данных на основе найденных совпадений. Это позволяет автоматизировать обработку и сохранение больших объемов данных.

Стоит обратить внимание на регулярные выражения разработанные для поиска искомых значений

3.2.2 Алгоритм заполнения расписания

После того, как файл был прочитан и вспомогательные модели были заполнены необходимо приступить к поиску расписания занятий. Необходимо сопоставить все найденные значения, для этого используется алгоритм, показанный ниже:

Алгоритм заполнения расписания

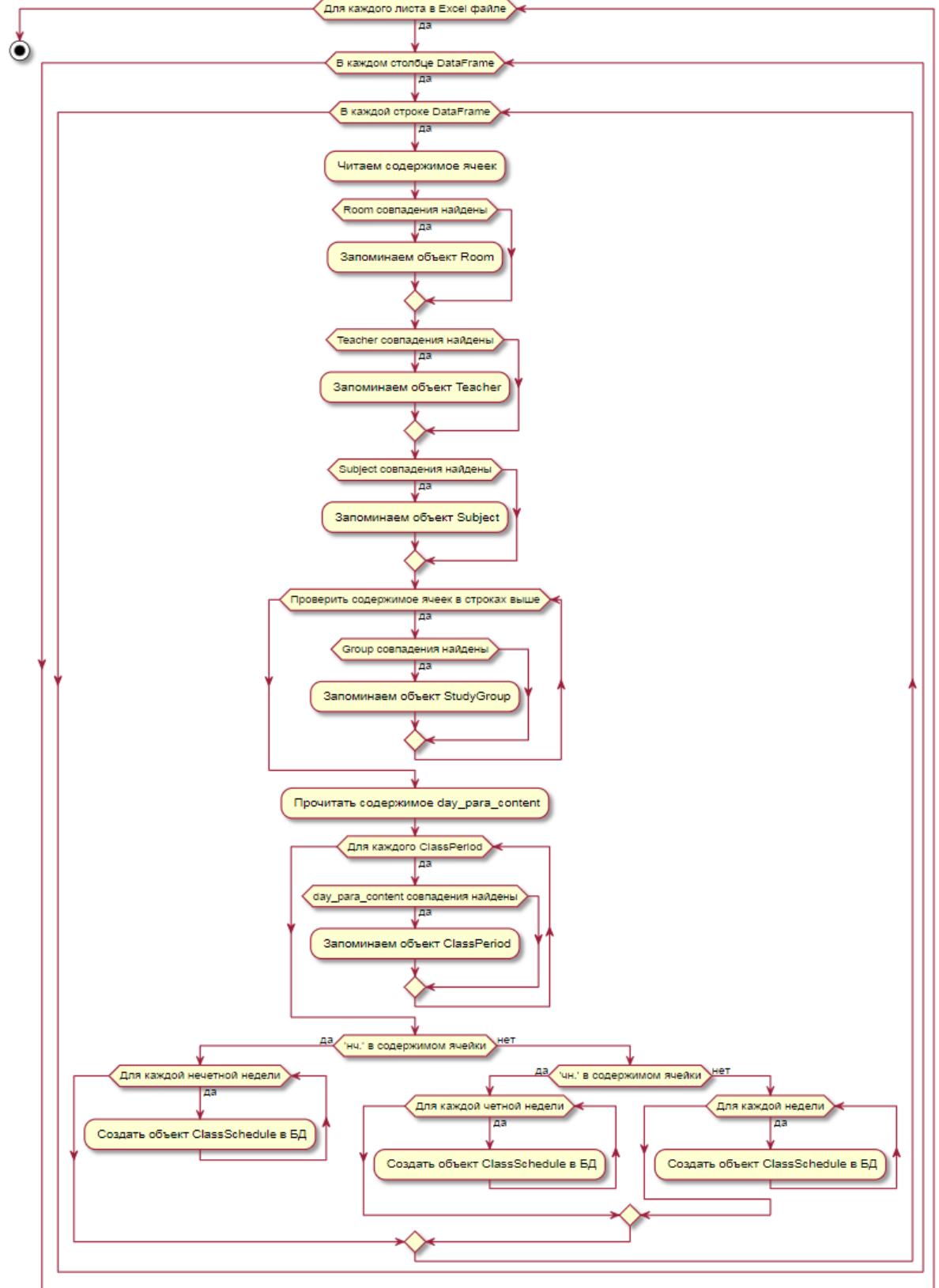


Рисунок ***- алгоритм заполнения расписания

Листинг алгоритма заполнения расписания:

```
for sheet in sheets:
    data_frame = pd.read_excel(xls, sheet)
    for column in range(data_frame.shape[1]):
        for row in range(data_frame.shape[0]):
            cell_content = str(data_frame.iloc[row, column])
            room_matches = room_pattern.findall(cell_content)

            if room_matches:
                room_match = room_matches[0].replace(' ', '')
                building_number = room_match[0]
                floor_number = room_match[1]
                room, created = Rooms.objects.get_or_create(room_id=room_match,
                                                            defaults={'building_number': building_number,
                                                                      'floor_number': floor_number})

                teacher = None
                subject = None
                group = None
                class_period = None

                teacher_matches = teacher_pattern.findall(cell_content)
                if teacher_matches:
                    teacher_match = teacher_matches[0].strip()
                    teacher, created = Teacher.objects.get_or_create(name=teacher_match)

                subject_matches = subject_pattern.findall(cell_content)
                if subject_matches:
                    subject_match = subject_matches[0].strip()
                    if subject_match:
                        subject, created = Subject.objects.get_or_create(name=subject_match)

                for above_row in range(row - 1, -1, -1):
                    above_cell_content = str(data_frame.iloc[above_row, column])
                    group_matches = group_pattern.findall(above_cell_content)
                    if group_matches:
                        group_match = group_matches[0].strip()
                        group, created = StudyGroup.objects.get_or_create(group_number=group_match)
                        break

                day_para_content = str(data_frame.iloc[row, 0]) + ' ' + str(
                    data_frame.iloc[row, 1])
                for day_choice, _ in ClassPeriod.DAY_CHOICES:
                    for para_choice, _ in ClassPeriod.PARA_CHOICES:
                        if day_choice in day_para_content and para_choice in day_para_content:
                            class_period, created = ClassPeriod.objects.get_or_create(day=day_choice,
                                                                                       para=para_choice)
```

```

        break

    if 'нч.' in cell_content:
        odd_weeks = [week for week in weeks if week.week_number % 2 != 0]
        for week in odd_weeks:
            ClassSchedule.objects.get_or_create(
                room=room,
                teacher=teacher,
                subject=subject,
                study_group=group,
                class_period=class_period,
                week=week,
            )
    elif 'чн.' in cell_content:
        even_weeks = [week for week in weeks if week.week_number % 2 == 0]
        for week in even_weeks:
            ClassSchedule.objects.get_or_create(
                room=room,
                teacher=teacher,
                subject=subject,
                study_group=group,
                class_period=class_period,
                week=week,
            )
    else:
        for week in weeks:
            ClassSchedule.objects.get_or_create(
                room=room,
                teacher=teacher,
                subject=subject,
                study_group=group,
                class_period=class_period,
                week=week,
            )

```

Опишем этапы алгоритма:

1. С помощью библиотеки pandas каждый лист считывается в DataFrame.
2. В цикле обрабатываются все столбцы DataFrame.
3. В каждом столбце просматриваются все строки.
4. Содержимое каждой ячейки конвертируется в строку, и затем ищутся совпадения с заданными регулярными выражениями для комнаты, преподавателя и предмета. Если совпадения найдены, создаются соответствующие объекты в базе данных.

5. Для каждой ячейки алгоритм также ищет информацию о группе, проверяя ячейки в том же столбце выше текущей ячейки. Если найдено совпадение, создается объект группы.

6. В каждой строке алгоритм также проверяет первые две ячейки, чтобы определить день и период занятий. Если найдено совпадение с возможными значениями, создается объект периода занятий.

7. После того, как все необходимые объекты были созданы или найдены, алгоритм определяет, к каким неделям относится ячейка (нечетные, четные или все), и создает объекты расписания занятий в базе данных для каждой соответствующей недели.

3.2.3 Алгоритм поиска занятых и свободных аудиторий

В ходе выполнения предыдущих алгоритмов по результатам их работы были получены данные по преподавателям, учебным занятиям, аудиториям, группам, времени занятия. Далее эти данные были преобразованы алгоритмом поиска расписания в расписание учебных занятий по аудиториям. Следующий этап подразумевает создание алгоритма поиска занятых и свободных аудиторий аудиторий из построенного расписания в базе данных. Алгоритм предоставлен на диаграмме ниже.

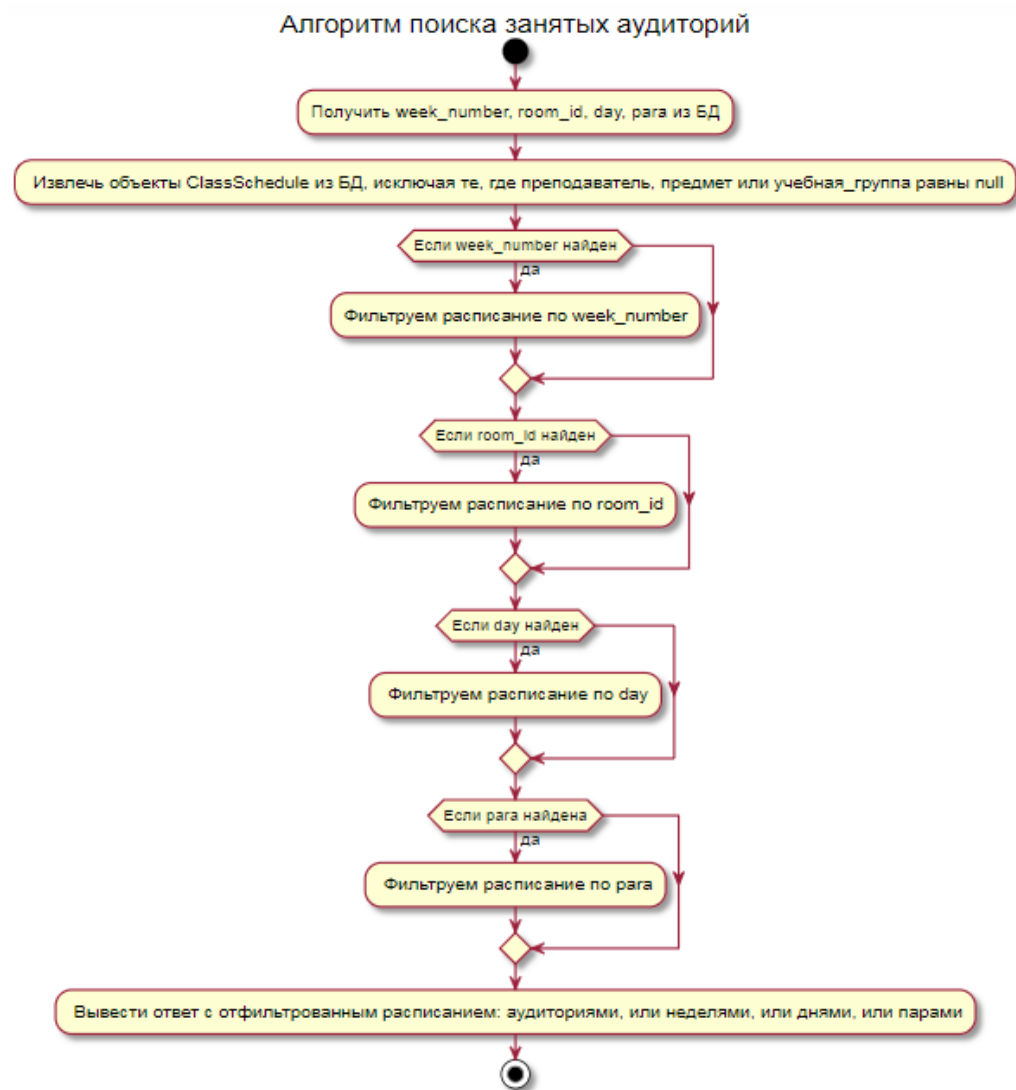


Рисунок 4.4 - алгоритм поиска занятых аудиторий

Листинг алгоритма поиска занятых аудиторий:

```

class BusyRoomsView(LoginRequiredMixin, View):
    template_name = 'main/busy_rooms.html'

    def get(self, request):
        week_number = request.GET.get('week_number', "")
        weeks = Week.objects.all()
        room_id = request.GET.get('room_id', "")
        rooms = Rooms.objects.all().order_by('room_id')
        day = request.GET.get('day', "")
        para = request.GET.get('para', "")

        schedule = ClassSchedule.objects.exclude(teacher__isnull=True, subject__isnull=True,
study_group__isnull=True)
  
```

```

if week_number:
    schedule = schedule.filter(week__week_number=week_number)
if room_id:
    schedule = schedule.filter(room__room_id=room_id)
if day:
    schedule = schedule.filter(class_period__day=day)
if para:
    schedule = schedule.filter(class_period__para=para)

schedule = schedule[:100]

return render(request, self.template_name, {'schedule': schedule, 'rooms': rooms, 'weeks': weeks,
                                             'days': ClassPeriod.DAY_CHOICES, 'paras':
ClassPeriod.PARA_CHOICES})

```

Данный алгоритм обрабатывает GET-запрос на веб-страницу, используя информацию, переданную в запросе, для фильтрации расписания занятий:

1. Алгоритм начинается с получения параметров week_number, room_id, day, para из GET-запроса.
2. Алгоритм получает все объекты "Week" и "Rooms" из базы данных. Затем он получает все объекты "ClassSchedule", исключая те, у которых teacher, subject или study_group равны null.
3. Если параметры week_number, room_id, day, para были предоставлены в запросе, алгоритм фильтрует расписание, используя эти параметры. То есть, он оставляет только те объекты "ClassSchedule", которые соответствуют предоставленным параметрам.
4. Затем алгоритм ограничивает размер расписания до первых 100 объектов, чтобы предотвратить передачу слишком большого объема данных.
5. Наконец, алгоритм создает ответ, в котором рендерит определенный шаблон с контекстом, который включает отфильтрованное расписание по всем комнатам, всем неделям, дням и парам. Этот ответ затем отображается пользователю на html странице.

Алгоритм поиска свободных аудиторий отличается только тем, что расписание ClassSchedule фильтруется не по присутствующим в расписании преподавателе, занятии и группе, а наоборот по отсутствующим.

```
schedule = ClassSchedule.objects.filter(
    Q(teacher__isnull=True) | Q(teacher__exact='') |
    Q(subject__isnull=True) | Q(subject__exact='') |
    Q(study_group__isnull=True) | Q(study_group__exact='')
)
```

3.2.4 Алгоритм авторизации

Для обеспечения безопасности данных информационной системы обязательно должны быть применены механизмы защиты информации. Один из таких механизмов – механизм авторизации. Авторизация предотвращает возможность пользователям иметь доступ к данным, к которым разрешения не были предоставлены. В веб-приложении был разработан алгоритм авторизации, который показан на схеме ниже:



Рисунок 3.2.4 - алгоритм авторизации

Листинг алгоритма авторизации:

```
def loginPage(request):
    if request.method == 'POST':
        username = request.POST.get('login')
        password = request.POST.get('password')
```

```

try:
    user = User.objects.get(username=username)
except:
    messages.error(request, 'Логин или пароль не совпадают')

user = authenticate(request, username=username, password=password)

if user is not None:
    login(request, user)
    return redirect('start-page')
else:
    messages.error(request, 'Логин или пароль не совпадают')

context = {}
return render(request, 'main/login.html', context)

```

Описание алгоритма авторизации:

1. В начале процесса авторизации пользователь вводит учетные данные, а именно логин и пароль. Данные переданы через форму авторизации на веб-странице.
2. Затем система проверяет, существует ли пользователь с введенным логином в базе данных.
3. Если пользователь найден, то система продолжает процесс авторизации, иначе переходит к шагу 7.
4. Django authenticate - это функция, которая используется для проверки введенного пользователем пароля. Эта функция принимает в качестве аргументов имя пользователя и пароль, и возвращает объект пользователя, если указанные учетные данные верны, иначе возвращает None.
5. Если функция authenticate вернула объект пользователя, это означает, что пароль верный, и система авторизует пользователя. Если функция authenticate вернула None, это означает, что пароль неверный.
6. Если пароль верный, пользователь авторизуется в системе.
7. Если пароль неверный или пользователь с таким логином не найден, система отправляет сообщение об ошибке "Логин или пароль не совпадают".

3.4 Создание интерфейса и структуры веб-приложения

Информация о структуре и функционировании основных файлов разработанного приложения, а также с расположением директорий и файлов, подробно изложена в приложении. Маршрутизация веб-запросов осуществляется через файл `main/urls.py`.

```
from django.urls import path, include
from . import views
from .views import FreeRoomsView, BusyRoomsView, AddRoomView, IndexView, UploadFileView

urlpatterns = [
    path('login/', views.loginPage, name='login'),
    path("", IndexView.as_view(), name='start-page'),
    path('about-us/', views.about, name='about'),
    path('upload/', UploadFileView.as_view(), name='upload'),
    path('add/', AddRoomView.as_view(), name='add'),
    path('busy-rooms/', BusyRoomsView.as_view(), name='busy-rooms'),
    path('free-rooms/', FreeRoomsView.as_view(), name='free-rooms'),
    path('free-rooms/<int:pk>/', FreeRoomsView.as_view(), name='free-rooms-detail'),
]
```

Этот файл содержит список URL-маршрутов, каждый из которых обслуживается отдельным классом обработки запросов, возвращающим данные клиенту. В файле `main/views.py` хранятся классы, которые представляют эти обработчики. Здесь стоит отметить, что, хотя можно использовать функции, но использование классов обеспечивает большую гибкость, позволяет применять наследование.

Django предоставляет большой набор классов, упрощающих работу с данными и базами данных. Вот несколько из них, которые можно использовать при создании собственных представлений: `View`, `TemplateView`, `CreateView`, `FormView`, `ListView`, `UpdateView`, `DeleteView`. Специальные классы, называемые моделями (`models.Model`), служат для описания сущностей в уже созданной базе данных. Эти модели располагаются в директории `main/models.py`. Особенность классов представления Django в том, что они уже имеют встроенные методы для работы с

базой данных. Поэтому, при их использовании, достаточно указать только имя класса модели, с которым должен взаимодействовать класс представления.

Рассмотрим пример: если нам требуется отобразить список занятых аудиторий.

```
class BusyRoomsView(LoginRequiredMixin, View):  
    template_name = 'main/busy_rooms.html'
```

В этом классе применяется Django-класс View, в котором уже реализован метод для вывода списка объектов из базы данных, а также имя шаблона (`template_name = 'main/busy_rooms.html'`), который будет использоваться для отображения этих данных. LoginRequiredMixin является классом-примесью (mixin) в Django, который используется для добавления функционала требования аутентификации к классам представлений. Если пользователь не вошел в систему (не авторизован), он будет перенаправлен на страницу входа ('login/'), и после успешной авторизации вернется обратно на первоначальную страницу. Данный класс-примесь добавлен во все классы, для того, чтобы ограничить доступ к информации.

Для удобства предоставления информации с одинаковым интерфейсом используется шаблон, от которого наследуется шапка и общая структура страницы на всех остальных шаблонах. Этот шаблон находится по адресу: `templates/main/base.html`.

Указание `{% extends 'main/base.html' %}` в шаблоне говорит о том, что он наследуется от базового шаблона. Это означает, что верстка из базового шаблона будет отображаться в текущем шаблоне, как написано выше.

Кроме того, в шаблонах применяется модуль для работы с bootstrap, который можно активировать следующим образом:

```
<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/css/bootstrap.min.css"  
integrity="sha384-
```

```
rbsA2VBKQhggwzxH7pPCaAqO46MgnOM80zW1RWuH61DGLwZJEdK2Kadq2F9CUG65"
crossorigin="anonymous">
```

Рассмотрим шаблон 'main/base.html' подробнее для того, чтобы показать, как используется Bootstrap:

```
<!doctype html>
<html lang="ru">
<head>
  <meta charset="UTF-8">
  <meta name="viewport"
    content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-scale=1.0,
    minimum-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <title>{% block title %}{% endblock %}</title>
  <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/css/bootstrap.min.css" integrity="sha384-
rbsA2VBKQhggwzxH7pPCaAqO46MgnOM80zW1RWuH61DGLwZJEdK2Kadq2F9CUG65"
crossorigin="anonymous">
</head>
<body>
<div class="d-flex flex-column flex-md-row align-items-center pb-3 mb-4 border-bottom"
style="background-color: #f8f9fa;">
  <a href="/" class="d-flex align-items-center text-decoration-none" style="color: #007bff; margin-
right: auto;">
    <h5 class="fs-4">Портал Аудитории</h5>
  </a>

  <nav class="d-inline-flex mt-2 mt-md-0">
    <a class="me-3 py-2 text-decoration-none" href="{% url 'start-page' %}" style="color: #6c757d;
margin: 10px;">Главная</a>
    <a class="me-3 py-2 text-decoration-none" href="{% url 'about' %}" style="color: #6c757d;
margin: 10px;">О нас</a>
    <a class="me-3 py-2 text-decoration-none" href="{% url 'upload' %}" style="color: #6c757d;
margin: 10px;">Загрузка расписания</a>
    <a class="me-3 py-2 text-decoration-none" href="{% url 'add' %}" style="color: #6c757d;
margin: 10px;">Добавить</a>
    <a class="me-3 py-2 text-decoration-none" href="{% url 'busy-rooms' %}" style="color:
#6c757d; margin: 10px;">Занятые аудитории</a>
    <a class="me-3 py-2 text-decoration-none" href="{% url 'free-rooms' %}" style="color:
#6c757d; margin: 10px;">Свободные аудитории</a>
  </nav>
</div>
<div class="container" style="margin-top: 20px;">
  {% block content %}{% endblock %}
```


`</div>`

`</body>`

`</html>`

d-flex flex-column flex-md-row align-items-center pb-3 mb-4 border-bottom - это набор классов Bootstrap, которые добавляют различные стили элементам HTML. Вот что они делают:

- **d-flex**: превращает элемент в гибкий контейнер, что позволяет использовать другие свойства flexbox для управления расположением дочерних элементов;

- **flex-column**: устанавливает основное направление гибкого контейнера как колонку. Дочерние элементы будут стекаться вертикально;

- **flex-md-row**: Начиная с размера экрана "md" (среднего) и выше, направление гибкого контейнера изменяется на строку. Дочерние элементы будут выстраиваться горизонтально;

- **align-items-center**: Выравнивает дочерние элементы по вертикали по центру в гибком контейнере;

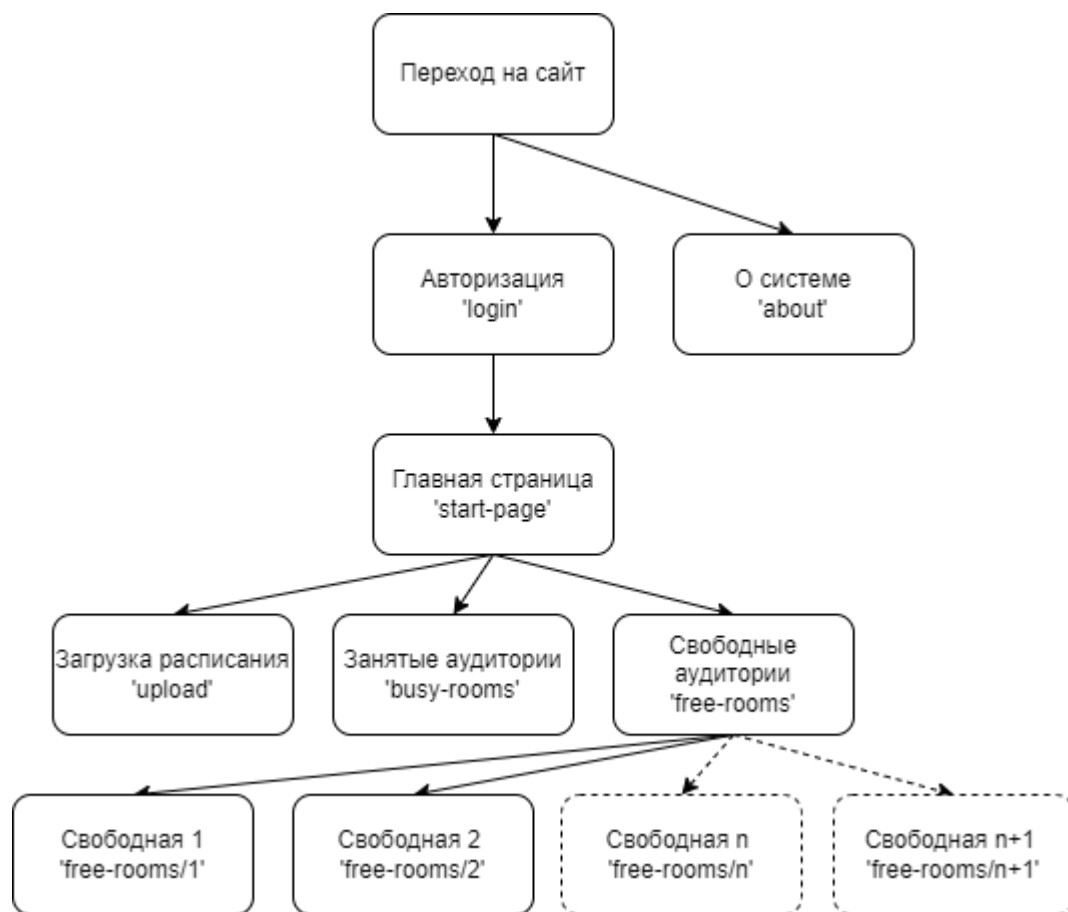
- **pb-3**: Добавляет отступ снизу (padding-bottom) в размере 3 единицы;

- **mb-4**: Добавляет внешний отступ снизу (margin-bottom) в размере 4 единицы;

- **border-bottom**: Добавляет границу снизу к элементу;

Этот набор классов представляет собой очень эффективный способ быстро создавать сложные макеты без необходимости писать собственный CSS-код.

Сама структура шаблонов веб-приложения при переходе между страницами выглядит следующим образом:



Перейдя на главную страницу или на любую другую страницу, кроме «about» предлагается авторизоваться в системе для доступа к данным.

Портал Аудиторий
[Главная](#)
[О нас](#)
[Загрузка расписания](#)
[Добавить](#)
[Занятые аудитории](#)
[Свободные аудитории](#)

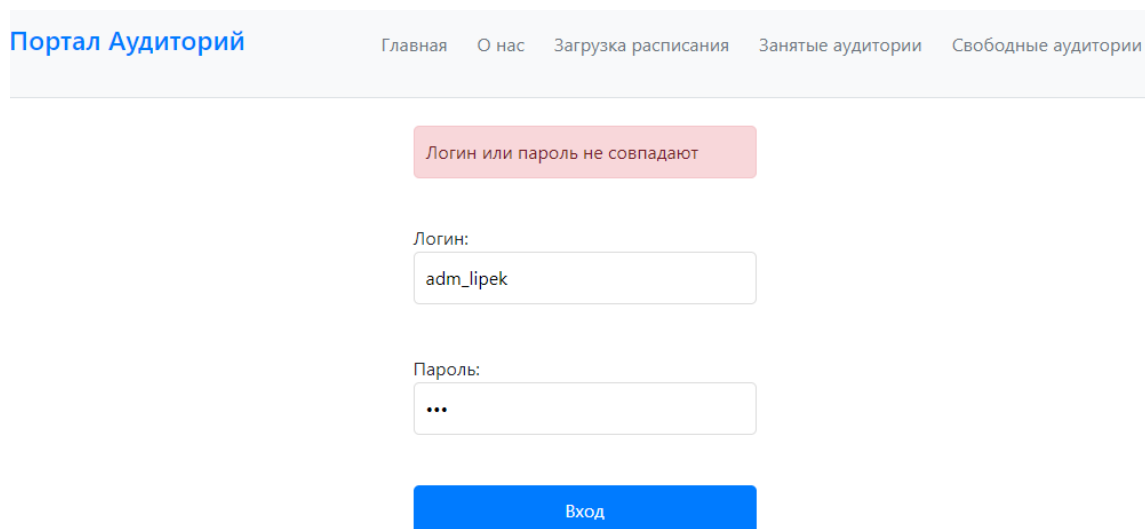
Логин:

Пароль:

Вход

Рисунок  - авторизация

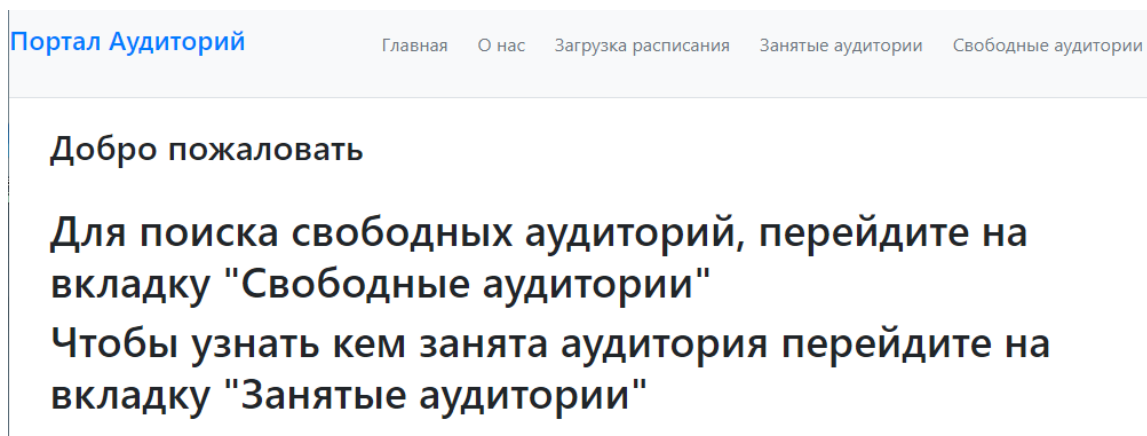
Чтобы авторизоваться необходимо ввести логин и пароль, предоставленный администратором. Если пользователь ведет некорректный логин или пароль появится информационное сообщение «Логин или пароль не совпадают»



The screenshot shows the login interface of the 'Портал Аудиторий' (Auditorium Portal). At the top, there is a navigation bar with the portal's name and links: Главная, О нас, Загрузка расписания, Занятые аудитории, and Свободные аудитории. Below the navigation bar, a red error message box displays the text 'Логин или пароль не совпадают'. Underneath this, there are two input fields: 'Логин:' with the text 'adm_lipek' and 'Пароль:' with masked characters '***'. At the bottom of the form is a blue button labeled 'Вход'.

Рисунок ** - ошибка авторизации

После успешного входа в систему откроется главная страница «start-page», которая выглядит следующим образом:




The screenshot shows the main page of the 'Портал Аудиторий'. It features the same navigation bar at the top. The main content area has a heading 'Добро пожаловать' followed by two lines of text: 'Для поиска свободных аудиторий, перейдите на вкладку "Свободные аудитории"' and 'Чтобы узнать кем занята аудитория перейдите на вкладку "Занятые аудитории"'. The text is presented in a clean, sans-serif font.

Рисунок ** - главная страница

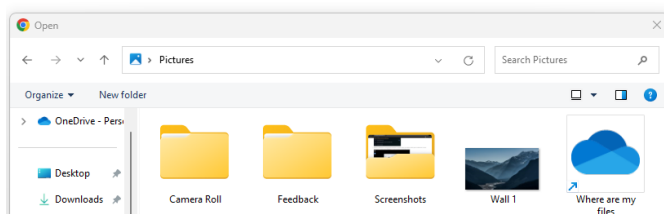
В шапке отображаются ссылки для перехода между страницами, для заполнения расписания можно перейти по ссылке в шапке «Загрузка расписания», перейдя по которой откроется страница загрузки расписания (см. Рисунок **). На этой странице можно нажать на кнопку Choose file для загрузки файла расписания с компьютера, на котором был открыт сайт (см. Рисунок **) и можно Загрузить.

Загрузка Excel файла

 No file chosen


Рисунок  - загрузка расписания

Загрузка Excel файла

 No file chosen

Загрузка Excel файла

 kurs_1.xlsx

Рисунок  - загрузка расписания из файла

После того, как расписание загрузится и обработается пользователь перейдет на главную страницу, с которой можно перейти по ссылке «Занятые аудитории» и посмотреть список занятых аудиторий.

Список занятых аудиторий

Аудитория
-- select room --

Неделя
-- select week --

День недели
-- select day --

Пара
-- select para --

Фильтр

Аудитория: 6339
Группа: 22 РЭС (36)
Преподаватель: Приблудова Е.Н.
Предмет: Информационные технологии
: 1
Вторник
1 пара

Аудитория: 6339
Группа: 22 РЭС (36)
Преподаватель: Приблудова Е.Н.
Предмет: Информационные технологии
: 2
Вторник
1 пара

Рисунок ** - список занятых аудиторий

Для каждой аудитории указаны: группа, преподаватель, предмет, номер недели, день недели, номер пары. Также на этой странице есть возможность отфильтровать результаты поиска по аудитории, неделе, дню недели, номеру пары для более детального поиска. Фильтры можно применять, как все вместе, так и по отдельности. Например, можно выбрать аудиторию 6555, 1 неделю, вторник, 3 пару.

Аудитория
6555

Неделя
1

День недели
Вторник

Пара
3 пара

Фильтр

Рисунок ** - фильтр

Отобразится результат:

Список занятых аудиторий

Аудитория: -- select room --
Неделя: -- select week --
День недели: -- select day --
Пара: -- select para --
[Фильтр](#)

Аудитория: 6555

Группа: 22 РЭС (36)

Преподаватель: Новоселова Н.А.

Предмет: Физика

: 1

Вторник

3 пара

Рисунок ** - примененный фильтр

По такой же логике работает страница «Свободные аудитории», которая выглядит так:

[Портал Аудиторий](#)

[Главная](#)

[О нас](#)

[Загрузка расписания](#)

[Занятые аудитории](#)

[Свободные аудитории](#)

Список свободных аудиторий

Аудитория: -- select room --
Неделя: -- select week --
День недели: -- select day --
Пара: -- select para --
[Фильтр](#)

[Аудитория: 6429](#)

: 2

Вторник

4 пара

[Аудитория: 6429](#)

: 4

Вторник

4 пара

Рисунок ** - свободные аудитории

Разница в том, что, перейдя по ссылке в номере аудитории откроется форма бронирования аудитории:

Аудитория: 3309

Преподаватель

Предмет

Группа

Рисунок 3.5 - бронирование аудитории

3.5 Тестирование

Заключение

В процессе выполнения данного проекта были проанализированы различные платформы по управлению учебными процессами, были исследованы современные языки программирования веб-разработки, основные преимущества и недостатки этих языков. Также были рассмотрены актуальные Python фреймворки: Django и Flask. После этого, было разработано веб-приложение для анализа и представления данных об использовании учебных аудиторий и спроектировали базу данных этого приложения. В выборе технологий было решено остановиться на стеке технологий Python, Django и SQLite. В итоге, было успешно реализованно веб-приложение.

Целью проекта было проектирование и разработка веб-приложения, которое поможет с анализом и представлением данных об использовании учебных аудиторий, включающее предоставление информации об использовании учебных аудиторий с возможностью бронирования свободных учебных аудиторий. Поставленные цели были успешно достигнуты.

Для этого были выполнены следующие задачи:

- Проанализированы существующие программные продукты для управления учебными процессами;
- Изучены современные языки веб-разработки, был выбран подходящий язык программирования и фреймворк;
- Спроектирована база данных с критериями данных по результатам парсинга учебного расписания;
- Реализованы алгоритмы: поиска данных в файле электронных таблиц, наполнения расписания в базе данных, авторизации, поиска занятых и свободных аудиторий;
- Разработано веб-приложение;

Благодаря выбранному стеку технологий, возможно дальнейшее развитие и расширение функционала веб-приложения. На текущий момент, проект считается завершенным, так как он полностью соответствует всем поставленным задачам.

Список литературы

