Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

«МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ» (МОСКОВСКИЙ ПОЛИТЕХ)

КУРСОВОЙ ПРОЕКТ

По курсу «Разработка веб-приложений»

ТЕМА

«Разработка веб-приложения для логопедического центра с системой абонементов и личными кабинетами пользователей»

Выполнил Самохвалов Вячеслав Дмитриевич

Группа 231-3210

Проверил Кружалов А.С.

  Москва, 2025

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

**(МОСКОВСКИЙ ПОЛИТЕХ)**

|  |  |
| --- | --- |
|  | УТВЕРЖДАЮ  заведующая кафедрой  «Инфокогнитивные технологии»  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / Е. А. Пухова /  «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_ 2025 г. |

**ЗАДАНИЕ**

**на выполнение курсовой работы (проекта)**

Самохвалову Вячеславу Дмитриевичу,

(ФИО обучающегося)

обучающемуся группы 231-3210,

направления подготовки 09.03.01 «Информатика и вычислительная техника»

по дисциплине «Разработка веб-приложений»

на тему «Разработка веб-приложения для логопедического центра с системой абонементов и личными кабинетами пользователей»

1. Исходные данные к работе (проекту): информационные ресурсы в сети интернет, научные публикации в открытой печати.

2. Содержание задания по курсовой работе (проекту) – перечень вопросов, подлежащих разработке:

|  |  |  |  |
| --- | --- | --- | --- |
| **Разрабатываемый вопрос** | **Объем от всего задания, %** | **Срок выполнения** | **Примечание** |
| Раздел 1. Анализ предметной области |  |  |  |
| Задача 1.1. Обзор существующих программных продуктов по теме работы | 10 | 22.03.2025 |  |
| Задача 1.2. Анализ программных инструментов разработки веб-приложений | 7 | 26.03.2025 |  |
| Задача 1.3. Формулировка цели и задач работы | 8 | 30.03.2025 |  |
| Раздел 2. Проектирование веб-приложения |  |  |  |
| Задача 2.1. Анализ целевой аудитории | 5 | 02.04.2025 |  |
| Задача 2.2. Описание функциональности приложения (диаграмма вариантов использования, user story и т. д.) | 7 | 06.04.2025 |  |
| Задача 2.3. Проектирование модели данных (ER-диаграмма, логическая и физическая схемы БД) | 8 | 10.04.2025 |  |
| Задача 2.4. Разработка макетов страниц (Wireframe) | 5 | 14.04.2025 |  |
| Раздел 3. Разработка веб-приложения |  |  |  |
| Задача 3.1. Разработка базовой структуры приложения и вёрстка шаблонов страниц | 8 | 21.04.2025 |  |
| Задача 3.2. Реализация аутентификации пользователей | 7 | 27.04.2025 |  |
| Задача 3.3. Реализация CRUD-интерфейса для пользователей, абонементов и домашних заданий | 7 | 03.05.2025 |  |
| Задача 3.4. Реализация системы управления абонементами | 12 | 09.05.2025 |  |
| Задача 3.5. Разработка системы домашних заданий | 6 | 15.05.2025 |  |
| Раздел 4. Оформление итогов работы |  |  |  |
| Задача 4.1. Создание Git-репозитория с кодом проекта | 3 | 22.03.2025 |  |
| Задача 4.2. Деплой приложения на хостинг | 4 | 26.05.2025 |  |
| Задача 4.3. Оформление отчёта о проделанной работе | 3 | 26.05.2025 |  |

Руководитель курсовой работы (проекта): ст. преподаватель кафедры «Инфокогнитивные технологии»

|  |  |  |  |
| --- | --- | --- | --- |
| «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_ 2025 г. | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | А. С. Кружалов | |
| Дата выдачи задания | | | «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_ 2025 г. |
| Дата сдачи выполненной работы (проекта) | | | «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_ 2025 г. |

Задание принял к исполнению

|  |  |  |
| --- | --- | --- |
| «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_ 2025 г. | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (И. О. Фамилия) |

СОДЕРЖАНИЕ

[СОДЕРЖАНИЕ 3](#_Toc194094533)

[ВВЕДЕНИЕ 4](#_Toc194094534)

[1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ 5](#_Toc194094535)

[1.1 Обзор существующих программных продуктов 5](#_Toc194094536)

[1.2 Анализ программных инструментов разработки веб-приложений 7](#_Toc194094537)

[1.2 Формулировка цели и задач работы 9](#_Toc194094538)

[ЗАКЛЮЧЕНИЕ 11](#_Toc194094539)

[СПИСОК ЛИТЕРАТУРЫ 12](#_Toc194094540)

# ВВЕДЕНИЕ

Современные информационные технологии предоставляют широкие возможности для автоматизации процессов в сфере образования и коррекционной педагогики. Особенно актуальным становится внедрение информационных систем в логопедическую практику, где важно учитывать индивидуальные особенности учеников, гибкое расписание занятий и абонементную форму обучения.

Данная работа посвящена разработке информационной системы для логопедического центра. Система ориентирована на три роли пользователей: ученик, преподаватель и администратор. Она предусматривает прикрепление учеников к преподавателям по специальному коду либо через администратора, оформление абонементов на занятия, ведение индивидуального расписания, учет отмен с учетом 24-часового правила, архивирование завершенных абонементов и отслеживание активности по каждому из них.

Отдельное внимание уделяется функциональности управления расписанием и системе домашних заданий. Преподаватель может прикреплять материалы к занятию, а ученик — загружать выполненные задания. Администратор при этом имеет доступ к общему расписанию центра и может контролировать организацию учебного процесса в целом.

В рамках работы были спроектированы логическая и физическая модели базы данных, разработан макет интерфейса и продумана структура системы с учетом реальных требований логопедического центра. Реализация проекта позволит оптимизировать взаимодействие между всеми участниками образовательного процесса, повысить эффективность ведения занятий и улучшить контроль за выполнением заданий.

# 1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

## Обзор существующих программных продуктов

В настоящее время на рынке представлено множество программных решений для организации образовательного процесса и управления учебными центрами. Однако существующие продукты не в полной мере удовлетворяют требованиям логопедического центра, поскольку не предоставляют удобных инструментов для работы с абонементами и интеграции логопедических заданий.

Одной из популярных CRM-систем для учебных центров является Битрикс24. Данная система предоставляет функционал для управления клиентами, ведения расписания и коммуникации с пользователями. Однако в ней отсутствует встроенная поддержка абонементной системы, что делает учет посещений менее удобным. Также в CRM-системах, ориентированных на образовательные учреждения, нет специализированных инструментов для работы с логопедическими заданиями и их проверки.

Существуют также LMS-системы (Learning Management System), предназначенные для организации образовательного процесса. Примеры таких систем – Moodle и Mirapolis. Они позволяют создавать онлайн-курсы, загружать учебные материалы и тестировать учеников. Однако эти платформы ориентированы именно на дистанционное обучение и не содержат функционала для управления расписанием занятий, организации системы абонементов и учета оплат.

Сравнение существующих программных продуктов, а именно CRM-систем и LMS-систем представлено в таблице 1.

Таблица 1 – Сравнение существующих программных продуктов

|  |  |  |  |
| --- | --- | --- | --- |
| **Функционал** | **CRM-системы (Битрикс24 и аналоги)** | **LMS-системы (Moodle, Mirapolis)** | **Требуемое веб-приложение** |
| Управление расписанием | Есть инструменты для планирования встреч и событий, но нет гибкой настройки расписания для учеников и преподавателей | Отсутствует. Можно загружать курсы и лекции, но нет системы календаря занятий | Полноценная система расписания с возможностью корректировки преподавателем |
| Привязка учеников к преподавателям | В CRM можно вручную назначать клиентов сотрудникам, но нет автоматизированной системы привязки учеников к преподавателям по коду или администратором | Отсутствует. LMS предполагает самостоятельное обучение, поэтому логика привязки учеников к преподавателям не предусмотрена | Ученики могут привязываться к преподавателям через код или администратора |
| Система абонементов | CRM-системы работают по модели разовых платежей или подписок, но нет встроенной системы отслеживания оставшихся занятий | Отсутствует. LMS ориентированы на доступ к курсам, а не на модель посещений по абонементу | Полноценный учет абонементов: количество оставшихся занятий, списание при посещении, сохранение при своевременной отмене |
| Домашние задания и логопедические материалы | В CRM можно прикреплять файлы к задачам, но нет системы проверки и отслеживания выполнения домашнего задания | LMS-системы позволяют загружать задания и проверять их, но они не адаптированы под логопедические занятия | Преподаватель может загружать материалы, а ученик — прикреплять выполненные задания, отслеживать статус проверки |
| Гибкость настройки под логопедический центр | CRM можно адаптировать, но требуется сложная настройка. Основной функционал ориентирован на бизнес-задачи | LMS ориентированы на курсы и тестирование, нет инструментов управления очными занятиями | Полная адаптация под специфику логопедического центра: учет посещаемости, контроль выполнения заданий, удобное расписание |

Таким образом, существующие решения не обеспечивают комплексного подхода к организации логопедического центра. Это подтверждает актуальность разработки специализированного веб-приложения, которое объединит удобный учет абонементов, управление расписанием занятий и систему домашних заданий.

## 1.2 Анализ программных инструментов разработки веб-приложений

Для разработки веб-приложения логопедического центра было рассмотрено несколько стеков технологий. В итоге был выбран стек Flask (REST API) + React.js (Frontend) + MySQL (база данных). Однако существуют и альтернативные решения, которые также широко применяются в веб-разработке.

Для разработки серверной части веб-приложения можно использовать различные фреймворки и платформы. Сравнение наиболее популярных фреймворков представлено в таблице 2.

Таблица 2 – Сравнение популярных серверных фреймворков

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Критерий** | **Flask (Python)** | **Django (Python)** | **Node.js (JavaScript)** | **Spring Boot (Java)** |
| Архитектура | Микрофреймворк, минималистичный | Монолитный фреймворк | Легковесный, событийно-ориентированный | Полноценный фреймворк |
| Гибкость | Позволяет выбирать нужные библиотеки | Встроенные решения (ORM) | Позволяет использовать любые модули | Предлагает мощные встроенные инструменты |
| Поддержка REST API | Отличная поддержка | Встроенная поддержка API | Отличная поддержка, нативная работа с JSON | Хорошая поддержка, но сложная настройка |

Flask был выбран за его гибкость, простоту и удобство интеграции с REST API. Django – хороший вариант, но он более строгий и ориентирован на монолитные приложения. Express (Node.js) подходит для высоконагруженных и асинхронных задач, но уступает Flask в удобстве работы с Python-экосистемой. Spring Boot – мощное решение, но требует больше ресурсов и сложнее в освоении.

Фронтенд-приложения могут быть реализованы с помощью различных фреймворков и библиотек, приведенных в таблице 3.

Таблица 3 – Сравнение наиболее популярных фронтенд-фреймворков

|  |  |  |  |
| --- | --- | --- | --- |
| **Критерий** | **React.js** | **Angular** | **Vue.js** |
| Архитектура | Компонентная библиотека | Полноценный фреймворк | Компонентный фреймворк |
| Производительность | Высокая за счет Virtual DOM | Производительность ниже из-за двустороннего связывания | Высокая, но уступает React |
| Гибкость | Позволяет выбрать любые библиотеки | Включает все необходимые модули | Баланс между гибкостью и функционалом |
| Кривая обучения | Средняя, нужно знать JSX и концепции состояния | Средняя, требует знания TypeScript | Легкая, понятный синтаксис |
| Экосистема | Огромное количество готовых решений | Встроенные модули для работы с формами и HTTP | Хорошо подходит для небольших проектов |

React.js был выбран за его гибкость, производительность и широкую экосистему. Angular – мощное решение, но сложнее в освоении и менее гибкое. Vue.js – отличный вариант для небольших проектов, но для масштабируемых решений уступает React.

Выбор системы управления базами данных (СУБД) – критически важный этап при разработке веб-приложения. От него зависят скорость работы системы, удобство хранения и обработки данных, поддержка транзакций и возможность масштабирования. Для логопедического центра необходимо хранить информацию о пользователях (учениках, преподавателях, администраторах), абонементах, занятиях, расписании и домашних заданиях. Сравниние популярных СУБД представлено в таблице 4.

Таблица 4 – Сравнение популярных СУБД

|  |  |  |  |
| --- | --- | --- | --- |
| **Критерий** | **MySQL** | **PostgreSQL** | **MongoDB** |
| Тип | Реляционная (SQL) | Реляционная (SQL) | Документо-ориентированная (NoSQL) |
| Производительность | Высокая, но уступает PostgreSQL в сложных запросах | Лучшая производительность для сложных транзакций | Отлично подходит для работы с JSON |
| Гибкость | Хорошая, поддержка сложных запросов | Поддержка JSON, расширенные возможности SQL | Гибкая схема данных, нет строгих связей |
| Масштабируемость | Хорошая, поддерживает кластеризацию | Высокая, поддерживает репликацию | Отличная, горизонтальное масштабирование |

MySQL был выбран за его стабильность, производительность и широкую поддержку. PostgreSQL – мощный вариант, но сложнее в настройке.

## Формулировка цели и задач работы

Цель работы: разработка веб-приложения для логопедического центра, включающего систему абонементов и личные кабинеты пользователей (учеников, преподавателей и администраторов). Приложение должно обеспечивать удобную организацию занятий, ведение расписания, управление абонементами и интеграцию логопедических заданий.

Задачи работы:

1. Обзор существующих CRM и LMS систем для учебных центров.
2. Анализ инструментов для разработки веб-приложений (фреймворки, базы данных, библиотеки).
3. Формулировка цели и задач работы.
4. Определение целевой аудитории приложения (ученики, преподаватели, администраторы).
5. Описание функциональности приложения (абонементы, расписания, домашние задания).
6. Проектирование модели данных (ER-диаграмма, схемы базы данных).
7. Разработка wireframe для страниц приложения.
8. Создание структуры приложения и верстка шаблонов страниц.
9. Реализация системы аутентификации пользователей.
10. Реализация CRUD-интерфейса для управления пользователями и абонементами.
11. Разработка системы управления абонементами (создание, отправка в архив, отмена занятий по абонементам).
12. Реализация системы домашних заданий (загрузка и выполнение заданий).
13. Создание Git-репозитория с кодом проекта.
14. Деплой приложения на хостинг.

# 2 ПРОЕКТИРОВАНИЕ ВЕБ-ПРИЛОЖЕНИЯ

# 2.1 Описание работы системы

Разрабатываемое веб-приложение представляет собой информационную систему для автоматизации процессов логопедического центра. Система ориентирована на упрощение взаимодействия между участниками образовательного процесса, управление абонементами, расписанием и домашними заданиями. В системе предусмотрено три типа пользователей: ученик, преподаватель и администратор, каждый из которых имеет определённый набор функциональных возможностей.

Ролевая структура:

1. Ученик (или родитель ученика) регистрируется в системе и может либо самостоятельно прикрепиться к преподавателю по специальному коду, либо быть добавлен в список преподавателя администратором.
2. Преподаватель управляет своими учениками, создаёт для них расписание занятий, прикрепляет домашние задания, отслеживает прогресс.
3. Администратор обладает расширенными полномочиями: он управляет пользователями, назначает преподавателей ученикам, контролирует глобальное расписание центра и следит за функционированием всей системы.

Ключевая особенность проекта – гибкое управление абонементами. Ученики приобретают абонементы на определённое количество занятий (не менее пяти), при этом согласовывают с преподавателем удобное время. Занятие считается использованным, если не было отменено заранее — за 24 часа до его начала. Это условие необходимо для соблюдения дисциплины и учёта времени преподавателя. В системе реализовано отображение:

* активных абонементов;
* количества оставшихся, использованных и "сгоревших" занятий;
* архива истёкших абонементов.

У каждого ученика и преподавателя формируется собственное расписание. Преподаватель может редактировать расписание индивидуально под каждого ученика. Администратор, в свою очередь, имеет доступ к общему расписанию центра и может видеть все занятия, включая информацию о времени, ученике и преподавателе. Это позволяет избежать пересечений, лучше планировать нагрузку и использовать помещения центра более эффективно.

Преподаватель может прикреплять к будущим занятиям домашние задания в виде логопедических или дидактических материалов. Ученик, в свою очередь, имеет возможность загрузить выполненное задание обратно в систему, обеспечивая обратную связь и повышая качество обучения.

Ключевые преимущества системы:

1. Централизованное управление расписанием и абонементами;
2. Чёткое распределение ролей и полномочий;
3. Прозрачный механизм переноса/отмены занятий;
4. Интеграция домашних заданий в процесс обучения;
5. Возможность масштабирования и подключения новых преподавателей и учеников.

Таким образом, проект направлен на повышение эффективности организации учебного процесса в логопедическом центре, улучшение коммуникации между участниками и автоматизацию ключевых процессов, таких как расписание, учет занятий и абонементов.

## 2.2 Анализ целевой аудитории

Целевую аудиторию веб-приложения логопедического центра можно разделить на три основные группы: ученики (или их родители), преподаватели и администраторы. У каждой группы — свои цели, задачи и сценарии использования системы. Информация, описывающая особенности каждой из групп, представлена в таблице 5.

Таблица 5 – Особенности каждой из категорий пользователей

|  |  |  |
| --- | --- | --- |
| Категории пользователей | Основные задачи | Особенности |
| Ученики / Родители | Просмотр расписания занятий, получение домашних заданий, загрузка выполненных заданий | Интерфейс должен быть простым, понятным, с акцентом на удобство и информативность. |
| Преподаватели | Управление расписанием, назначение и проверка домашних заданий, прикрепление учеников, просмотр информации об абонементах | Пользуются системой регулярно. Необходима удобная навигация по ученикам и занятиям. Поддержка загрузки и прикрепления файлов. |
| Администраторы | Создание и редактирование учетных записей, просмотр глобального расписания центра, прикрепление учеников к преподавателям, контроль абонементов и активности пользователей | Нужен доступ ко всей информации. Интерфейс более функциональный, с возможностями фильтрации и управления пользователями. |

При проектировании интерфейса и функционала приложения важно учитывать потребности всех категорий пользователей и обеспечивать удобный доступ к информации в соответствии с их ролью.

## 2.3 Построение диаграммы вариантов использования (Use Case Diagram)

Диаграмма вариантов использования отражает ключевые действия, доступные пользователям системы в зависимости от их ролей: ученик, преподаватель и администратор. Она позволяет визуализировать поведение системы, очертить границы её функциональности и понять, как пользователи взаимодействуют с основными компонентами веб-приложения.

Основные варианты использования для ученика:

* просмотр текущих и завершённых абонементов;
* просмотр расписания предстоящих занятий;
* получение домашних заданий от преподавателя.

Основные варианты использования для преподавателя:

* назначение занятий для прикреплённых учеников;
* просмотр личного расписания;
* прикрепление новых учеников по коду или вручную;
* прикрепление домашних заданий к занятиям;
* управление списком абонементов учеников.

Основные варианты использования для администратора:

* регистрация и управление пользователями всех ролей;
* прикрепление учеников к преподавателям;
* управление абонементами (создание, продление, удаление);
* просмотр и анализ глобального расписания центра;
* контроль корректности выполнения домашних заданий.

Таким образом, диаграмма вариантов использования демонстрирует взаимодействие трёх ключевых ролей с системой и охватывает все основные сценарии, необходимые для эффективной работы логопедического центра. Этот этап проектирования позволяет обеспечить полноту и непротиворечивость требований при дальнейшем развитии архитектуры и интерфейса веб-приложения.

Диаграмма вариантов использования (Use Case Diagram) представлена на рисунке 1.

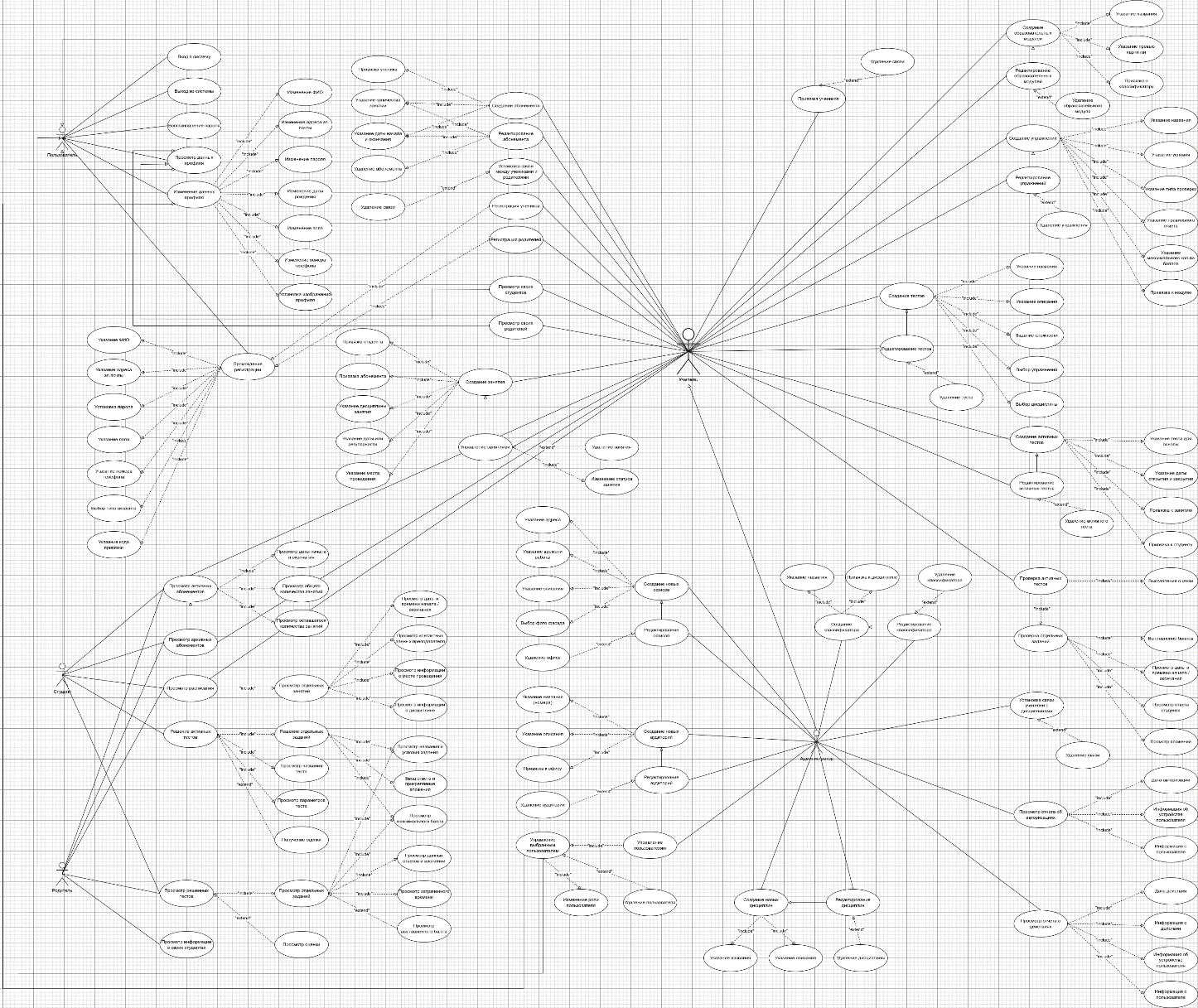


Рисунок 1 – Диаграмма вариантов использования

## 2.4 Диаграмма пользовательских историй (User Story Diagram)

Диаграмма пользовательских историй представляет взаимодействие пользователей с системой с точки зрения их конкретных целей. В отличие от диаграммы вариантов использования, которая описывает доступные действия, User Story диаграмма сосредотачивается на мотивации и ожидаемом результате этих действий, что особенно важно при проектировании ориентированных на пользователя решений.

Пользовательские истории формулируются по шаблону: «Я как [роль], хочу [цель и результат].»

Такой подход позволяет чётко зафиксировать потребности каждой роли в системе и служит основой для реализации функциональности, соответствующей реальным ожиданиям пользователей. Это важно как для формирования технических требований, так и для дальнейшего UX-дизайна и разработки интерфейсов.

Некоторые из основных пользовательских историй для каждой из групп приведены в таблице 6. Полная диаграмма пользовательских историй приведена в Приложении 1.

Таблица 6 – Основные пользовательские истории

|  |  |
| --- | --- |
| Роль | Пользовательская история |
| Пользователь | Я, как обычный пользователь, хочу выполнять вход в систему и выход из нее |
| Пользователь | Я, как обычный пользователь, хочу иметь возможность самостоятельно зарегистрироваться в системе |
| Студент | Я, как обучающийся, хочу просматривать свои активные и архивные абонементы |
| Студент | Я, как обучающийся, хочу просматривать расписание моих занятий |
| Учитель | Я, как учитель, хочу иметь возможность просматривать своих учеников и их родителей |
| Учитель | Я, как учитель, хочу, при создании абонемента, указывать количество приобретаемых занятий, дату начала и окончания срока действия абонемента |
| Администратор | Я, как администратор, хочу просматривать созданные мной офисы (места проведения занятий) |
| Администратор | Я, как администратор, хочу, при создании классификатора, указать его название и привязать его к дисциплине |
| Родитель | Я, как родитель, хочу просматривать все абонементы и информацию по ним для каждого закрепленного за мной обучающегося |
| Родитель | Я, как родитель, хочу видеть дату и время начала и окончания, контактные данные преподавателя, информацию о месте проведения, информацию о дисциплине по каждому занятию в расписании |

Диаграмма пользовательских историй является важным шагом в проектировании приложения, направленного на реальное удовлетворение потребностей участников логопедического центра – учеников, преподавателей и администраторов.

# 3 ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ ПРИЛОЖЕНИЯ

## 3.1 Построение ER-модели базы данных

Для проектирования базы данных была разработана ER-диаграмма (Entity-Relationship Diagram), отражающая взаимосвязи между ключевыми сущностями образовательной платформы. Данная база данных обеспечивает хранение и управление информацией о пользователях, преподавателях, учениках, администраторах, занятиях, подписках, образовательных модулях и других объектах системы.

Основные сущности базы данных:

1. Пользователи: главная сущность, содержащая общую информацию обо всех типах пользователей: ФИО, email, хэш пароля, дата рождения, пол, город, телефон, ссылка на фото и уникальный код.
2. Ученики: является подмножеством пользователей. Содержит дополнительную информацию: номер класса и название школы.
3. Преподаватели: также являются пользователями, но с полями: стаж и основное место работы.
4. Администраторы: представляют пользователей с правами управления. Атрибут «уровень доступа» определяет параметры доступа (например, только логи или полный доступ).
5. Филиалы: отражают местоположения, где могут проходить занятия. Содержат информацию об адресе, режиме работы, описании и ссылке на фото. Привязаны к администратору.
6. Кабинеты: местоположения внутри филиалов, в которых проводятся очные занятия. Содержат название, описание, ссылку на филиал и администратора.
7. Дисциплины: представляют собой учебные направления (например, математика, информатика). Содержат название, описание и связаны с администратором.
8. Подписки: связывают учеников и преподавателей в рамках обучения. Включают общее количество занятий, даты начала и окончания подписки, флаг архивации.
9. Занятия: конкретные учебные сессии, привязанные ко времени, ученику, преподавателю, дисциплине и, при необходимости, к кабинету или онлайн-ссылке. Имеют статус (запланировано, завершено, отменено и т.д.).
10. Классификаторы: используются для группировки дисциплин по направлениям. Содержат название и дату создания. Связаны с дисциплиной и администратором.
11. Образовательные модули: представляют собой учебные блоки с описанием и превью. Привязаны к классификатору и преподавателю.

ER-модель базы данных веб-приложения в нотации Питера Чена приведена на рисунке 2.

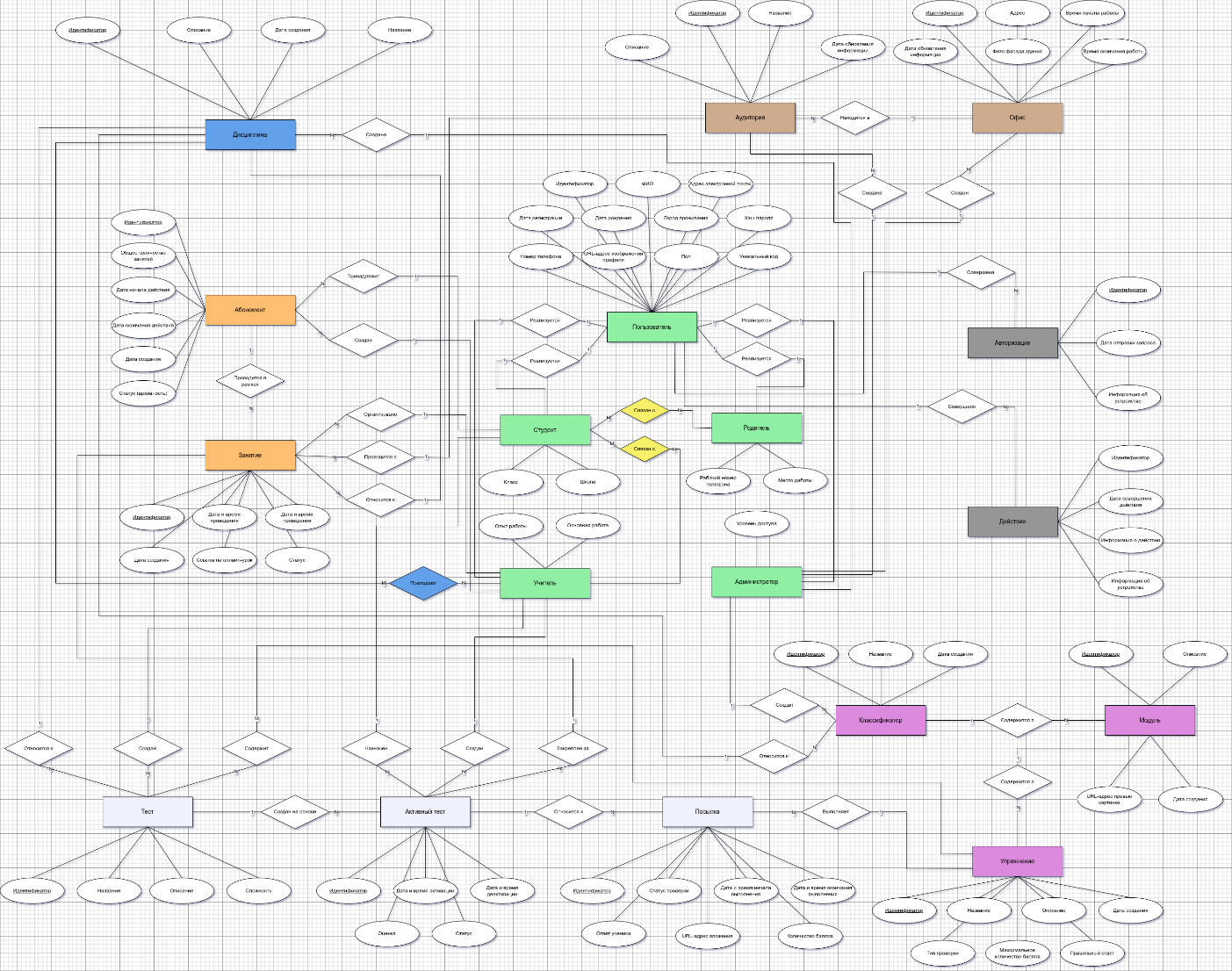


Рисунок 2 – ER-модель базы данных

## 3.2 Логическая модель базы данных

Логическая модель базы данных отражает структуру данных и взаимосвязи между сущностями, не привязанную к конкретной СУБД. Для построения модели использовалась методология IDEF1X (Integration Definition for Information Modeling) – стандарт, широко применяемый для проектирования реляционных баз данных.

Модель обеспечивает представление сущностей, их атрибутов, а также логических связей между ними. Все отношения между сущностями определяются в терминах ключевых зависимостей и правил целостности.

Описание основных сущностей и их атрибутов:

1. Сущность Users предназначена для хранения информации обо всех пользователях системы. Она содержит следующие атрибуты: user\_id — целочисленный идентификатор (тип INT, автоинкремент, первичный ключ); full\_name — полное имя пользователя (тип VARCHAR(255), обязательный); email — адрес электронной почты (тип VARCHAR(255), уникальный, обязательный); password\_hash — хеш пароля (тип VARCHAR(255), обязательный); birthday — дата рождения (тип DATE, необязательный); gender — пол (тип ENUM('male', 'female', 'other'), по умолчанию other); city — город проживания (тип VARCHAR(100), необязательный); phone\_number — номер телефона (тип VARCHAR(20), необязательный); profile\_picture\_url — URL-адрес фотографии профиля (тип TEXT, необязательный); unique\_code — уникальный код пользователя (тип VARCHAR(32), уникальный, обязательный).
2. Сущность Students расширяет Users, связываясь с ней по внешнему ключу user\_id (тип INT, внешний ключ на Users.user\_id, первичный ключ одновременно). Дополнительно содержит атрибуты: class\_number — номер класса (тип INT, обязательный); school\_name — наименование школы (тип VARCHAR(255), обязательный).
3. Сущность Teachers также расширяет Users, используя user\_id в качестве внешнего и первичного ключа. Включает атрибуты: experience — стаж работы в годах (тип INT, необязательный); main\_work — основное место работы (тип VARCHAR(255), необязательный).
4. Сущность Administrators определяет административных пользователей и включает один атрибут Users\_user\_id (тип INT, внешний ключ на Users.user\_id, одновременно первичный ключ). Также содержит атрибут access\_level — уровень доступа администратора (тип ENUM('low', 'medium', 'high'), по умолчанию low, обязательный).
5. Сущность Branches описывает филиалы. Содержит следующие атрибуты: branch\_id — уникальный идентификатор филиала (тип INT, автоинкремент, первичный ключ); administrator\_id — ссылка на администратора (тип INT, внешний ключ на Administrators.Users\_user\_id, обязательный); address — адрес филиала (тип VARCHAR(255), обязательный); working\_start и working\_end — время начала и окончания рабочего дня (оба типа TIME, обязательные); description — текстовое описание (тип TEXT, необязательный); photo\_url — ссылка на изображение филиала (тип TEXT, необязательный); updated\_at — дата последнего обновления (тип DATETIME, необязательный).
6. Сущность Classrooms описывает аудитории внутри филиалов. Содержит: classroom\_id — уникальный идентификатор аудитории (тип INT, автоинкремент, первичный ключ); branch\_id — внешний ключ на Branches.branch\_id (тип INT, обязательный); administrator\_id — внешний ключ на администратора (тип INT, обязательный); name — наименование аудитории (тип VARCHAR(100), обязательный); description — описание (тип TEXT, необязательный); updated\_at — время последнего изменения (тип DATETIME, необязательный).
7. Сущность Disciplines содержит список дисциплин, доступных в системе. Атрибуты включают: discipline\_id — уникальный идентификатор дисциплины (тип INT, автоинкремент, первичный ключ); administrator\_id — внешний ключ на администратора (тип INT, обязательный); name — название дисциплины (тип VARCHAR(100), обязательный); description — описание дисциплины (тип TEXT, необязательный); created\_at — дата создания (тип DATETIME, обязательный).
8. Сущность Subscriptions хранит данные об абонементах студентов. Включает: subscription\_id — уникальный идентификатор (тип INT, автоинкремент, первичный ключ); student\_id — внешний ключ на студента (тип INT, обязательный); teacher\_id — внешний ключ на преподавателя (тип INT, обязательный); total\_lessons — общее количество занятий (тип INT, обязательный); start\_date и end\_date — даты начала и окончания абонемента (тип DATE, обязательные); created\_at — дата создания абонемента (тип DATETIME, обязательный); in\_archive — логическое поле, указывающее, находится ли абонемент в архиве (тип BOOLEAN, по умолчанию FALSE).
9. Сущность Lessons описывает конкретные учебные занятия. Содержит: lesson\_id — уникальный идентификатор (тип INT, автоинкремент, первичный ключ); subscription\_id — внешний ключ на Subscriptions.subscription\_id (тип INT, обязательный); classroom\_id — внешний ключ на Classrooms.classroom\_id (тип INT, обязательный); discipline\_id — внешний ключ на Disciplines.discipline\_id (тип INT, обязательный); teacher\_id и student\_id — внешние ключи на Teachers.user\_id и Students.user\_id соответственно (тип INT, обязательные); lesson\_date\_time — дата и время проведения занятия (тип DATETIME, обязательный); duration — продолжительность занятия в минутах (тип INT, обязательный); status — статус занятия (тип ENUM('scheduled', 'completed', 'cancelled'), по умолчанию scheduled); created\_at — дата создания записи (тип DATETIME, обязательный); online\_call\_url — ссылка на онлайн-занятие (тип TEXT, необязательный).
10. Сущность Education\_classifier служит для классификации образовательных направлений. Имеет следующие поля: classifier\_id — идентификатор классификатора (тип INT, автоинкремент, первичный ключ); administrator\_id — внешний ключ на администратора (тип INT, обязательный); discipline\_id — внешний ключ на дисциплину (тип INT, обязательный); name — название классификатора (тип VARCHAR(100), обязательный); created\_at — дата создания (тип DATETIME, обязательный).
11. Сущность Education\_modules описывает модули обучения в рамках классификатора. Включает: module\_id — идентификатор модуля (тип INT, автоинкремент, первичный ключ); classifier\_id — внешний ключ на Education\_classifier.classifier\_id (тип INT, обязательный); teacher\_id — внешний ключ на Teachers.user\_id (тип INT, обязательный); description — описание модуля (тип TEXT, необязательный); preview\_url — ссылка на изображение (тип TEXT, необязательный); created\_at — дата создания (тип DATETIME, обязательный).

Логическая модель базы данных в нотации IDEF1X приведена на рисунке 3.

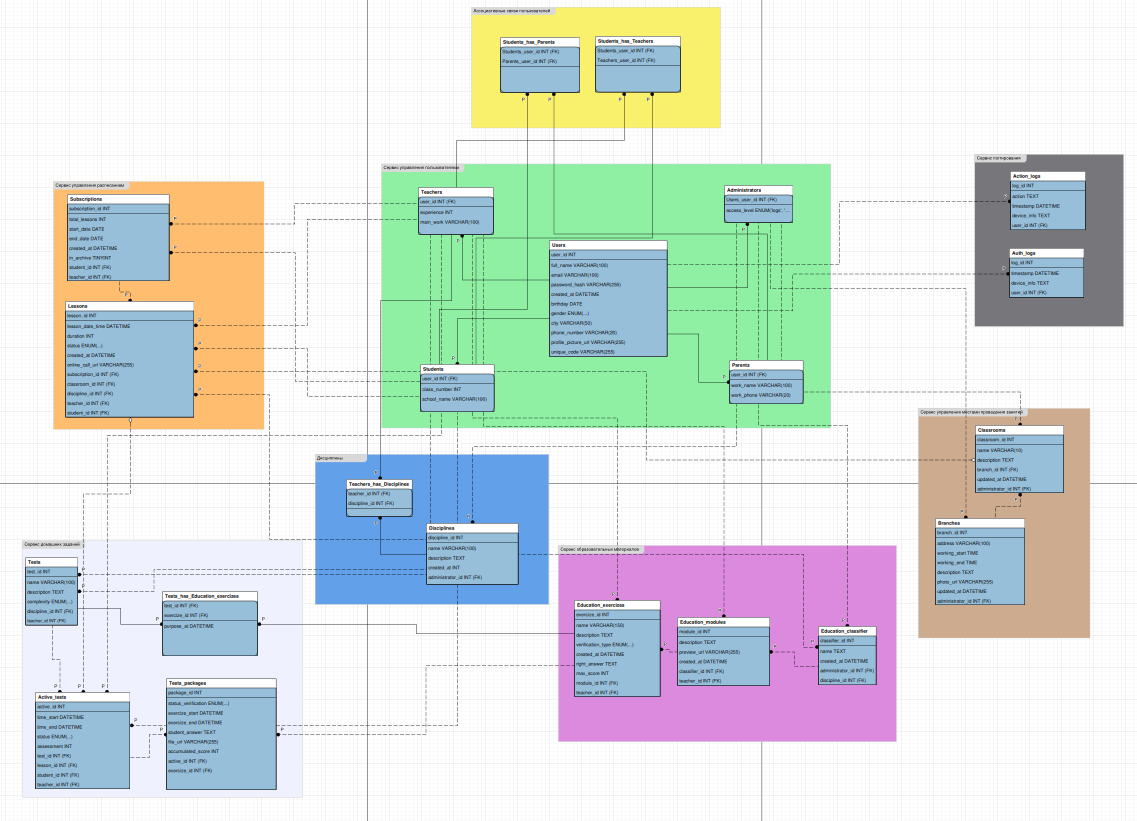


Рисунок 3 – Логическая модель базы данных в нотации IDEF1X

## 3.3 Физическая модель базы данных

Физическая модель представляет собой реализацию логической модели базы данных на конкретной СУБД, в нашем случае – MySQL. Она оформляется в виде SQL-скрипта, который содержит инструкции для создания всех таблиц, определения типов данных, ограничений (ключей, связей, уникальности, NOT NULL и др.), а также возможных индексов и дефолтных значений. Эта модель обеспечивает физическое хранение данных, соблюдение целостности и поддержку производительности при выполнении запросов. SQL-скрипт для создания базы данных приведен в Приложении 2.

# 4 РАЗРАБОТКА СЕРВЕРНОЙ ЧАСТИ ПРИЛОЖЕНИЯ

## 4.1 Структура проекта

Проект организован в виде набора файлов и папок, что способствует удобству разработки, тестирования и поддержки. Основные компоненты системы разделены по функциональному признаку, что позволяет ясно разграничить области ответственности.

В корне проекта находятся файлы конфигурации и инициализации, а также несколько ключевых директорий:

* repositories — содержит модули для работы с данными. Здесь реализованы классы и функции, отвечающие за взаимодействие с базой данных, включая выполнение запросов и логику доступа к сущностям. Такой подход позволяет изолировать бизнес-логику от деталей хранения данных;
* routes — содержит описание маршрутов (endpoint-ов) веб-приложения. Здесь прописываются URL-адреса и методы HTTP-запросов (GET, POST и др.), а также связанная с ними логика обработки запросов. Это центральное место для реализации API и взаимодействия с клиентом;
* utils — папка с вспомогательными утилитами и общими функциями, которые используются в разных частях проекта. Это могут быть инструменты для валидации данных, генерации ответов, обработки ошибок и другие полезные функции, не относящиеся напрямую к бизнес-логике;
* migrations — директория, предназначенная для хранения файлов миграций базы данных. Миграции позволяют последовательно изменять структуру базы данных (создавать таблицы, добавлять колонки и т.п.) без потери данных и с возможностью отката.

Такое разделение кода упрощает масштабирование приложения, облегчает тестирование и позволяет разным разработчикам работать параллельно над разными аспектами проекта.

# ЗАКЛЮЧЕНИЕ

СПИСОК ЛИТЕРАТУРЫ