Министерство цифрового развития, связи и массовых коммуникаций Российской Федерации
федеральное государственное бюджетное образовательное учреждение высшего образования
«Сибирский государственный университет телекоммуникаций и информатики»
(СибГУТИ)

09.03.01 Информатика и вычислительная техника
(направление подготовки/специальность)
Программное обеспечение средств вычислительной техники и автоматизированных систем
(профиль/специализация)
очная
(форма обучения)

ОТЧЕТ ПО ПРОИЗВОДСТВЕННОЙ ПРАКТИКЕ
(*вид практики)*

Тип практики Преддипломная практика
в/на кафедре прикладной математики и кибернетики
(наименование профильной организации /структурного подразделения СибГУТИ)

ТЕМА ИНДИВИДУАЛЬНОГО ЗАДАНИЯ

Веб-сервис для изучения SQL с применением больших языковых моделей

Выполнил: Фунтиков Илья Олегович
студент института Информатики и вычислительной техники
группа ИП212

_____/Фунтиков И.О./
«30» мая 2026г. *(подпись)* *(ФИО)*

Проверил:
Руководитель практики от СибГУТИ

_____/Янченко Е.В./
«30» мая 2026г. *(подпись)* *ФИО)*

отметка _____ _____«30» мая 2026г.

Новосибирск 2026

НАПРАВЛЕНИЕ НА ПРАКТИКУ

На основе договора[1] _____от _____
федеральное государственное бюджетное образовательное учреждение высшего образования "Сибирский государственный университет телекоммуникаций и информатики" (СибГУТИ) направляет студента 4 курса, института информатики и вычислительной техники гр. ИП212
_____Фунтикова Илью Олеговича_____
(ФИО обучающегося)
для прохождения практики с 27.04.2026 по 30.05.2026
в/на кафедре прикладной математики и кибернетики_____
(наименование организации/кафедры/структурного подразделения СибГУТИ)

МП Директор института[3] _____/_____/

ОХРАНА ТРУДА[2]

Вводный инструктаж доцент каф.ПМиК Янченко Е.В._____
(должность, Ф.И.О.)

«27» апреля 2026 г. _____
(подпись)

Зачет принял доцент каф.ПМиК Янченко Е.В._____
(должность, Ф.И.О.)

«27» апреля 2026 г. _____
(подпись)

1 При прохождении практики на кафедре оставить пустым.

2 Вводный инструктаж проводится по месту прохождения практики.

План-график проведения практики

Направление: 09.03.01 Информатика и вычислительная техника _____
(код, наименование направления (специальности))
Направленность (профиль)/ специализация: Программное обеспечение средств вычислительной техники и автоматизированных систем
Объем практики: 252 / 7 часов/ЗЕ
Тип практики: Преддипломная практика

Содержание практики:
Тема индивидуального задания практики Веб-сервис для изучения SQL с применением больших языковых моделей
(тема ВКР)

Наименование видов деятельности в соответствии с рабочей программой практики	Дата (начало – окончание)
Постановка задачи на практику, определение конкретной индивидуальной темы, формирование плана работ	27.04.2026 – 29.04.2026
Работа с библиотечными фондами, сбор и анализ материалов по теме практики	30.04.2026 – 03.05.2026
Выполнение работ в соответствии с составленным планом: *1. Провести анализ предметной области* *2. Сформулировать функциональные требования к приложению* *3. Разработать серверную часть* *4. Разработать клиентскую часть*	04.05.2026 – 09.05.2026 10.05.2026 – 15.05.2026 16.05.2026 – 21.05.2026 22.05.2026 – 28.05.2026
Анализ полученных результатов и произведенной работы, составление отчета по практике	29.05.2026 – 30.05.2026

Руководитель практики
от профильной организации[3] _____/_____./
«24» апреля 2026 г. *(подпись)* *(ФИО)*

Руководитель практики от СибГУТИ _____/ Янченко Е.В./
«24» апреля 2026 г. *(подпись)* *(ФИО)*

Студент _____/Фунтиков И.О./
«24» апреля 2026 г. *(подпись)* *(ФИО)*

3 При прохождении на кафедре оставить пустым.

ДНЕВНИК РАБОТЫ

Дата/период	Рабочее место и краткое содержание выполняемых работ	Отметка о выполнении[4] (выполнено/ не выполнено)
27.04.2026 - 28.04.2026	Ознакомление со структурным подразделением предприятия, вводный инструктаж по технике безопасности	выполнено
28.04.2026 - 29.04.2026	Получение задание на практику, определение конкретной индивидуальной темы, формирование плана работ	выполнено
29.04.2026 - 02.05.2026	Работа с библиотечными фондами структурного подразделения, сбор и анализ материалов по теме практики	выполнено
04.05.2026 - 06.05.2026	Проведение анализ предметной области	выполнено
07.05.2026 - 11.05.2026	Разработка архитектуры и формулирование функциональных требований	выполнено
12.05.2026 - 13.05.2026	Разработка функциональных возможностей приложения	выполнено
14.05.2026 - 15.05.2026	Разработка серверной части	выполнено
16.05.2026 - 22.05.2026	Разработка клиентской части	выполнено
27.05.2026 - 30.05.2026	Анализ полученных результатов, составление отчета по практике.	выполнено

4 Заполняется руководителем практики от предприятия или руководителем практики от СибГУТИ при прохождении на кафедре.

КРАТКАЯ ХАРАКТЕРИСТИКА РАБОТЫ ПРАКТИКАНТА

Отзыв руководителя от предприятия о работе студента (выполнение программы практики, овладение производственными навыками, отношение к работе, трудовая дисциплина и др.)[5]

За время прохождения практики Фунтиков Илья Олегович показал свои профессиональные навыки и хороший уровень теоретической и практической подготовки; проявил себя, как ответственный исполнитель. Студент способен самостоятельно находить способы и методы решения профессиональных задач.
Программа практики выполнена в полном объеме.

Руководитель практики от предприятия: _____/_____/

Отзыв руководителя от СибГУТИ о работе студента (выполнение программы практики, овладение производственными навыками, отношение к работе, трудовая дисциплина и др.)

Компетенции	Уровень сформированности компетенций[6]
ПК-22 Способен проводить оптимизацию функционирования баз данных	
ПК-4 Способен разрабатывать компоненты системных программных продуктов	
ПК-25 Способен выполнять интеграцию программных модулей и компонент, проверять работоспособность выпусков программного продукта	
ПК-3 Способен разрабатывать графический дизайн интерфейса, проектировать пользовательский интерфейсы по готовому образцу или концепции интерфейса	

отметка о зачете (оценка) _____

Руководитель практики от СибГУТИ: _____/Янченко Е.В./

5 Оставить пустым в случае прохождения практики на кафедре.

6 Уровень сформированности компетенций: высокий, средний, низкий, не сформирована.

СОДЕРЖАНИЕ

ТЕКСТ ЗАДАНИЯ НА ПРЕДДИПЛОМНУЮ ПРАКТИКУ

1. Провести подробный аналитический обзор существующих платформ и тренажеров для интерактивного изучения SQL, выявить их преимущества, недостатки и функциональные разрывы.
2. Сформулировать функциональные и нефункциональные требования к проектируемому веб-приложению.
3. Разработать архитектуру веб-приложения, основанную на микросервисном подходе.
4. Реализовать механизмы аутентификации пользователей, хранения метаданных, результатов тестирования и разграничения прав доступа.
5. Разработать специализированный микросервис для динамического управления базами данных, безопасного выполнения запросов во временном изолированном окружении и автоматической генерации ER-диаграмм связей.
6. Разработать микросервис для интеграции с локальной языковой моделью, обеспечивающей генерацию корректных SQL-вопросов разного уровня сложности на основе схемы БД.
7. Разработать клиентскую часть приложения.
8. Провести анализ разработанной системы и подготовить отчет по преддипломной практике.

ВВЕДЕНИЕ

В современных образовательных учреждениях и онлайн-образовательных программах преподавание SQL традиционно строится вокруг готовых методических материалов и ручного создания упражнений преподавателями. Несмотря на наличие большого количества онлайн-курсов и учебников, формирование эффективной практической базы часто оказывается затрудненным из-за ограниченного числа заданий и трудоемкости их подготовки. Преподаватели вынуждены тратить значительные ресурсы на проектирование баз данных, продумывание разнообразных сценариев задач и ручную проверку ответов. Студенты при этом сталкиваются с недостатком адаптивных упражнений, что замедляет их развитие и снижает вовлеченность.

В то же время рынок труда предъявляет всё более высокие требования к владению SQL: профессиональные аналитики, дата-инженеры и специалисты по машинному обучению должны уверенно использовать как базовые конструкции запросов, так и продвинутые возможности реляционных СУБД. Однако существующие платформы для практики SQL либо не предоставляют инструментов для создания индивидуализированных заданий, либо не предлагают преподавателям средств для оценки и анализа результатов, что создает разрыв между образовательным процессом и потребностями индустрии.

На фоне этих вызовов растет интерес к инструментам, способным автоматизировать рутинные задачи преподавателей и предоставить студентам интерактивную и адаптивную среду обучения. Особенно перспективной представляется интеграция локальных больших языковых моделей (LLM), которая позволит генерировать разнообразные задания на основе реальных структур баз данных, а также автоматизировать проверку решений, сохраняя при этом конфиденциальность учебных данных.

Таким образом, цель данной работы — разработка веб-сервиса, обеспечивающего автоматизацию создания и проверки заданий на изучение SQL, визуализацию схем баз данных и инструменты аналитики, что улучшит процесс обучения как для преподавателей, так и для студентов.

1 ИССЛЕДОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ

По оценкам, спрос на специалистов с навыками работы с данными будет расти (США: вакансий для дата-аналитиков - +36 % к 2033 г. по данным Бюро статистики труда США [1]). По данным опроса разработчиков 2023 г. 48,66 % разработчиков и аналитиков регулярно используют SQL в своей работе [2].

Анализ открытых вакансий также свидетельствует о растущем спросе: годовой рост вакансий с упоминанием SQL за год составил 46 % [3]. Компании из самых разных отраслей (IT, финансы, медицина, ритейл) включают SQL в требования к аналитикам и разработчикам [3], поскольку «ведущие компании собирают и анализируют огромные объёмы данных», что характерно для четвёртой промышленной революции.

Таким образом, владение SQL входит в число ключевых аналитических навыков, необходимых для работы с данными [4].

В университете SQL обычно преподают в рамках курсов баз данных и анализа данных. Практические занятия (работа с реальными базами данных и наборами данных) оказываются наиболее эффективными: исследования показывают, что hands-on обучение - лучший способ освоения SQL [5].

В современном мире искусственный интеллект становится основной технологией, которая помогает автоматизировать задачи, которые раньше были доступны только человеку. Особенно ярко это проявляется в сфере IT-образования. Большие языковые модели (LLM) дают новые возможности для создания учебных материалов, проверки заданий и анализа прогресса студентов. Их способность обрабатывать информацию, включая схемы баз данных, позволяет создавать учебные планы, адаптированные под уровень подготовки учащихся.

Возможность использования локальных языковых моделей обеспечивает не только снижение затрат на инфраструктуру (при наличии необходимых мощностей), но и повышает безопасность: данные остаются в пределах внутренней сети, что критически важно для образовательных учреждений.

Успешные кейсы, такие как интеграция YandexGPT в учебную платформу “Яндекс Практикум”, или использование GPT-4 на платформе Khan Academy, подтверждают эффективность ИИ в образовании.

Однако большинство существующих решений фокусируются на RAG системах для дополнительного объяснения студенту материала. Проект, представленный в рамках этой дипломной работы, делает акцент именно на автоматизацию процесса создания материалов для тестирования уровня студентов. Преподавателям сложно создавать разнообразные тестовые задания. Разработка каждого вопроса требует ручного проектирования баз данных, формулировки условий и проверки корректности ответов. Это не только увеличивает нагрузку на преподавателей, но и ограничивает вариативность заданий, что снижает мотивацию студентов.

Вывод. SQL остаётся ключевым навыком в аналитике и образовательных технологиях. Несмотря на трудности обучения и нужду в новых инструментах, сочетание традиционных подходов (hands-on, групповые проекты) и инновационных средств (адаптивные задания, AI-ассистенты) обещает повысить эффективность обучения SQL. Успешные практики уже демонстрируют, что вовлечение ИИ может ускорить получение навыка и сделать обучение более индивидуальным.

1.1 Анализ аналогичных продуктов

1.1.1 Сервис HackerRank для SQL

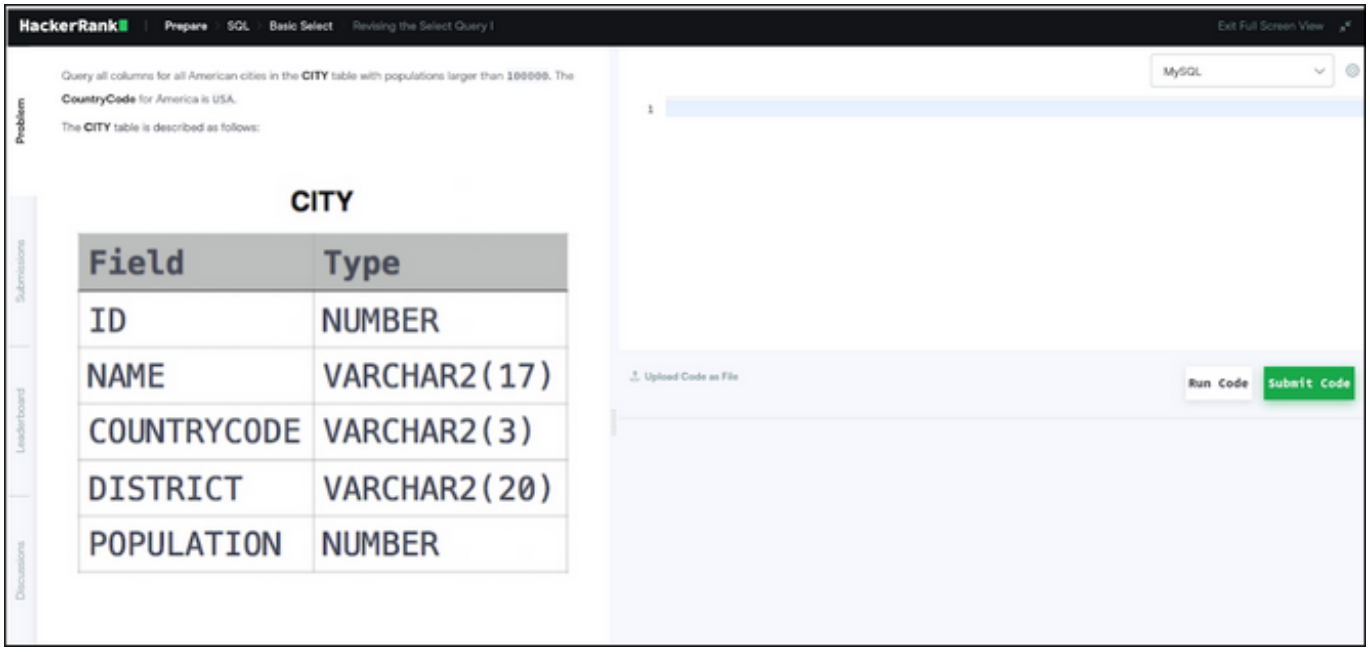

Рис. 1. Интерфейс решения задачи на сервисе HackerRank

- **Сильные стороны**: готовые задания, интеграция с корпоративными учебными программами.
- **Ограничения**: создание заданий преподавателем вручную, отсутствие автоматической визуализации данных и ER-диаграмм, невозможность отслеживания результатов студентов.

1.1.2 Сервис LeetCode для SQL

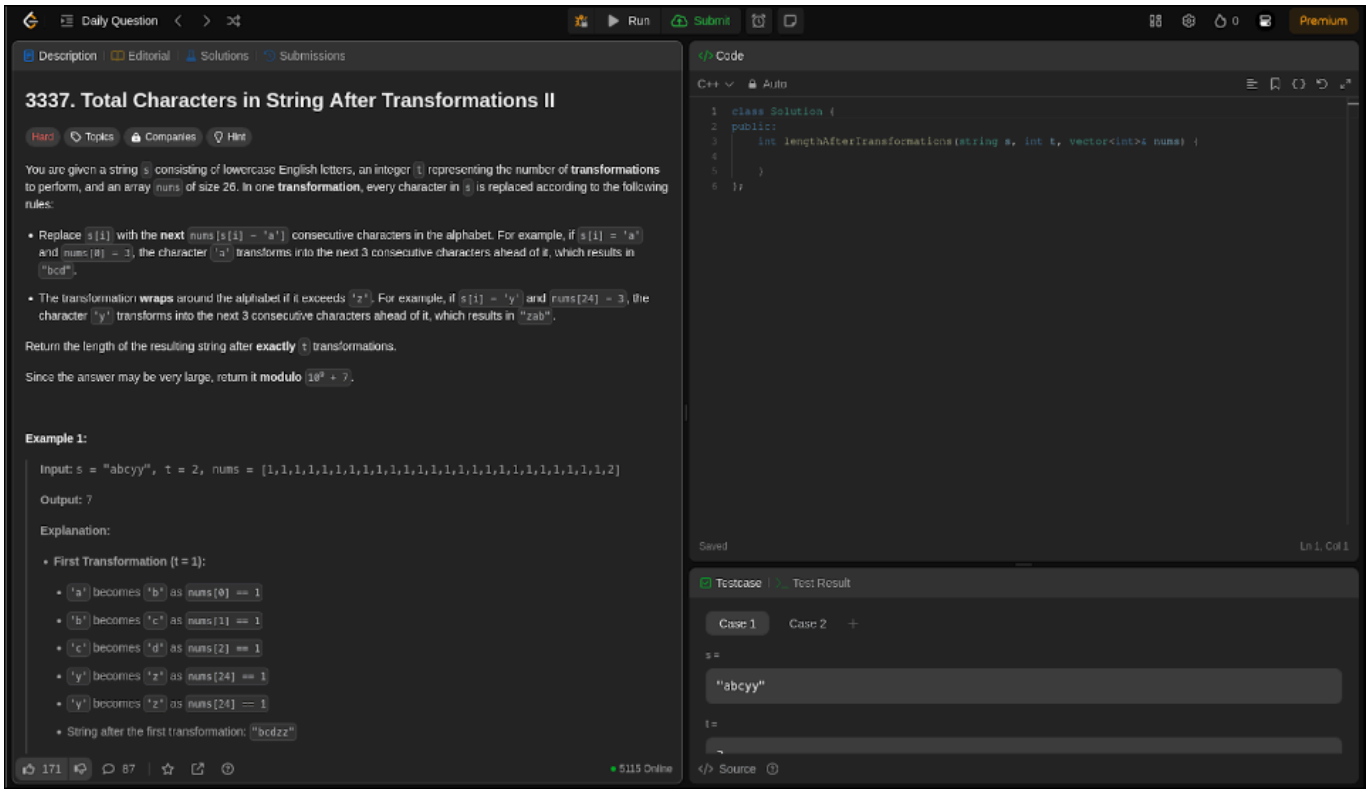

Рис. 2. Интерфейс решения задачи на сервисе LeetCode

- **Сильные стороны**: большой выбор заданий, популярность среди сообщества, удобный интерфейс.
- **Ограничения**: задания создаются вручную, нет ER-диаграмм, нет инструментов преподавателю для отслеживания прогресса студентов.

Вывод по анализу аналогов. Существующие платформы предлагают удобный интерфейс и готовый контент, но они направлены в первую очередь на самообразование и формирование банка задач с помощью сообщества. Они не автоматизируют процесс создания вопросов и не предоставляют полноценную аналитику для преподавателя.

2 ПОСТАНОВКА ЗАДАЧИ

2.1 Функциональные требования к системе

Задачей выпускной квалификационной работы являлась разработка веб-сервиса для интерактивного обучения студентов языку SQL на практических задачах.

Система должна обеспечивать выполнение следующих функций:

1. Работа с базами данных: возможность загрузки файлов баз данных в формате SQLite, автоматическое извлечение и визуализация схемы данных в виде ER-диаграмм, получение текстового представления схемы для анализа LLM.
2. Автоматизация создания заданий: интеграция с большими языковыми моделями для автоматической генерации вопросов на основе анализа схемы базы данных, настройка уровней сложности заданий.
3. Проверка решений: автоматическое выполнение SQL-запросов студента на загруженной базе данных и сравнение результатов с верными ответами.
4. Система управления тестированием: интерфейс для преподавателей для создания и настройки тестов, выбора баз данных, модуль отслеживания прогресса студентов, формирование отчетов о результатах прохождения каждого теста.

2.2 Требования к интерфейсу пользователя

Интерфейс системы должен быть разделен на две основные роли с соответствующим функционалом:

Для преподавателя:

1. Личный кабинет с возможностью загрузки баз данных, управления тестами и просмотра аналитики.
2. Инструменты создания вопросов вручную или автоматически с настройкой сложности.
3. Просмотр статистики прохождения тестов студентами (таблицы результатов и диаграммы распределения оценок).

Для студента:

1. Интерактивный редактор SQL-запросов с подсветкой синтаксиса.
2. Визуализация структуры базы данных (ER-диаграмма) непосредственно в интерфейсе решения задания.
3. Возможность получения мгновенной обратной связи по результатам выполнения запроса.
4. Возможность отправки ответа и завершения тестирования с сохранением результата.

3 ВЫБОР СРЕДСТВ РАЗРАБОТКИ

3.1 Система управления базами данных

В качестве основы для хранения пользовательских данных и структуры схем баз данных преподавателей был выбран Supabase [6] - полноценная Backend as a Service платформа с открытым исходным кодом, построенная на базе PostgreSQL. Supabase предоставляет мощный RESTful и realtime API поверх PostgreSQL, встроенную систему аутентификации и авторизации, совместимое с S3 хранилище файлов, а также удобную панель управления. В рамках данного проекта S3 хранилище используется для хранения SQLite-баз данных, которые загружают преподаватели для каждого теста.

Основные причины выбора этого решения:

1. Автоматическая генерация RESTful и Realtime API на основе схемы PostgreSQL - Supabase мгновенно создаёт API-эндпоинты для каждой таблицы и представления.
2. Встроенная система аутентификации с поддержкой OAuth - провайдеров, магических ссылок и управления ролями, что позволяет гибко настраивать доступ преподавателей и студентов [6].
3. Панель управления с возможностью визуального редактирования схем, управления RLS политиками доступа и мониторинга запросов.
4. Официальные клиентские библиотеки для JavaScript, Python и других языков, которые предоставляют удобный ORM-подобный интерфейс для работы с данными.
5. Встроенное S3-совместимое хранилище файлов (Supabase Storage), которое в проекте применяется для загрузки и хранения SQLite-баз данных преподавателями.

Благодаря открытому исходному коду Supabase может быть развёрнут на собственном сервере, что даёт полный контроль над данными и инфраструктурой, что может быть критично, учитывая политику обработки персональных данных в образовательных учреждениях. Однако, доступна и облачная версия, которая позволяет быстро начать разработку без настройки серверного окружения.

3.2 Серверная часть

Серверная часть проекта была реализована на Python 3.13 с использованием FastAPI [7] - асинхронного веб-фреймворка, построенного на Starlette [8] и Pydantic [9]. Благодаря автоматической генерации интерактивной документации в Swagger UI есть возможность быстро интегрировать фронтенд и с комфортом тестировать реализованные API пути, а строгая валидация входных данных и ответов через Pydantic-модели сокращает число ошибок на этапе обмена данными.

Асинхронная обработка запросов обеспечивает хорошую пропускную способность бэкенда при одновременных соединениях, а в качестве сервера был применён Uvicorn [10], поддерживающий режим autoreload для ускорения локальной разработки.

3.3 Клиентская часть

Клиентская часть была построена на основе стандартных технологий HTML, CSS и JavaScript, дополненных современным фреймворком Vue.js [11] и сборщиком Vite.

Выбранный подход обеспечил компонентную архитектуру, где каждый элемент интерфейса оформлен в виде Vue-компонента, что дало возможности для повторного использования кода и улучшило архитектуру проекта. Реактивная система Vue автоматически отслеживает изменения состояний и обновляет представление без ручных манипуляций с DOM деревом.

3.4 Интеграция с большими языковыми моделями

Для автоматической генерации заданий был использован проект Ollama [12], который позволяет локально развернуть любую LLM, веса которой находятся в свободном доступе. API интерфейс

Ollama совместим с GPT REST API интерфейсом [13], который уже стал стандартом в индустрии и используется во всех приличных инструментах для разработки приложений с LLM.

Такой подход позволил запустить dense-версию модели qwen3:8b [14] объёмом около 5,2 ГБ на локальном оборудовании, обеспечив быстрое и предсказуемое время ответа без зависимости от внешних сервисов.

Оркестрация вызовов модели и пост-обработка результатов были организованы с помощью Python фреймворка LangChain [15]. Этот фреймворк позволяет сформировать цепочки обработки запросов - от загрузки входных данных и вызова LLM до проверки структуры ответа модели.

Также стоит отметить, что использование LangChain позволило унифицировать работу с разными провайдерами моделей. Хоть в данной дипломной работе используется локальная модель, инференс которой был произведен через Ollama, крайне просто заменить провайдера на любого облачного (OpenAI, DeepSeek, Claude, Qwen, OpenRouter, и т.д.).

3.5 Среда разработки

В качестве основной среды программирования была применена Visual Studio Code, обеспечивающая богатую экосистему расширений для Python, Go, JavaScript. Встроенные возможности терминала и система задач позволили запускать тесты и скрипты без переключения между окнами, а графический интерфейс Git облегчил работу с ветвлением и коммитами.

4 ПРОГРАММНАЯ РЕАЛИЗАЦИЯ

4.1 Архитектура системы

Веб-сервис имеет трёхзвенную архитектуру и состоит из трёх основных компонентов: клиентская часть (Frontend), два микросервиса на FastAPI и облачная инфраструктура Supabase.

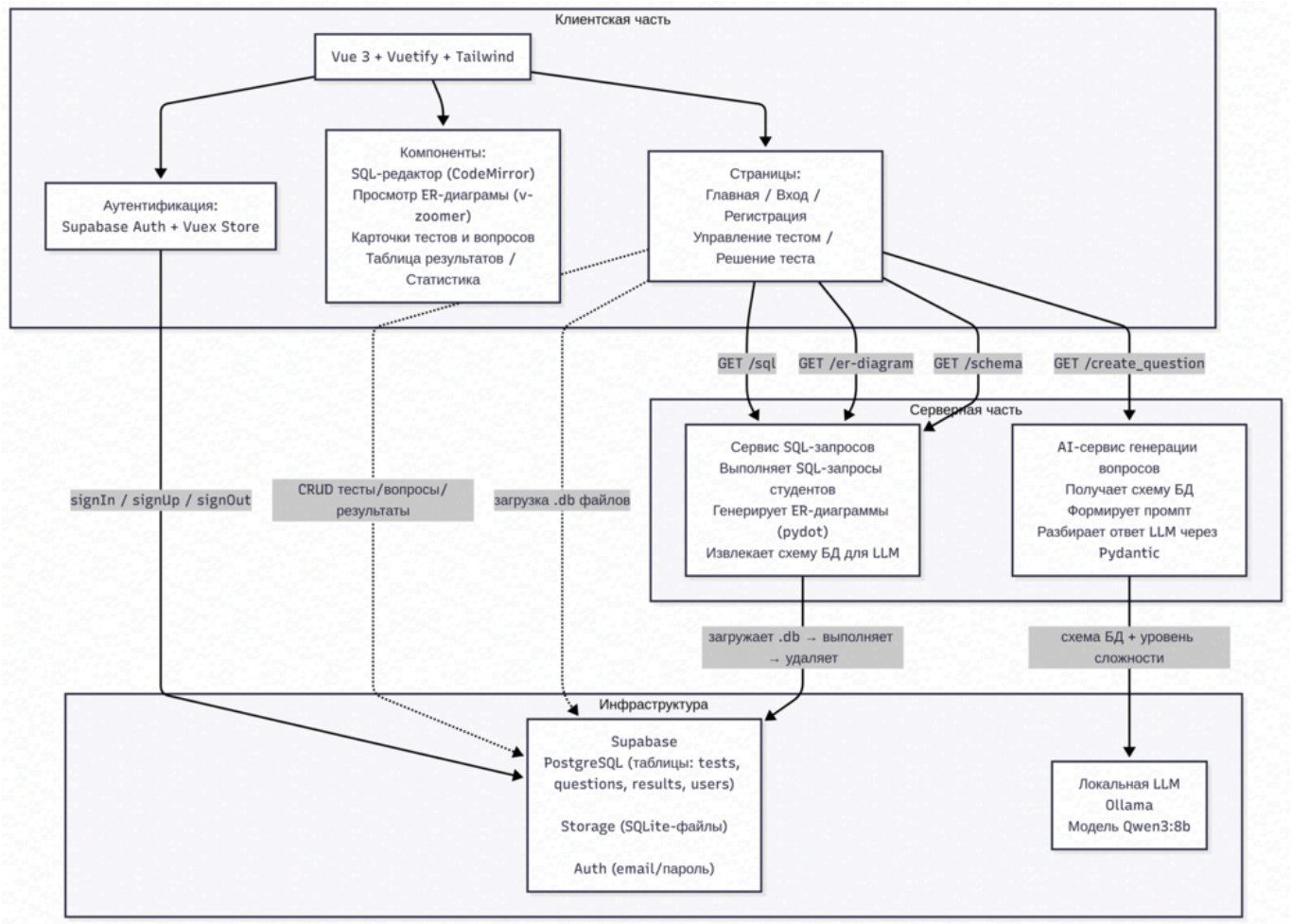

Рис. 3. Архитектура веб-приложения

Клиентская часть (Frontend). Реализована на веб фреймворке Vue3. В современной веб-разработке редко встретишь чистый CSS, поэтому в данной работе была использована самая актуальная на данный момент библиотека для верстки - Tailwind CSS [16]. Ключевой особенностью данной библиотеки является возможность писать в классе HTML тэга классические css свойства. Такой подход сильно ускоряет и упрощает разработку, особенно в связке с Vuetify [17].

Vuetify - это библиотека компонентов интерфейса, написанная специально для фреймворка Vue.js. Она предоставляет обширный набор компонентов, таких как модальные окна, кнопки, поля ввода, контейнеры, карточки и т.д. Этот набор компонентов строго соблюдает все правила и гайдлайны от Google в дизайн системе Material 3 [18], что позволяет сконцентрироваться на функционале, а не создании собственной дизайн-системы.

Серверная часть (Backend). Разделена на два микросервиса для изоляции ответственности и возможности независимого масштабирования при возрастании нагрузки на одну из частей сервиса:

1. SQLite Query Service - основной бэкенд для работы с базами данных. Он отвечает за обработку запросов и выполнение SQL-кода на базе данных для теста, извлечение схемы и генерацию ER-диаграмм. Сервис получает UUID теста, получает по ID теста путь к базе данных, загружает

нужный файл .db из Supabase S3 Storage во временный файл, выполняет операцию и удаляет файл.

2. AI Service - сервис для генерации вопросов. Получает текстовое описание схемы БД из SQLite Query Service и передаёт его в LLM через LangChain.

Инфраструктура Supabase. Используется как: база данных PostgreSQL, сервис аутентификации, файловое хранилище (Storage для SQLite-файлов), RESTful API с автоматической генерацией эндпоинтов.

4.1.1 Схема взаимодействия при выполнении ключевых сценариев

Сценарий 1. Создание теста.

Преподаватель заполняет форму с полями «Вопрос» и «Ответ», где «Вопрос» — это формулировка задачи для которой студент должен написать SQL запрос, а «Ответ» — это SQL запрос, который выполняет поставленную задачу.

Фронтенд загружает файл `.db` в Storage и получает ссылку на файл базы данных из Storage

URL файла сохраняется в поле `db_file_url`.

Фронтенд создаёт запись в таблице `tests` через API запрос к Supabase

id uuid	title text	description text	teacher_id uuid	db_file_url text
b19c90c5-7ad8-4c01-b2e2-fb9d71d62973	Тест на JOIN	Тест проверяет умение использовать оп	1abe7b5a-372b-4958-a2c6-c6f95...	https://hlbwkzdazfqxhqylpwne.supabase

Рис. 4. Запись в таблице tests

Сценарий 2: Генерация вопроса через AI.

Преподаватель нажимает «Сгенерировать»

Фронтенд обращается к эндпоинту `/schema` в SQLite Query Service

Получает текстовую схему

Отправляет в `/create_question` из AI Service

LLM генерирует reasoning, формулировку вопроса и правильный ответ

Фронтенд отображает результат для подтверждения

Преподаватель сохраняет вопрос в Supabase.

Сценарий 3: Решение теста студентом.

Студент открывает страницу теста

Фронтенд получает список вопросов из Supabase и ER-диаграмму из SQLite Query Service

Студент пишет SQL-запрос в редакторе

Нажимает «Выполнить»

Запрос отправляется на `/sql` в SQLite Query Service

Результат отображается в таблице

При нажатии «Отправить ответ» результат запроса студента сравнивается с результатом эталонного запроса

После ответа на все вопросы результаты отправляются в таблицу `results` для дальнейшего отображения статистики в личном кабинете преподавателя

4.2 Детальная реализация SQLite Query Service

Сервис реализован на FastAPI. Он предоставляет три эндпоинта, каждый из которых работает по общему принципу: получение записи теста из Supabase, загрузка .db файла во временную директорию, выполнение операции, удаление временного файла.

4.2.1 Эндпоинт `/sql` выполнение SQL-запросов

Параметры: `record_id: str` (UUID теста), `query: str` (SQL-запрос).

```
@router.get("/sql")
async def execute_sql(record_id: str, query: str):
    file_name = None
    conn = None
    try:
        record = get_test_record(record_id)
        if not record:
            raise HTTPException(status_code=404)
        db_file_url = record.get("db_file_url")
        if not db_file_url:
            raise HTTPException(status_code=400)
        file_name = str(uuid.uuid4())
        download_db_file(db_file_url, file_name)
        conn = sqlite3.connect(file_name)
        cursor = conn.cursor()
        cursor.execute(query)
        result_data = cursor.fetchall()
        column_names = [desc[0] for desc in cursor.description]
        result = [
            {column: value for column, value in zip(column_names, row)}
            for row in result_data
        ]
        conn.close()
        os.remove(file_name)
        return {"columns": column_names, "result": result}
    except HTTPException:
        raise
    except Exception as e:
        if conn: conn.close()
        if file_name and os.path.exists(file_name):
            os.remove(file_name)
        raise HTTPException(status_code=500, detail=str(e))
```

Листинг 1. Функция выполнения SQL-запроса

Функция принимает идентификатор теста и текст SQL-запроса. Загружает соответствующую БД из Supabase Storage, выполняет запрос через sqlite3 [19], возвращает имена колонок и строки результата в формате JSON. Гарантирует очистку временного файла даже при ошибке.

4.2.2 Эндпоинт `/er-diagram` генерация ER-диаграммы

Параметры: `record_id: str`.

Возвращает ER-диаграмму в формате PNG, закодированную в Base64. Генерация выполняется библиотекой `pydot` [20]:

```
        def generate_er_diagram(database_file):
            conn = sqlite3.connect(database_file)
            cursor = conn.cursor()
            cursor.execute("SELECT name FROM sqlite_master WHERE type='table';")
            tables = cursor.fetchall()
            table_info = {}
            for table in tables:
                cursor.execute(f"PRAGMA table_info({table[0]});")
                columns = cursor.fetchall()
                cursor.execute(f"PRAGMA foreign_key_list({table[0]});")
                foreign_keys = cursor.fetchall()
                table_info[table[0]] = (columns, foreign_keys)
            graph = pydot.Dot(graph_type='graph', rankdir='LR')
            for table, (columns, foreign_keys) in table_info.items():
                node = pydot.Node(table, shape='record')
                label = f'{{ {table} | {{'
                for column in columns:
                    label += f' {column[1]} : {column[2]}\\l'
                label += '} }}'
                node.set_label(label)
                graph.add_node(node)
                for fk in foreign_keys:
                    arrowhead = 'crow' if table < fk[2] else 'tee'
                    edge = pydot.Edge(
                        f"{table}:{fk[3]}",
                        f"{fk[2]}:{fk[4]}",
                        arrowhead=arrowhead
                    )
                    graph.add_edge(edge)
            return graph.create(format='png')
```

Листинг 2. Функция генерации ER-диаграммы

Функция подключается к SQLite-файлу, извлекает метаданные о таблицах через `PRAGMA table_info` и `PRAGMA foreign_key_list`, строит граф с помощью pydot. Каждая таблица отображается как узел с перечислением колонок, внешние ключи - как рёбра с разными типами стрелок для указания направления связи.

4.2.3 Эндпоинт `/schema` получение схемы БД

Параметры: `record_id: str`.

Возвращает текстовое представление схемы базы данных, очищенное от лишних пробелов и модификаторов `NOT NULL`. Используется для передачи в AI Service при генерации вопросов.

4.2.4 Клиент Supabase (`supabase_client.py`)

Модуль предоставляет набор функций для взаимодействия с Supabase через HTTP-запросы:

- `get_test_record(record_id)` — получает запись теста из таблицы `tests`.
- `get_db_file_url(record_id, db_filename)` — получает URL по ID теста из таблицы `tests` для загрузки файла из Storage.
- `download_db_file(url, destination)` — скачивает файл БД во временную директорию.

4.3 Детальная реализация AI Service

Сервис предоставляет единый эндпоинт для генерации вопросов.

4.3.1 Эндпоинт `/create_question` - генерация вопроса

Параметры: `schema: str` (текст схемы базы данных sqlite), `complexity: str` (уровень сложности 1–5).

Механизм генерации:

1. Схема БД передаётся в LangChain-цепочку.
2. Модель LLM получает системный промпт с подробной инструкцией, как интерпретировать уровни сложности, на что обращать внимание в схеме БД при генерации вопроса и т.д.
3. Запрос выполняется с использованием Structure Output, и ответ парсится в заранее заданном формате через Pydantic-модель `GeneratedQuestion`
4. Сгенерированный запрос валидируется на предмет синтаксических ошибок SQL
5. Сгенерированный запрос выполняется на БД, загруженной для теста
6. Если на проверках в пунктах 4 или 5 оказывается, что запрос не валидный, то генерация повторяется начиная с шага 3.

```
class GeneratedQuestion(BaseModel):
    thinking: str = Field(
        description="Разбор схемы БД, таблиц и колонок, план SQL-запроса"
    )
    question: str = Field(
        description="Вопрос на русском языке для студента"
    )
    correct_sql: str = Field(
        description="Правильный SQL SELECT-запрос"
    )
```

Листинг 3. Модель для структурированного вывода

Класс `ChatOpenAI` с параметром `with_structured_output` гарантирует соблюдение схемы ответа.

Настройка LLM:

```
def _make_llm():
    return ChatOpenAI(
        model=os.getenv("MODEL_NAME", "gpt-4o-mini"),
        base_url=os.getenv("OPENAI_BASE_URL"),
        api_key=os.getenv("OPENAI_API_KEY"),
    )

def _make_chain():
    llm = _make_llm().with_structured_output(GeneratedQuestion)
    return prompt | llm

async def generate_question(schema, complexity):
    chain = _make_chain()
    result = await chain.ainvoke({
        "schema": schema,
        "complexity": complexity
    })
    return {
        "question": result.question,
        "correct_sql": result.correct_sql,
        "thinking": result.thinking,
        "time": time.time() - start,
    }
```

Листинг 4. Настройка и запуск цепочки генерации

Системный промпт включает правила генерации: только SELECT-запросы, шкала сложности 1–5, валидный SQLite-синтаксис. Промпт пользователя содержит схему БД и желаемый уровень сложности.

Благодаря использованию LangChain и совместимого с OpenAI API формата, модель легко заменяется: достаточно изменить переменные окружения `MODEL_NAME`, `OPENAI_BASE_URL` и `OPENAI_API_KEY`. В проекте по умолчанию настроена локальная модель Qwen3:8b через Ollama, работающая по адресу `http://localhost:11434/v1`.

В продакшен-среде рекомендуется использовать либо облачных провайдеров, либо производить инференс языковой модели на отдельном сервере с GPU.

4.3.2 Структура базы данных Supabase

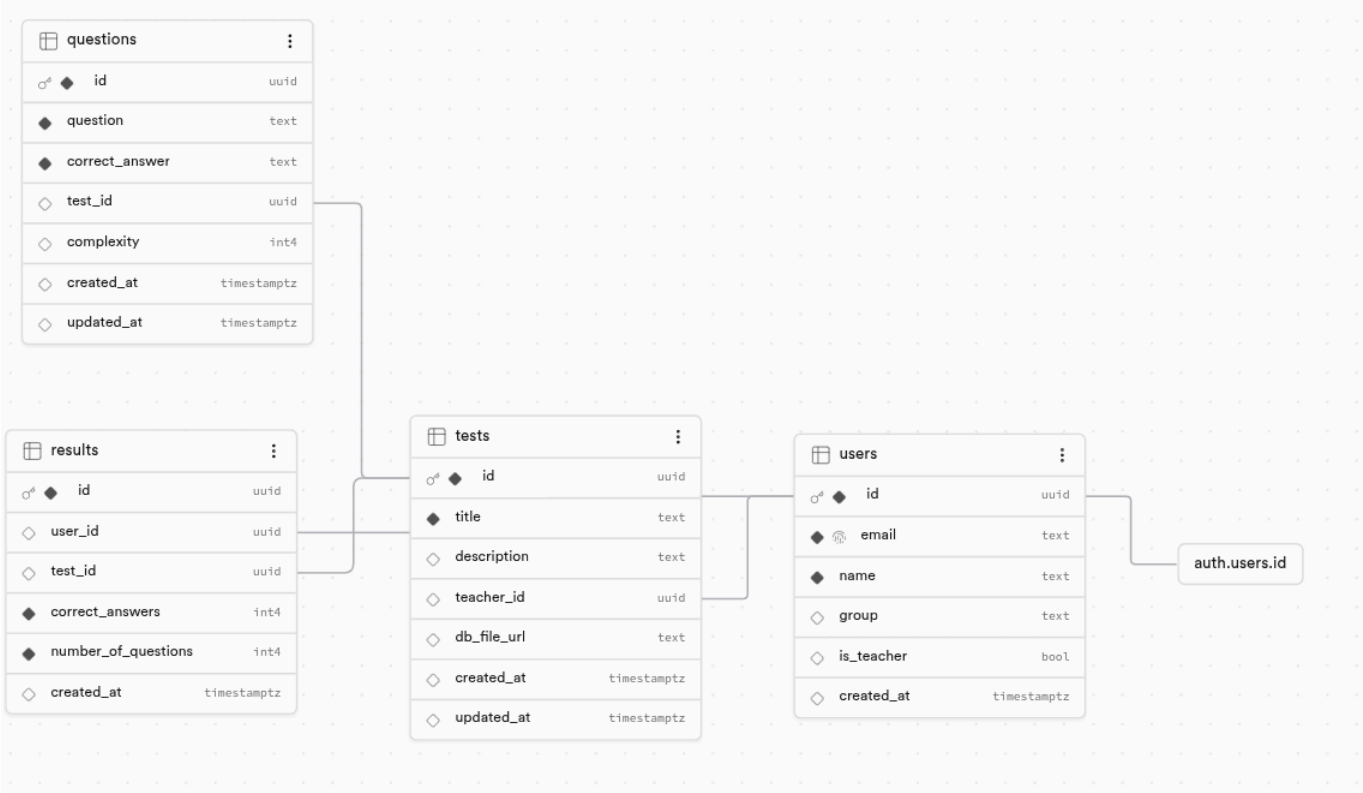

Рис. 5. Схема базы данных

Сервис использует следующие таблицы в Supabase (PostgreSQL):

Таблица `tests` - метаданные тестов:

- `id` (uuid, PK) - уникальный идентификатор.
- `title` (text) — название теста.
- `description` (text) — описание теста.
- `teacher_id` (uuid, FK -> auth.users.id) — идентификатор преподавателя-создателя.
- `db_file_url` (text) — URL к SQLite-файлу в Storage.
- `created_at` (timestamptz) — дата создания.

Таблица `questions` - вопросы теста:

- `id` (uuid, PK) — идентификатор вопроса.
- `test_id` (uuid, FK -> tests.id) — тест, к которому относится вопрос.
- `question` (text) — текст вопроса на русском языке.
- `correct_answer` (text) — правильный SQL-запрос.
- `complexity` (integer, 1–5) — уровень сложности.
- `created_at` (timestamptz) — дата создания.

Таблица `results` - результаты студентов:

- `id` (uuid, PK).
- `user_id` (uuid, FK -> auth.users.id) — студент.
- `test_id` (uuid, FK -> tests.id) — тест.
- `correct_answers` (integer) — количество правильных ответов.
- `number_of_questions` (integer) — общее количество вопросов.
- `created_at` (timestamptz) — дата и время сдачи.

Таблица `users` - профили пользователей:

- `id` (uuid, PK, FK -> auth.users.id).
- `email` (text) — email пользователя.
- `name` (text) — отображаемое имя.
- `group` (text) — учебная группа (для студентов).
- `is_teacher` (boolean) — флаг роли (true = преподаватель, false = студент).

Хранилище `db-files` предназначено для SQLite-файлов, загруженных преподавателями. Файлы организуются по произвольным именам, а ссылка на файл сохраняется в поле `db_file_url` таблицы `tests`.

4.4 Реализация клиентской части (Frontend)

4.4.1 Маршрутизация и навигация

Приложение использует Vue Router [21] с пятью основными маршрутами:

- `/` — главная страница со списком тестов;
- `/register` — регистрация;
- `/login` — вход;
- `/test/:testId` — страница управления тестом (для преподавателя);
- `/test/solve/:testId` — страница решения теста (для студента).

В работе реализован Navigation-guard — это механизм, который проверяет статус текущего пользователя и запрещает доступ к определенным маршрутам. В частности, проверяет статус аутентификации и перенаправляет неавторизованных пользователей на страницу входа.

4.4.2 Аутентификация и управление состоянием

Аутентификация реализована через Supabase Auth. Vuex [22] store управляет состоянием текущего пользователя. При старте приложения store проверяет наличие сессии в localStorage браузера и при необходимости восстанавливает её.

Сервис `auth.js` (`services/auth.js`) предоставляет функции:

- `getSession()` — получение текущей сессии;
- `signIn(email, password)` — вход;
- `signUp(email, password, userData)` — регистрация;
- `signOut()` — выход;

4.4.3 Компонентная архитектура

Ключевые компоненты Vue:

- `TheHeader.vue` — шапка с логотипом, кнопками входа/регистрации, выпадающим меню пользователя.
- `CreateTestBtn.vue` и `EditTestBtn.vue` — диалоги создания и редактирования теста с загрузкой .db файла в Supabase Storage.
- `TestCard.vue` — карточка теста: отображает название, описание, статус прохождения (для студента), кнопки редактирования/удаления (для преподавателя).

- `QuestionCard.vue` — карточка вопроса с отображением текста, правильного ответа, индикатора сложности, возможностью удаления и редактирования.
- `CreateQuestionBtn.vue` — диалог создания вопроса: ручной ввод или генерация через AI. Поле сложности реализовано через слайдер (v-slider).
- `SqlEditor.vue` — Поле ввода для SQL-кода с поддержкой подсветки синтаксиса и нумерации строк.
- `ResponseTable.vue` — таблица результатов SQL-запроса.
- `ComplexityLine.vue` — 5-сегментный индикатор сложности, который отображается в интерфейсе студента.

4.4.4 Интеграция с бэкендом

Модуль `api/index.js` создаёт два экземпляра axios [23]: `apiClientUserDB` (для работы с сервисом SQLite Query Service) и `apiClientAI` (для работы с сервисом AI Service). Модуль экспортирует два объекта: `user_db_api` и `ai_api`.

Объект `user_db_api` предоставляет методы:

`user_db_api.query_to_user_db(test_id, query)` - используется для выполнения SQL-кода

`user_db_api.create_er_diagram(test_id)` - возвращает ER-диаграмму, закодированную в base64

`user_db_api.get_schema(test_id)` - возвращает текст схемы базы данных

Объект `ai_api` предоставляет метод:

`ai_api.create_question(schema, complexity)` - используется для генерации вопроса на основании схемы базы данных.

5 ОПИСАНИЕ ИНТЕРФЕЙСА ПОЛЬЗОВАТЕЛЯ

5.1 Страница аутентификации

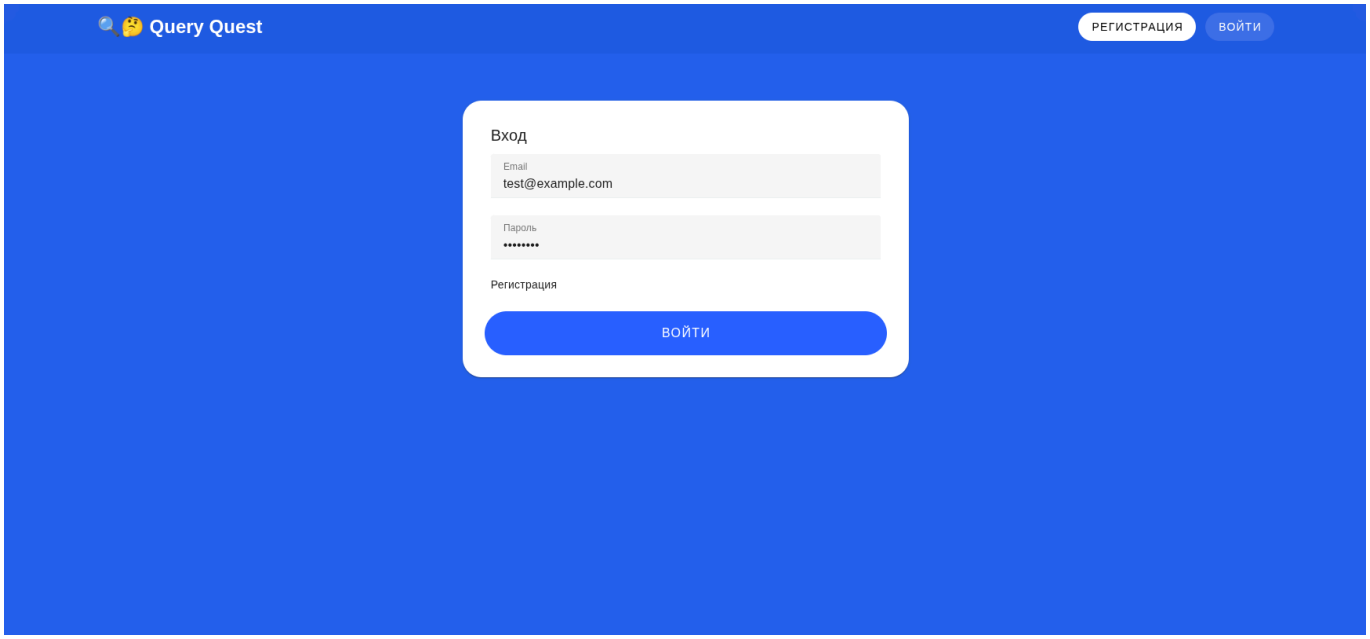

Рис. 6. Скриншот страницы входа

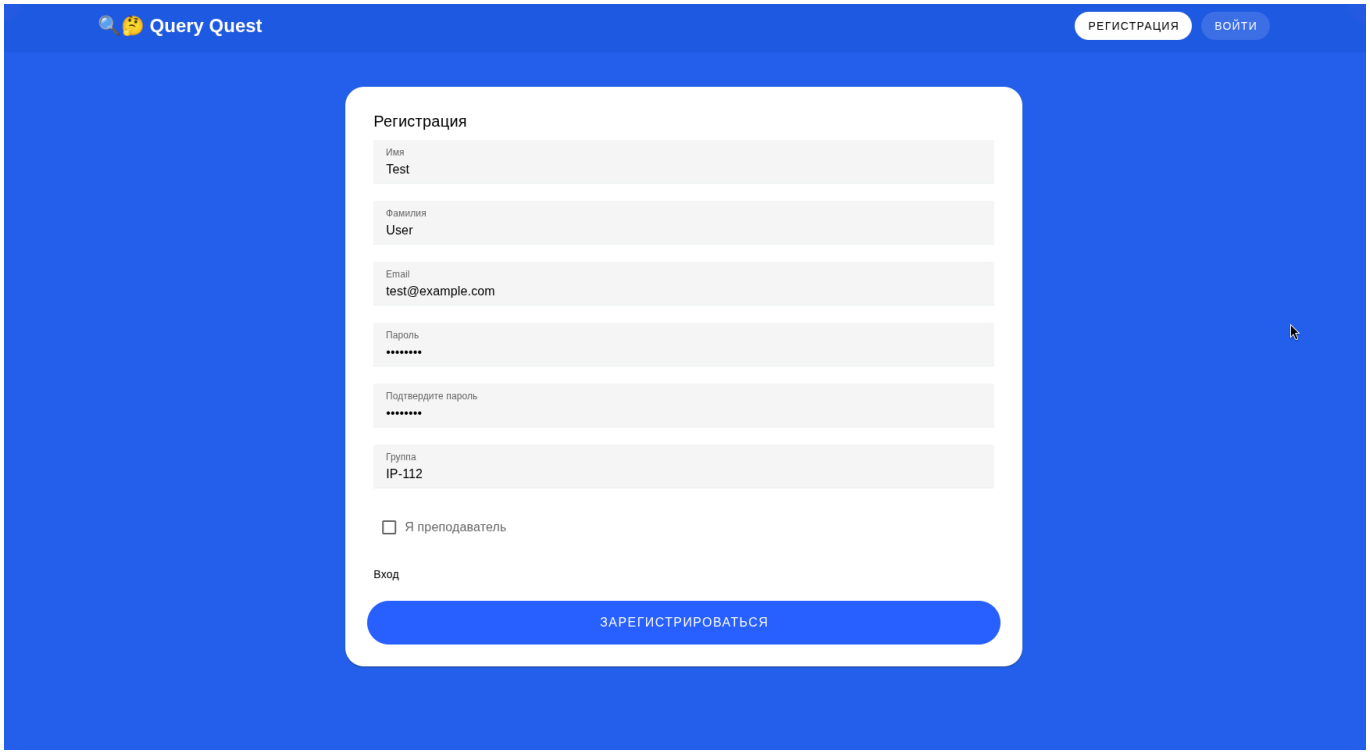

Рис. 7. Скриншот страницы регистрации

Страницы регистрации и входа реализованы по гайдлайнам дизайн системы Google Material Design 3 с использованием Vuetify-компонентов. Формы содержат поля email и пароль, а также переключатель роли при регистрации. После успешного входа пользователь перенаправляется на главную страницу.

5.2 Главная страница (список тестов)

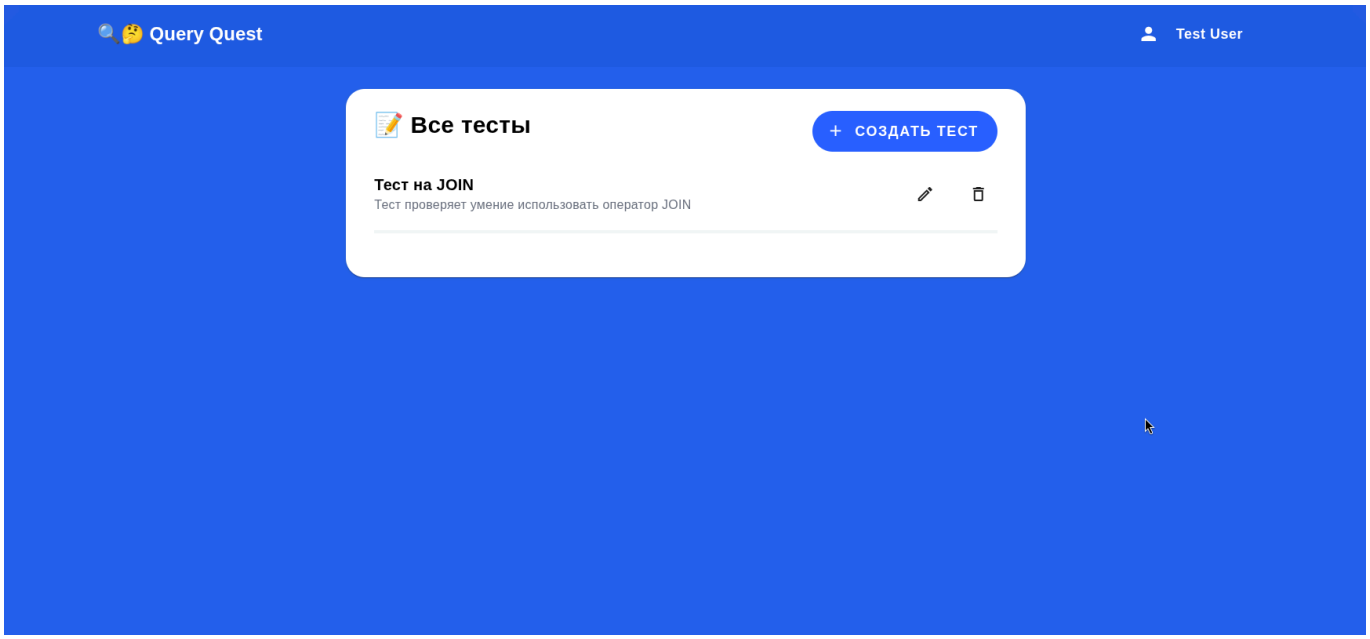

Рис. 8. Скриншот главной страницы от лица преподавателя

На главной странице отображается список всех доступных тестов. Для преподавателя показываются только его собственные тесты с возможностью редактирования названия, описания и удаления. Для студента отображаются все тесты с индикацией результата прохождения при наличии сохранённого результата.

Кнопка «Создать тест» доступна только преподавателям.

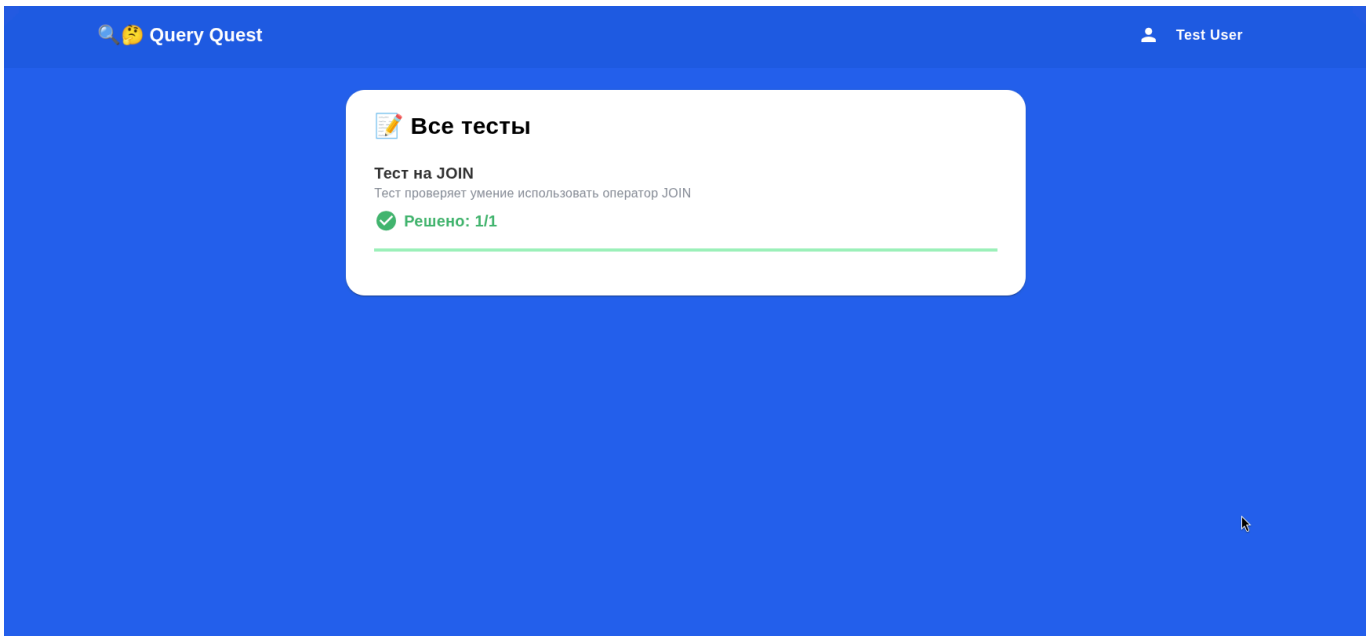

Рис. 9. Скриншот главной страницы от лица студента

5.3 Страница управления тестом (преподаватель)

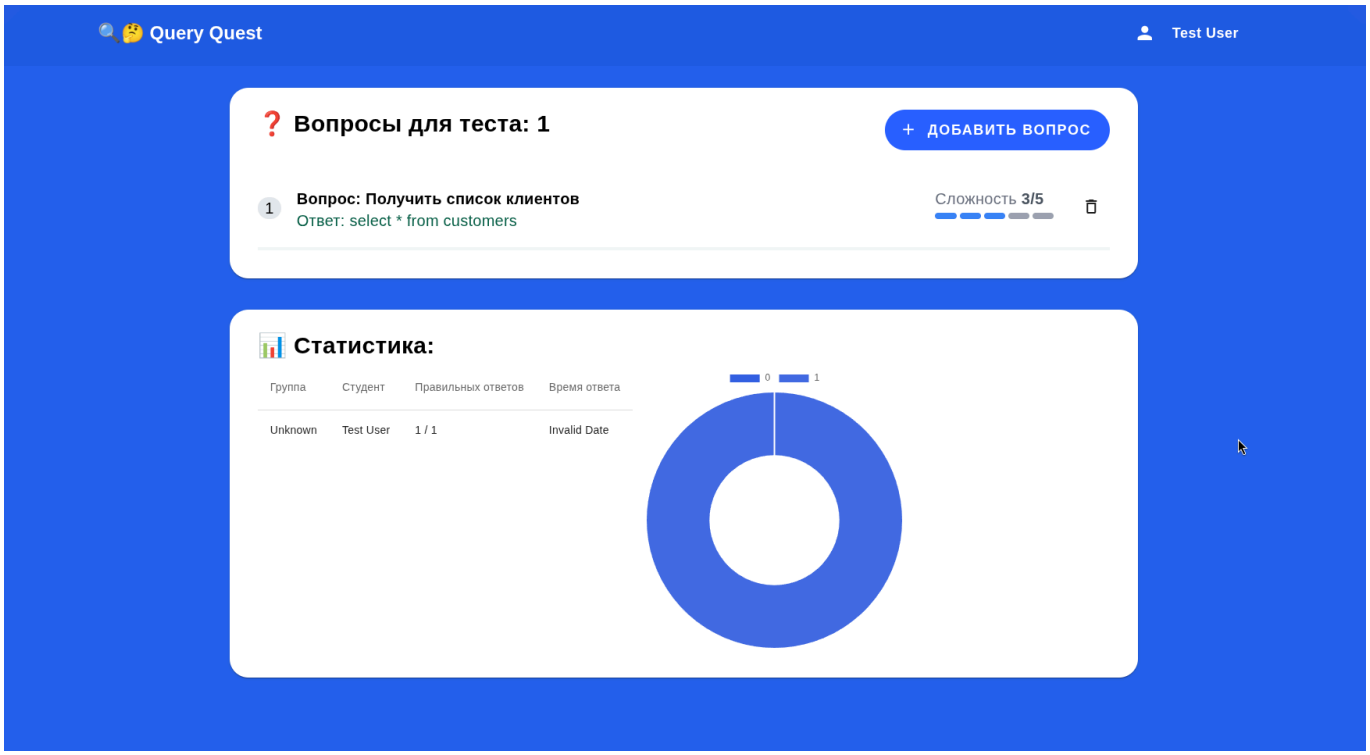

Рис. 10. Скриншот страницы управления тестом (преподаватель)

В нижней части страницы находится блок статистики, в котором отображаются все прохождения выбранного теста. Для каждого прохождения отображается группа студента, его имя и фамилия, кол-во правильных ответов и время когда был пройден тест.

Справа от таблицы показывается диаграмма pie-chart, которая показывает распределение, сколько студентов решили верно каждое возможное кол-во заданий в тесте.

Страница содержит список вопросов в тесте, для каждого вопроса отображается корректный SQL-ответ и индикатор сложности вопроса. По клику на карточку с вопросом открывается модальное окно с возможностью отредактировать вопрос, ответ и изменить уровень сложности. Справа от индикатора сложности располагается кнопка для удаления вопроса из текущего теста.

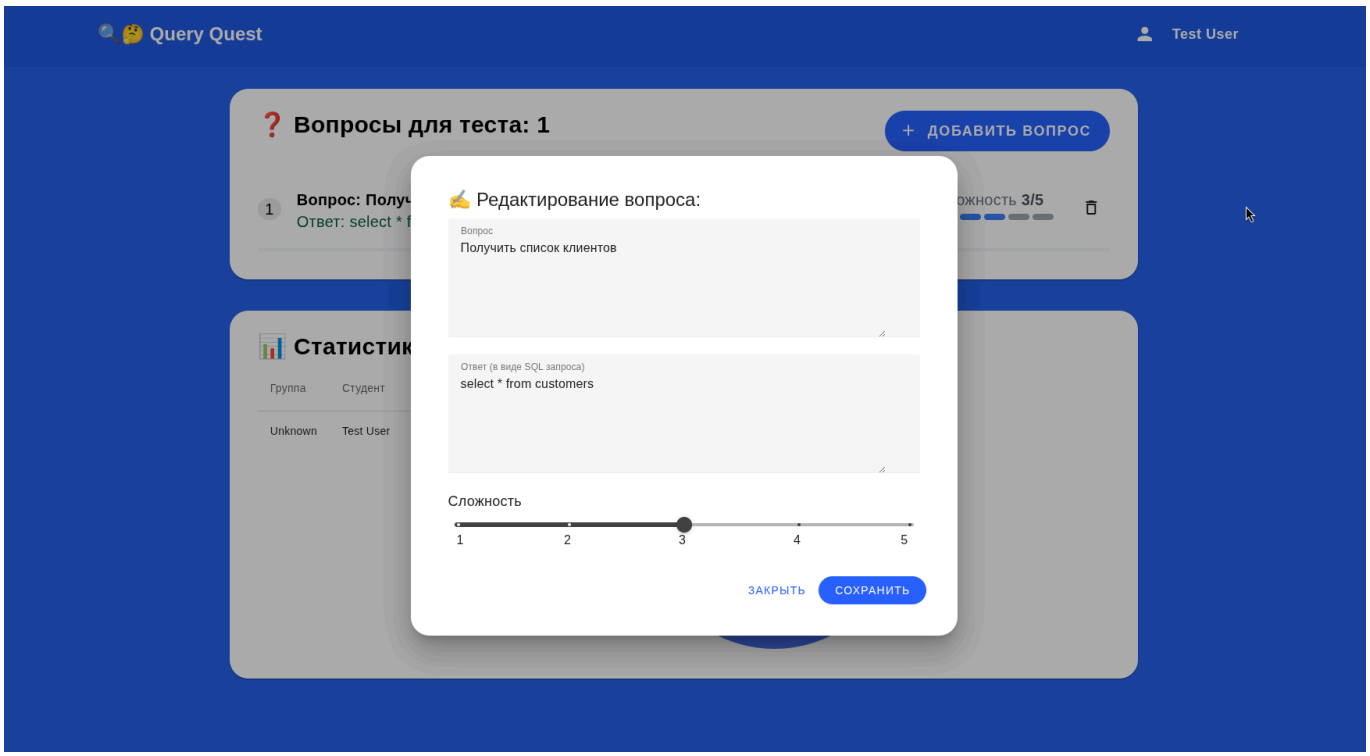

Рис. 11. Модальное окно редактирования вопроса

Кнопка «Добавить вопрос» открывает модальное окно, в котором преподавателю предлагается вписать формулировку вопроса, указать SQL запрос, который корректно выполняет поставленную в вопросе задачу и указать уровень сложности с помощью слайдера.

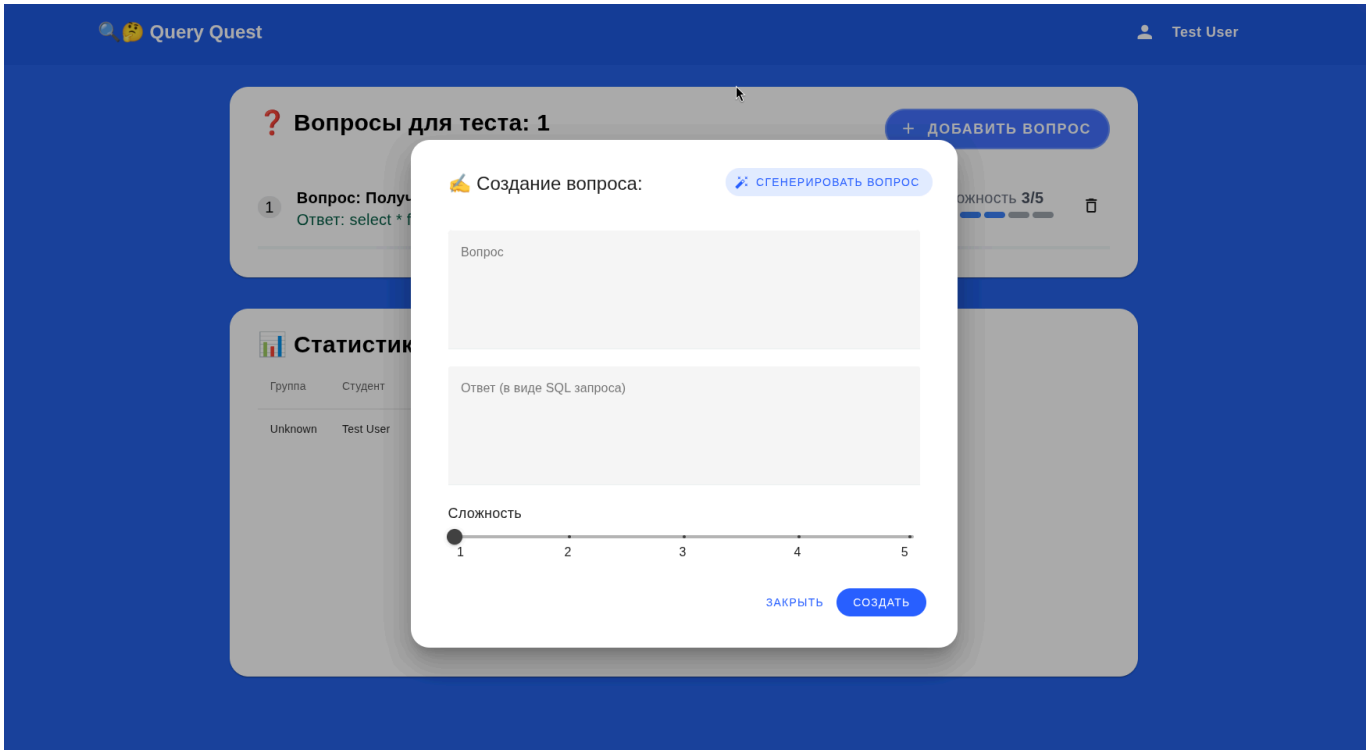

Рис. 12. Модальное окно создания вопроса

В этом же модальном окне есть кнопка «Сгенерировать вопрос», которая обращается к LLM. Сложность сгенерированого вопроса будет напрямую зависеть от того какое значение указал пользователь с помощью слайдера.

5.4 Страница решения теста (студент)

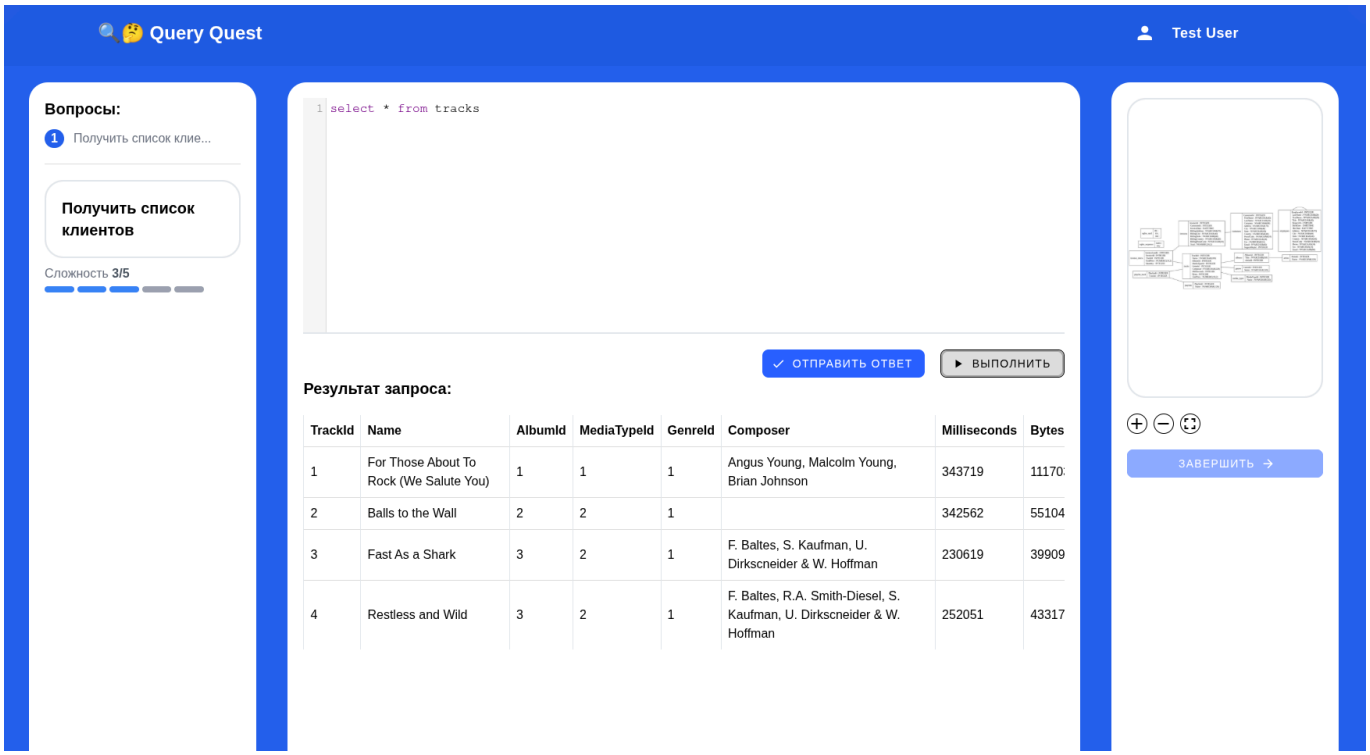

Рис. 13. Скриншот страницы решения теста от лица студента

Страница решения имеет трёхколоночный макет:

Левая колонка — это список вопросов с номерами. Каждый вопрос отображает статус проверки: зелёная галочка — правильно, красный крестик — неправильно. При нажатии на вопрос в центральной панели отображается его текст и сложность.

Центральная колонка - это редактор SQL-запросов на основе CodeMirror [24] с подсветкой синтаксиса и нумерацией строк. Под редактором расположены:

- Кнопка «Выполнить», которая отправляет запрос на `/sql` сервис, результат отображается в таблице `ResponseTable`.
- Кнопка «Отправить ответ» — сравнивает результат студента с эталонным (путём сравнения JSON-представления результирующих наборов данных). Доступна только один раз для каждого вопроса.
- Прогресс-бар при выполнении sql запроса.
- Сообщения об ошибках выполнения SQL.

Правая колонка содержит ER-диаграмму текущей базы данных, отображаемую через компонент `v-zoomer` с возможностью масштабирования и просмотра в полноэкранном режиме. Также этот блок интерфейса содержит кнопку «Завершить», которая становится активной только после отправки ответов на все вопросы.

При завершении результаты сохраняются в таблицу `results`, и студент перенаправляется на главную страницу.

6 ЗАКЛЮЧЕНИЕ

В результате преддипломной практики был разработан веб-сервис для интерактивного обучения студентов языку SQL.

В ходе выполнения работы были решены следующие задачи:

1. Проведён анализ предметной области и существующих аналогов, выявлены их ограничения в контексте образовательного процесса.
2. Сформулированы функциональные требования к системе и к интерфейсу пользователя.
3. Выбран и обоснован стек технологий.
4. Разработана архитектура системы, включающая два независимых микросервиса.
5. Реализованы все ключевые эндпоинты: выполнение SQL-запросов, генерация ER-диаграмм, получение схемы БД, генерация вопросов через LLM.
6. Разработан пользовательский интерфейс на Vue 3 с поддержкой двух ролей пользователей (преподаватель, студент).
7. Реализована интеграция с LLM, обеспечивающая автоматическую генерацию вопросов на основе анализа схемы любой загруженной базы данных.

В результате система готова к использованию в учебном процессе. Автоматическая генерация вопросов позволяет преподавателям создавать разнообразные тестовые задания, существенно сокращая время на подготовку материалов. Визуализация схем данных в виде ER-диаграмм помогает студентам быстрее понимать структуру базы данных. Механизм автоматической проверки через сравнение результирующих наборов данных обеспечивает объективную оценку навыков студента.

Перспективы дальнейшего развития включают: расширение типов генерируемых заданий (INSERT, UPDATE, DELETE), добавление групповой аналитики для потоков студентов, интеграцию с системами управления обучением.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. U.S. Bureau of Labor Statistics. Occupational Outlook Handbook: Data Scientists. Таблица 6: Employment change and projected growth [Электронный ресурс]. 2025. URL: https://www.bls.gov/ooh/math/data-scientists.htm#tab-6 (дата обращения: 14.05.2025).
2. Stack Overflow. Developer Survey 2023 [Электронный ресурс]. 2023. URL: https://survey.stackoverflow.co/2023 (дата обращения: 14.05.2025).
3. Verougstraete R. What Job Postings Say About Demand for SQL [Электронный ресурс]. Lightcast, 2021. URL: https://lightcast.io/resources/blog/what-job-postings-say-about-demand-for-sql (дата обращения: 14.05.2025).
4. Мейкшан В.И. Основы языка SQL в примерах и задачах. Новосибирск: СибГУТИ, 2013. С. 41.
5. Taipalus T., Seppänen V. SQL and database education: обзор рекомендаций по практическому обучению и будущая повестка исследований [Электронный ресурс]. ResearchGate, 2020. URL: https://www.researchgate.net/publication/342759889_SQL_Education_A_Systematic_Mapping_Study_and_Future_Research_Agenda (дата обращения: 14.05.2025).
6. Supabase. Supabase Documentation [Электронный ресурс]. 2025. URL: https://supabase.com/docs (дата обращения: 20.05.2025).
7. Ramirez S. FastAPI — документация [Электронный ресурс]. 2025. URL: https://fastapi.tiangolo.com/ (дата обращения: 20.05.2025).
8. Encode. Starlette — документация [Электронный ресурс]. 2025. URL: https://www.starlette.io/ (дата обращения: 20.05.2025).
9. Pydantic. Pydantic — документация [Электронный ресурс]. 2025. URL: https://docs.pydantic.dev/latest/ (дата обращения: 20.05.2025).
10. Encode. Uvicorn — документация [Электронный ресурс]. 2025. URL: https://www.uvicorn.org/ (дата обращения: 20.05.2025).
11. Vue.js. Vue.js 3 — официальное руководство [Электронный ресурс]. 2025. URL: https://vuejs.org/guide/introduction.html (дата обращения: 20.05.2025).
12. Hiltgen D. ollama/docs [Электронный ресурс]. 2025. URL: https://github.com/ollama/ollama/tree/main/docs (дата обращения: 14.05.2025).
13. OpenAI. OpenAI API Reference [Электронный ресурс]. 2025. URL: https://platform.openai.com/docs/api-reference (дата обращения: 20.05.2025).
14. Team Q. Qwen 3: Think Deeper, Act Faster [Электронный ресурс]. 2025. URL: https://qwenlm.github.io/blog/qwen3/ (дата обращения: 14.05.2025).
15. LangChain. LangChain — документация [Электронный ресурс]. 2025. URL: https://python.langchain.com/docs/introduction/ (дата обращения: 20.05.2025).
16. Tailwind Labs. Tailwind CSS — документация [Электронный ресурс]. 2025. URL: https://tailwindcss.com/docs/installation (дата обращения: 20.05.2025).
17. Vuetify. Vuetify 3 — официальная документация [Электронный ресурс]. 2025. URL: https://vuetifyjs.com/en/getting-started/installation/ (дата обращения: 20.05.2025).
18. Google. Material Design 3 [Электронный ресурс]. 2025. URL: https://m3.material.io/ (дата обращения: 20.05.2025).
19. Python Software Foundation. sqlite3 — DB-API 2.0 interface for SQLite databases [Электронный ресурс]. 2025. URL: https://docs.python.org/3/library/sqlite3.html (дата обращения: 20.05.2025).

20. Carrera E. pydot — Python interface to Graphviz [Электронный ресурс]. 2025. URL: https://github.com/pydot/pydot (дата обращения: 20.05.2025).
21. Vue.js. Vue Router — официальное руководство [Электронный ресурс]. 2025. URL: https://router.vuejs.org/guide/ (дата обращения: 20.05.2025).
22. Vue.js. Vuex — официальное руководство [Электронный ресурс]. 2025. URL: https://vuex.vuejs.org/guide/ (дата обращения: 20.05.2025).
23. Axios. Axios — документация [Электронный ресурс]. 2025. URL: https://axios-http.com/docs/intro (дата обращения: 20.05.2025).
24. CodeMirror. CodeMirror 6 — документация [Электронный ресурс]. 2025. URL: https://codemirror.net/docs/ (дата обращения: 20.05.2025).