МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ

КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ

ордена Трудового Красного Знамени федеральное государственное

бюджетное образовательное учреждение высшего образования

**«Московский технический университет связи и информатики»**

Кафедра «Информатика»

Курсовая работа

**«Разработка базы данных и её подключение к веб-сайту конструкторского бюро KURS»**

по дисциплине

**Базы данных**

|  |  |
| --- | --- |
|  | Выполнил: студент гр. БЭИ2202  Кулешов А.С.  Проверил: ст. преп. Юсков И. О., к.т.н., доцент кафедры «Информатика» Шакин В. Н. |

Москва, 2023 г.

Содержание

[Введение 3](#_Toc153838577)

[Техническое задание 5](#_Toc153838578)

[1 Теоретическая часть 7](#_Toc153838579)

[1.1 Выбор базы данных 7](#_Toc153838580)

[2 Реализация базы данных 11](#_Toc153838581)

[3 Взаимодействие базы данных с сайтом 14](#_Toc153838582)

[3.1 Пользователи. Аутентификация и авторизация. Изменение данных пользователя 14](#_Toc153838583)

[3.2 Заказанные проекты. Заявки клиентов. 16](#_Toc153838584)

[3.3 Полный доступ к БД для Администраторов 17](#_Toc153838585)

[Заключение 18](#_Toc153838586)

[Список использованных источников 19](#_Toc153838587)

Введение

Современные технологии проникают во все сферы нашей жизни, в том числе и в индустрию архитектуры. Поставки материалов, процесс создания дизайна, покупка, продажа, обратная связь – всё это требует информацию, коммуникацию между участниками архитектурного процесса. Всю информацию хранить в памяти людей невозможно, а переносить на бумагу крайне трудоёмко и непродуктивно. Нужда хранить данные в цифровом формате **актуальна как никогда**. Для этого и будет использована база данных.

Во-вторых, **неоспоримую актуальность** **проблемы** демонстрирует внедрение механизма регистрации пользователя, проверки состояния готовности архитектурного проекта. Благодаря базе данных владельцы бюро смогут эффективно управлять проектами, оптимизировать распределение времени и ресурсов, уменьшить траты и время клиентов.

В-третьих, это **подтверждает актуальность проблемы** в обеспечении безопасности данных. База данных позволяет внедрить меры по защите личной информации клиентов, что становится все более важным в контексте растущей цифровизации и повышенного внимания к вопросам конфиденциальности.

В данной курсовой работе исследуется процесс разработки веб-приложения для архитектурного бюро, который позволит оптимизировать ряд процессов, выполняемых бюро. Проект включает в себя внедрение базы данных, что позволит собирать и хранить информацию о клиентах, их заказах, а также информацию о работниках бюро.

В работе будет рассмотрена архитектура сайта, разработка функциональности для регистрации и аутентификации пользователей, система просмотра состояние готовности проекта клиента, управление штатом сотрудников. Внедрение базы данных позволит эффективно управлять этой информацией, обеспечивая удобное и безопасное взаимодействие клиентов с архитектурным бюро. Вместе с тем будут рассмотрены аспекты безопасности данных и конфиденциальности пользователей, а также возможности для будущего расширения функционала приложения.

Подводя итоги, **цель этой работы** — упростить процесс коммуникации между клиентами и владельцами бюро. Владельцам бюро также смогут более эффективно управлять своим бизнесом.

**Объект исследования** — процесс создания информационной системы с использованием базы данных для упрощения работы с клиентами.

**Предмет исследования** — процесс разработки архитектуры базы данных, подходящих и производительных запросов, а также процесс усиления безопасности в базе данных.

Техническое задание

Целью проекта является расширение функциональности веб-сайта архитектурного бюро «KURS» с помощью добавления базы данных для хранения информации о сотрудниках, проектах, клиентах и их ассоциированных данных.

В настоящий момент веб-сайт «KURS» является статическим и предоставляет посетителям постоянную информацию о бюро. Проект заключается в добавлении базы данных для хранения следующей информации:

1. информация о сотрудниках,
2. информация о пользователях и их профилях,
3. информация о проектах.

**Функциональные требования**

Создание базы данных и реализация следующего функционала:

1. добавление информация о проектах;
2. Аутентификация и авторизация клиентов, а также редактирование его профиля;
3. добавление, редактирование и просмотр проекта, ассоциированного с клиентом.

**Требования к пользовательскому интерфейсу**

Интерфейс должен быть интуитивно понятным и удобным для администраторов сайта и пользователей, позволяя им управлять данными.

**Язык программирования**

Проект может быть реализован с использованием любого языка программирования.

**Система управления базами данных**

По ситуации и в зависимости от требований проекта, оставляется право выбора СУБД. Разработчик должен обосновать свой выбор СУБД, учитывая специфику проекта и его технические характеристики.

**Защита данных**

Обеспечить безопасность данных, включая защиту от SQL-инъекций и других атак.

# 1 Теоретическая часть

Для работы над проектом необходимо определить данные, которые база данных будет содержать. Для конструкторского бюро можно выделить основной функционал:

- Пользователи

1) Информация о пользователи (ФИО, дата рождения и т.п.)

2) Возможность редактировать информацию из пункта (1)

3) Просматривать проекты, связанные с пользователем

- Проекты

1) Информация о проекте (дата начала, предполагаемая дата окончания)

2) Проект должен быть корректно связан с нужным пользователем верной связью (как заказчик или как исполнитель)

- Работники

1) Информация о работнике (ФИО, дата рождения и т.п.)

2) Дополнительная информация, связывающая работника с компанией

3) Возможность просматривать и редактировать проекты, с которыми работник связан

База данных должна будет хранить в себе эти сущности: пользователи, проекты, работники и связи между ними.

## 1.1 Выбор базы данных

Для определения подходящей СУБД для проекта был проведён сравнительный анализ PostgreSQL, MySQL, Oracle, Microsoft Access и Sqlite:

* PostgreSQL:
  1. Открытый исходный код: PostgreSQL является открытой и свободной СУБД, что обеспечивает бесплатный доступ к исходному коду и высокую гибкость в изменении системы под конкретные нужды.
  2. Мощность и расширяемость: PostgreSQL поддерживает сложные SQL-запросы, транзакции, хранимые процедуры и имеет широкий набор встроенных типов данных. Он также позволяет создавать пользовательские функции и расширения.
  3. Надежность и целостность данных: PostgreSQL обеспечивает высокую степень целостности данных и надежности благодаря транзакционной модели и механизмам восстановления после сбоев.
  4. Поддержка JSON и геоданных: Встроенная поддержка JSON-типа данных и геоданных делает PostgreSQL подходящим для разнообразных приложений, включая веб-разработку и анализ больших данных.
* MySQL:
  1. Быстрота и производительность: MySQL известен своей высокой производительностью, особенно в сценариях с частым чтением данных.
  2. Простота использования: MySQL часто выбирают за его простоту использования и относительно низкую стоимость внедрения.
  3. Широкая поддержка: имеется обширное сообщество пользователей MySQL и большое количество сторонних инструментов и библиотек.
  4. Ограниченные возможности для сложных запросов: в некоторых случаях MySQL может оказаться менее мощным для сложных запросов по сравнению с PostgreSQL.
* Oracle:
  1. Масштабируемость и производительность: Oracle часто используется в крупных предприятиях благодаря своей высокой производительности и возможностям масштабирования.
  2. Расширенные функциональные возможности: Oracle предоставляет множество расширенных функций, таких как управление табличными пространствами, аналитические функции и многие другие.
  3. Высокие затраты: Oracle часто сопряжен с высокими затратами на лицензии и обслуживание, что может быть фактором, ограничивающим для малых и средних предприятий.
* Microsoft Access:

1. Легкость использования: Microsoft Access предоставляет простой способ создания баз данных, особенно для небольших проектов.
2. Ограниченные возможности: Access может быть ограничен в производительности и масштабируемости, что делает его менее подходящим для крупных и сложных приложений.
3. Основан на файле: Access использует файловую структуру, что может вызвать проблемы с одновременным доступом нескольких пользователей и безопасностью данных.

* Sqlite:

1. Открытый исходный код: Sqlite является открытой и свободной СУБД, что обеспечивает бесплатный доступ к исходному коду и высокую гибкость в изменении системы под конкретные нужды.
2. Дисковая архитектура, которая обеспечивает надёжность и высокую производительность
3. Простота резервного копирования и подключения: СУБД занимает лишь 1 файл
4. Лёгкость в работе с ЯП Python

Так как разрабатываемая система не требует больших мощностей, не имеет крайне сложной структуры, её поддержка не должна иметь больших затрат, но её реализация должна на основе уже построенных частей позволять расширять систему и реализовывать новые функции, то в качестве СУБД была выбрана Sqlite, благодаря отсутствию стоимости обслуживания, высококлассному уровню безопасности, удобству в копировании а также огромному количеству расширений, возможностей, а также доступных исходных программных решений на ЯП Python что позволяет выполнить любую предназначенную для СУБД задачу без особых затрат, а реализацию системы быстрой и дешёвой.

# 2 Реализация базы данных

Для реализации базы данных было принято решение создать 4 таблицы: «users», «employees», «jobs», «projects», «ююю»

Запрос для создания таблицы «users»:

CREATE TABLE users (

id INTEGER PRIMARY KEY AUTOINCREMENT,

name VARCHAR(50),

email VARCHAR(50),

login VARCHAR(50) NOT NULL,

password VARCHAR(50) NOT NULL,

status VARCHAR(50)

);

Как можно заметить, лишь поля «login» и «password» имеют ключевое слово «NOT NULL», т.к. остальные данные пользователь может заполнить уже после регистрации аккаунта.

Запрос для создания таблицы «employees»:

CREATE TABLE employees(

id INTEGER PRIMARY KEY AUTOINCREMENT, *-- код сотрудника*

name VARCHAR(50) NOT NULL, *-- Имя*

BirthDate DATE NOT NULL,

gender VARCHAR(10) NOT NULL,

phone VARCHAR(15),

job\_id INTEGER NOT NULL

);

Поле «job\_id» ссылается на должность, занимаемую сотрудником. Лишь поле «phone» является необязательным, а также является пустым, если у сотрудника нет контактного телефона.

Запрос для создания таблицы «jobs»:

CREATE TABLE jobs (

id INTEGER PRIMARY KEY AUTOINCREMENT,

name VARCHAR(50) NOT NULL,

salary INTEGER NOT NULL

);

У компании существует политика одинаковой заработной платы для сотрудников, находящихся на одинаковых должностях.

Запрос для создания таблицы «projects»:

CREATE TABLE projects(

id SERIAL PRIMARY KEY,

user\_id INTEGER NOT NULL, *-- Пользователь, который оставил заявку*

name VARCHAR(50) NOT NULL, *-- Имя*

SignUpDate DATE NOT NULL, *-- Дата подачи заявки*

status VARCHAR(50) NOT NULL, *-- pending(в ожидании рассмотрения компанией\\ оплаты\\ прочих бюрократических условностей), in\_action(в процессе строительства\\проектирования), finished(закончено)*

FinishDate DATE NOT NULL *-- Дата конца делания проекта (в случае ожидания - 1 Января 1970, в случае "в процессе", то предположительная дата, в случае законченного проекта - дата фактического завершения проекта)*

);

 Запрос для создания таблицы «project\_to\_employee\_linker»:

CREATE TABLE project\_to\_employee\_linker(

id INTEGER PRIMARY KEY AUTOINCREMENT,

project\_id INTEGER NOT NULL,

employee\_id INTEGER NOT NULL

);

Комментарии к каждому полю представлены в комментариях SQL-кода. Таблица «project\_to\_employee\_linker» является служебной, чтобы связать таблицы «projects» и «employees» связью много-к-много.

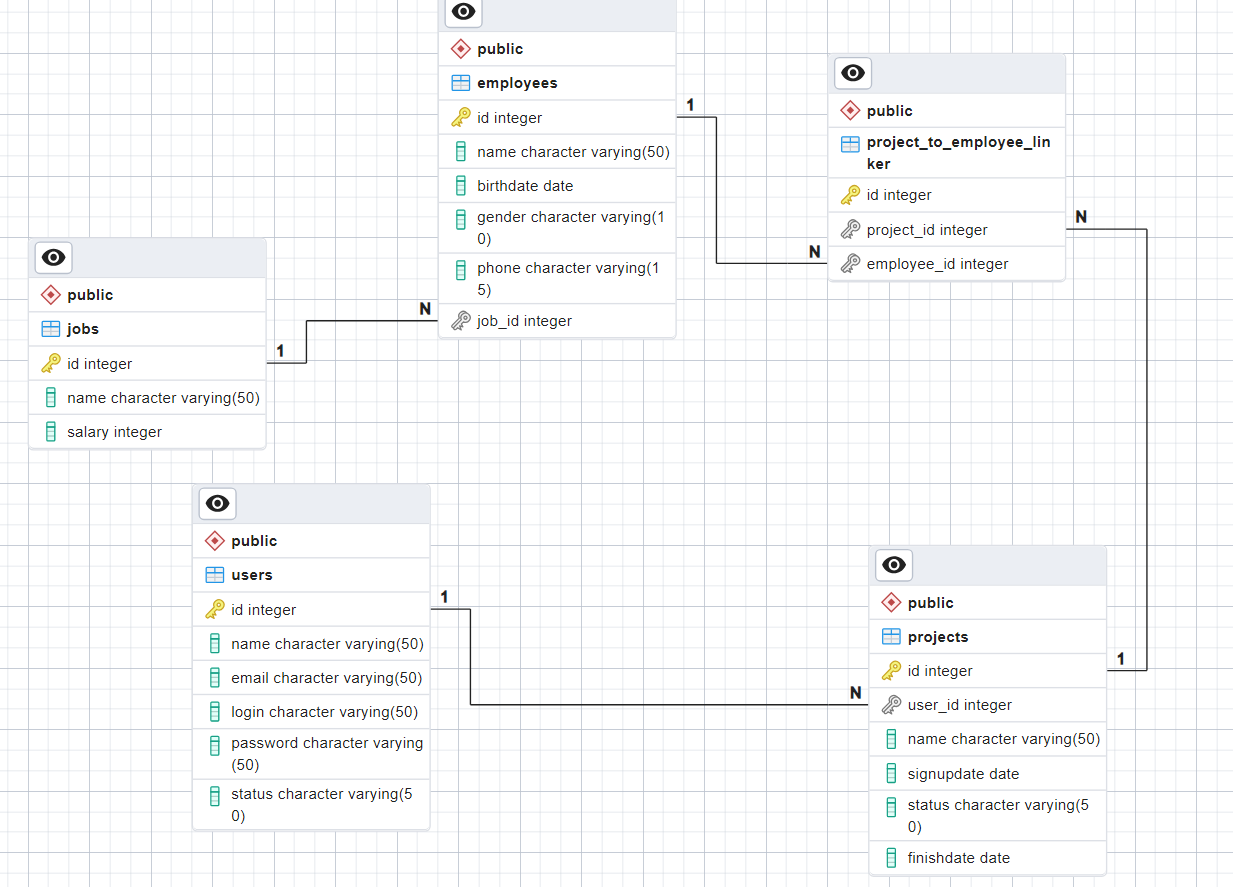


Рисунок 1 — ER-диаграмма полученной базы данных

# 3 Взаимодействие базы данных с сайтом

## 3.1 Пользователи. Аутентификация и авторизация. Изменение данных пользователя

Авторизация происходит с помощью формы авторизации, показанной на рисунке 2.



Рисунок 2 — Форма авторизации

Эта форма делает запрос на сервер, сравнивает результат с данными в базе данных и возвращает результат авторизации.

if (request.form.\_\_contains\_\_('submit\_login')):  
 # login  
  
 res = run\_script\_get(f"SELECT \* FROM users WHERE login='{request.form['username']}' AND password = '{request.form['pwd']}'")  
 print(res)  
  
 if len(res) == 0:  
 flash('Неверный пароль или логин', 'error')  
 return lk('get')  
  
  
 flash('Вы были успешно авторизированы')  
 session['username'] = request.form['username']  
  
if (request.form.\_\_contains\_\_('submit\_register')):  
 # register  
  
 res = run\_script\_get(f"SELECT \* FROM users WHERE login='{request.form['username']}'")  
 print(res)  
  
 if len(res) != 0:  
 flash('Такой пользователь уже есть в ДБ. Выберите другой логин', 'error')  
 return render\_template('lk.html')  
  
  
 req\_str = f"""INSERT INTO users (login, password)\nVALUES ({request.form['username']}, {request.form['pwd']})"""  
 print(req\_str)  
 res = run\_script\_write(f"""INSERT INTO users (login, password, status)\n  
 VALUES ('{request.form['username']}', '{request.form['pwd']}', 'user')""")  
  
 session['username'] = request.form['username']  
 flash('Вы были успешно зарегестрированы')  
  
return lk('get')

После авторизации пользователь может также поменять свои данные, используя соответствующие поля для ввода данных

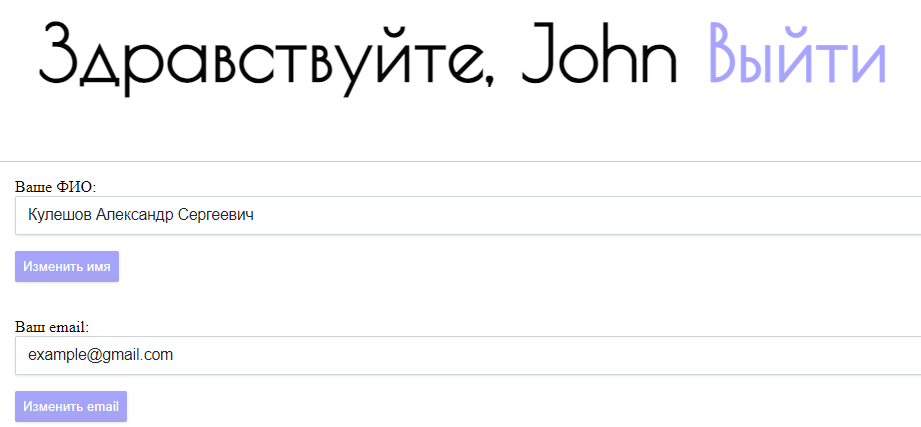


Рисунок 3 — Форма изменения данных

if (request.form.\_\_contains\_\_('name\_change')):  
 res = run\_script\_write(  
 f"UPDATE users SET name = '{request.form['name\_change']}' WHERE login='{session['username']}' ")  
  
if (request.form.\_\_contains\_\_('email\_change')):  
 res = run\_script\_write(  
 f"UPDATE users SET email = '{request.form['email\_change']}' WHERE login='{session['username']}' ")

## 3.2 Заказанные проекты. Заявки клиентов.

Клиент может посмотреть, заказанные им проекты, узнать примерную дату выполнения и статус заявки.

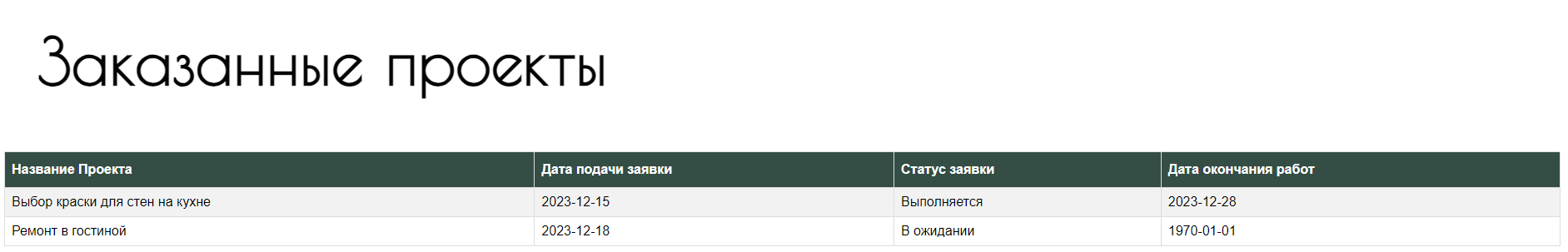


Рисунок 4 — Таблица заказанных проектов пользователем

Реализовать со стороны Python очень просто, буквально в одну строку.

res\_projects = run\_script\_get(f"SELECT \* FROM projects WHERE user\_id='{user\_id}'")

Основная часть реализации находится в прототипе странице, которая создаёт таблицу из двумерного массива SQL-запроса.

{% if pt\_len == 0 %}

<**h2**>У вас ещё нет заявок на проекты </**a**></**h2**>

{% else %}

<**h2**>Заказанные проекты </**a**></**h2**>

<**table** id="customers">

<**tr**>

<**th**>Название Проекта</**th**>

<**th**>Дата подачи заявки</**th**>

<**th**>Статус заявки</**th**>

<**th**>Дата окончания работ</**th**>

</**tr**>

{%for i in range(0, pt\_len)%}

<**tr**>

<**td**>{{proj\_table[i][2]}}</**td**>

<**td**>{{proj\_table[i][3]}}</**td**>

<**td**>{{proj\_table[i][4]}}</**td**>

<**td**>{{proj\_table[i][5]}}</**td**>

</**tr**>

{%endfor%}

</**table**>

{% endif %}

## 3.3 Полный доступ к БД для Администраторов

Администратор сайта может в любой момент, находясь на самом сайте совершить SQL-запрос. Для этого нужно лишь заполнить простую форму и получить результата в виде таблицы (если запрос типа «get»)

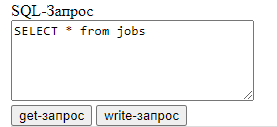


Рисунок 5 — Форма для SQL-запроса на сайте

if (request.form.\_\_contains\_\_('write')):  
 print(request.form['sql'])  
 res = run\_script\_write(request.form['sql'])  
  
if (request.form.\_\_contains\_\_('get')):  
 print(request.form['sql'])  
 res = run\_script\_get(request.form['sql'])  
 return lk('get', res)

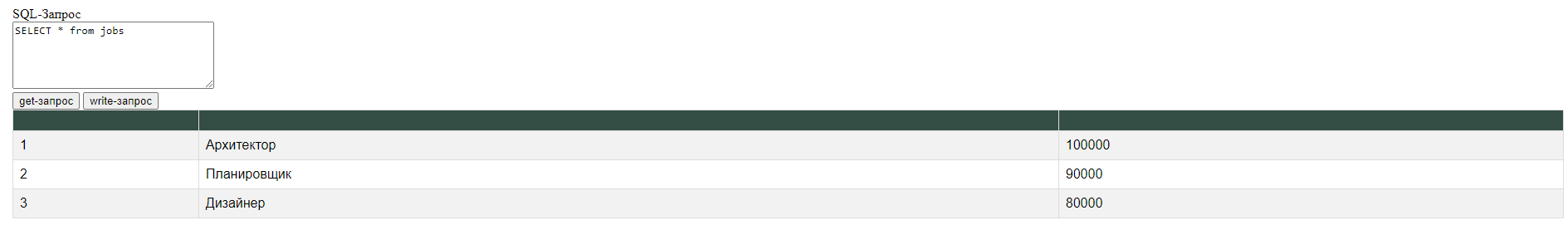


Рисунок 6 — Таблица результата запроса

Заключение

В ходе разработки веб-приложения для архитектурного бюро были предприняты значительные усилия для создания инновационного и удобного пространства виртуального взаимодействия. Одним из ключевых достижений является внедрение базы данных, что позволило эффективно управлять информацией о клиентах, проектах, администрировании.

Создание объектной модели, отражающей базу данных, в системе, стало краеугольным камнем для унификации разработки, обеспечивая более легкую поддержку и расширение функционала приложения. Понимание взаимосвязей между ключевыми объектами, такими как проекты, сотрудники, пользователи и их информация, позволило создать гибкую и масштабируемую архитектуру.

Формулировка запросов, представленных в теоретической части, способствовала обсуждению абстрактных концепций, что оказало положительное влияние на конечный результат проекта.

Освещение спорных идей в теоретической главе дало возможность взглянуть на проект с разных точек зрения, что позволяет принимать обоснованные решения в ходе разработки. Эти идеи стали катализаторами для размышлений о выборе между реляционными и NoSQL базами данных, стратегиями кэширования и вопросами безопасности данных.

В итоге, разработанное веб-приложение для архитектурного бюро представляет собой гармоничное сочетание теоретических концепций и практических решений. Этот проект не только соответствует заявленным целям, но и оставляет пространство для будущего роста и развития, что делает его примером современной индустрии веб-разработки.

Список использованных источников

1. SQLite: Documentation: SQLite 3.44.2 Documentation // SQLite URL: https://www.sqlite.org/docs.html (дата обращения: 06.12.2023).
2. Overview of Entity Framework Core - EF Core | Microsoft Learn // Microsoft Learn URL: https://learn.microsoft.com/en-us/ef/core/ (дата обращения: 06.12.2023).
3. Юсков И. О. Конспекты лекций по базам данных. - Москва: МТУСИ, 2023