

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования

«Иркутский национальный исследовательский технический
университет»

Институт информационных технологий и анализа данных

О Т Ч Ё Т

о прохождении

Производственная

(вид практики: учебная/производственная)

преддипломная практика

(тип практики: технологическая/научно-исследовательская работа/преддипломная и др.)

практики

ИРНИТУ

в

(наименование профильной организации)

Обучающегося Прохорова Е. В. ЭВМ6 20-1

(ФИО, группа, подпись)

Руководитель практики от института ИТиАД доцент Дорофеев А. С.

(ФИО, должность, подпись)

Оценка по практике

Оценено
Дорофеев А. С. 23.05.2024 г.

(ФИО, подпись, дата)

Содержание отчета на 29 стр.

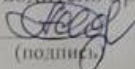
Приложение к отчету на — стр.

Иркутск, 2024 г.

Индивидуальное задание на прохождение

	преддипломной	практики
	(вид (тип) практики)	
для	Прохоров Евгений Викторович	
	(ФИО обучающегося полностью)	
обучающегося	4	курса группы ЭВМб-20-1
по направлению подготовки / специальности Вычислительные машины, комплексы, системы и сети		
Место прохождения практики: ФГБОУ ВО "Ирниту"		
Сроки прохождения практики с «22» апрель 2024 г. по «18» мая 2024 г.		
Цели и задачи прохождения практики: Задачи:		
1. Разработка программного средства для проектирования модели данных		
Содержание практики, вопросы, подлежащие изучению: Анализ языка SQL, Выбор стека технологий, Проектирование интерфейса, Проектирование базы данных, Создание программного средства, Тестирование		
Планируемые результаты практики: работоспособное приложение для проектирования модели данных		

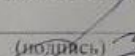
Руководитель практики от института ИТиАД



/ А.С. Дорофеев /
(ФИО)

Согласовано:

Руководитель ООП



/ А.Ф. Аношко /
(ФИО)

« 02 » апреля 2024 г.

С настоящим индивидуальным заданием и с программой практики ознакомлен (а), задание принято к исполнению



«22» апреля 2024 г.

Содержание

Введение.....	4
1 Выбор стека технологий	5
1.1 Backend	5
1.2 Frontend.....	7
1.3 База данных	8
Безопасность базы данных	10
1.4 Инструменты разработки.....	11
Visual Studio Code.....	12
Git	13
pgAdmin	13
2 Проектирование базы данных	15
3 Проектирования интерфейса	20
3.1 Исследование и анализ.....	20
3.2 Прототипирование.....	22
Заключение.....	28
Список использованных источников	29

Введение

Преддипломная практика — это одно из основных этапов подготовки студентов к защите дипломной работы. Ее целью является закрепление и расширение знаний, полученных во время обучения, а также повышение практических навыков в выбранной сфере деятельности.

Актуальность практики обусловлена тем, что в современном мире разработка приложений для работы с базами данных играет ключевую роль в обеспечении эффективного управления информацией. SQL (Structured Query Language) является одним из основных инструментов для работы с реляционными базами данных, позволяя разработчикам создавать, изменять и управлять данными. Однако, разработка SQL скриптов может быть сложной и трудоемкой задачей, особенно при работе с большими и сложными схемами баз данных.

Цель практики состоит в том, чтобы создать инструмент, который позволит разработчикам легко проектировать базы данных, сокращая время и усилия, затрачиваемые на разработку SQL скриптов. Мы сосредотачиваемся на создании интуитивного пользовательского интерфейса, который позволит пользователям визуально создавать и модифицировать структуру баз данных, автоматически генерируя соответствующий SQL код.

Задачи, которые позволяют достичь данной цели:

1. Анализ языка SQL
2. Выбор стека технологий
3. Проектирование интерфейса
4. Проектирование базы данных
5. Создание программного средства
6. Сравнение с аналогами
7. Тестирование

1 Выбор стека технологий

Наш проект представляет собой веб-приложение, созданное с целью облегчить процесс создания SQL скриптов. Предполагается, что данное приложение будет использоваться как для разработчиков, так и для аналитиков данных, упрощая им работу с базами данных. При разработке данного приложения мы учитываем ряд ключевых требований, которые должны быть удовлетворены для обеспечения эффективной работы и удовлетворения потребностей пользователей:

- **Производительность:** Приложение должно обеспечивать высокую производительность и отзывчивость интерфейса даже при работе с большими объемами данных.
- **Масштабируемость:** Необходимо, чтобы приложение могло масштабироваться в случае увеличения нагрузки или расширения функционала.
- **Безопасность:** Важно обеспечить защиту данных пользователей и приложения от угроз безопасности.
- **Интуитивный интерфейс:** Приложение должно иметь понятный и удобный интерфейс для пользователей всех уровней навыков.
- **Интеграция:** Возможность интеграции с другими инструментами разработки, такими как системы управления версиями и проектными досками.

1.1 Backend

Backend, или серверная часть, представляет собой основную часть веб-приложения, ответственную за обработку запросов от клиентской части (frontend) и взаимодействие с базами данных, внешними сервисами и другими компонентами системы. Он обеспечивает логику приложения, обработку данных, аутентификацию и авторизацию пользователей, а также другие бизнес-логику, необходимую для функционирования приложения.

Выбор технологий для backend-разработки играет ключевую роль в определении производительности, масштабируемости, безопасности и функциональности веб-приложения. При рассмотрении различных вариантов для backend, важно учитывать требования проекта и потребности пользователей, чтобы выбрать наиболее подходящий технологический стек.

Выбор Node.js в качестве бэкэнд-технологии для нашего проекта можно обосновать следующим образом:

1. **Высокая производительность:** Node.js построен на основе событийно-ориентированной архитектуры и асинхронного ввода-вывода, что делает его очень эффективным в обработке большого количества одновременных запросов. Это особенно полезно в приложениях с высокой нагрузкой.
2. **Единый язык программирования:** Использование JavaScript как языка программирования как на клиентской, так и на серверной стороне

позволяет уменьшить затраты на обучение и разработку, а также обеспечивает единый стиль кода и переиспользование некоторых компонентов.

3. Большое сообщество и экосистема: Node.js имеет огромное сообщество разработчиков и обширную экосистему библиотек и фреймворков, которые упрощают разработку и расширение функциональности ваших приложений.
4. Модульность и гибкость: Node.js поощряет модульную архитектуру приложений, что позволяет разрабатывать приложения из множества маленьких и переиспользуемых компонентов. Это делает код более чистым, поддерживаемым и масштабируемым.
5. Быстрый старт: Node.js предлагает легкий и быстрый способ создания прототипов приложений благодаря своей простоте и минималистичности.

Сравнение с аналогами:

1. Java (Spring Boot):
 - Java предлагает высокую производительность и надежность, особенно для крупных корпоративных приложений.
 - Spring Boot, в частности, предоставляет множество инструментов и функций для быстрого создания и развертывания приложений.
 - Однако Java имеет более высокий порог входа из-за необходимости в компиляции и более объемного кода.
2. Python (Django или Flask):
 - Python также является популярным выбором для бэкэнд-разработки благодаря своей простоте и выразительности.
 - Django и Flask предоставляют мощные инструменты и фреймворки для создания веб-приложений на Python.
 - Однако Python может быть менее эффективным в обработке большого количества одновременных запросов из-за своего многопоточного подхода.
3. Ruby (Ruby on Rails):
 - Ruby on Rails предоставляет быстрый способ создания веб-приложений с помощью принципа "соглашение больше, чем конфигурация" и обширной библиотеки готовых решений.
 - Ruby может быть привлекательным выбором для команд, предпочитающих элегантный и выразительный код.
 - Однако Ruby может быть менее эффективным в обработке большого количества одновременных запросов из-за своей медленной скорости выполнения.

В целом, выбор между Node.js и его аналогами зависит от конкретных требований вашего проекта, предпочтений команды разработчиков и контекста приложения. Node.js отлично подходит для создания быстрых и масштабируемых веб-приложений с высокой производительностью и удобством разработки.

1.2 Frontend

Frontend, или клиентская часть, представляет собой интерфейс веб-приложения, с которым взаимодействуют пользователи. Он отвечает за отображение данных, интерактивность и визуальное взаимодействие пользователя с приложением.

Выбор технологий для frontend-разработки имеет решающее значение для создания удобного, функционального и привлекательного пользовательского интерфейса. При выборе технологического стека для frontend необходимо учитывать требования проекта к дизайну, производительности, поддержке различных устройств и браузеров, а также опыт пользователя.

Ключевые факторы, которые следует учитывать при выборе технологий для frontend-разработки, включают в себя производительность, масштабируемость, поддержку современных стандартов веб-разработки, а также удобство использования и обучения для разработчиков.

Выбор React.js в качестве фронтенд-технологии для нашего проекта также может быть обоснован несколькими факторами:

1. Производительность и эффективность: React.js использует виртуальный DOM и механизм перерисовки только измененных компонентов, что обеспечивает высокую производительность и эффективное использование ресурсов браузера.
2. Компонентный подход: React.js основан на компонентах, что позволяет разрабатывать приложения из небольших и переиспользуемых элементов. Это делает код более организованным, легко поддерживаемым и масштабируемым.
3. Односторонний поток данных: React.js пропагандирует однонаправленный поток данных (от родительских компонентов к дочерним), что облегчает понимание и отслеживание данных в приложении и упрощает управление состоянием.
4. Широкая экосистема: React.js имеет обширную экосистему инструментов, библиотек и фреймворков, таких как Redux, React Router, Material-UI и многие другие, которые упрощают разработку и расширение функциональности приложений.
5. JSX синтаксис: React.js использует JSX - расширение JavaScript, позволяющее писать HTML-подобный код внутри JavaScript. Это делает код более декларативным, понятным и удобным для работы.
6. Виртуализация на стороне клиента: React.js позволяет создавать мощные интерактивные интерфейсы, включая сложные веб-приложения с асинхронной загрузкой данных и динамическим обновлением пользовательского интерфейса.

Сравнение с аналогами:

1. Angular:

- Angular также является популярным фронтенд-фреймворком, предоставляющим множество инструментов и функций для создания веб-приложений.
- Однако Angular имеет более высокий порог входа из-за своей сложной архитектуры и использования TypeScript.

2. Vue.js:

- Vue.js - это еще один современный фронтенд-фреймворк, который обеспечивает легкую изучаемость и простоту в использовании, а также поддержку компонентного подхода.
- Однако React.js часто предпочтительнее для крупных и сложных проектов благодаря своей более широкой экосистеме и поддержке со стороны крупных компаний.

3. Svelte:

- Svelte предлагает новый подход к созданию веб-приложений, основанный на компиляции компонентов в чистый JavaScript во время сборки.
- Хотя Svelte обещает лучшую производительность и меньший объем кода, React.js все еще остается более распространенным и широко используемым фреймворком.

В целом, React.js представляет собой мощный инструмент для создания современных веб-приложений, обладающий высокой производительностью, гибкостью и широкой поддержкой сообщества.

1.3 База данных

База данных представляет собой структурированное хранилище данных, которое используется для хранения, управления и организации информации, необходимой для функционирования приложений и систем. Выбор технологий для баз данных играет важную роль в обеспечении эффективного хранения, доступа и обработки данных.

При выборе базы данных необходимо учитывать требования проекта к масштабируемости, производительности, надежности, безопасности и поддержке специфических типов данных. Важно также оценить потенциальные интеграционные возможности с другими компонентами системы и удобство использования для разработчиков.

Ключевые факторы, которые следует учитывать при выборе технологии базы данных, включают в себя тип данных, модель хранения, поддержку транзакций, масштабируемость, производительность, а также возможности резервного копирования и восстановления данных.

Выбор PostgreSQL в качестве основной базы данных для нашего проекта можно обосновать по ряду причин:

1. Многоплатформенность: VS Code поддерживает операционные системы Windows, macOS и Linux, что делает его универсальным инструментом для разработчиков, работающих на различных платформах.

2. **Расширяемость:** VS Code предлагает обширный репозиторий расширений, который позволяет разработчикам настраивать среду разработки под свои потребности. От поддержки различных языков программирования до инструментов управления версиями и отладки, расширения обеспечивают широкий спектр функциональности.
3. **Интеграция с Git:** VS Code имеет встроенную поддержку Git, что делает управление версиями и совместную работу в проектах легкой и интуитивно понятной.
4. **Мощный редактор кода:** Редактор кода в VS Code обладает множеством возможностей, включая подсветку синтаксиса, автоматическое завершение кода, быструю навигацию, интегрированный поиск и замену текста, а также поддержку различных видов файлов.
5. **Отладка:** Интегрированная система отладки в VS Code облегчает процесс обнаружения и исправления ошибок в коде.
6. **Интеграция с различными фреймворками и средами разработки:** VS Code поддерживает множество популярных языков программирования, фреймворков и инструментов разработки, таких как JavaScript, Python, Node.js, .NET и многие другие.
7. **Автоматические обновления:** VS Code регулярно обновляется, предоставляя пользователям новые функции и улучшения без необходимости установки новой версии.

Учитывая эти факторы, PostgreSQL представляет собой привлекательное решение для многих проектов, особенно тех, где требуется надежная, масштабируемая и гибкая база данных.

Давайте сравним PostgreSQL с двумя из его основных аналогов - MySQL и SQLite - по нескольким ключевым аспектам:

1. **Функциональные возможности:**

- **PostgreSQL:** Обладает богатым набором функциональных возможностей, включая поддержку геоданных, полнотекстовый поиск, триггеры, процедуры, расширяемые типы данных и многое другое.
- **MySQL:** Предлагает широкий набор стандартных функций и возможностей, но несколько более ограничен в расширяемости и функциональности по сравнению с PostgreSQL.
- **SQLite:** Легкая встраиваемая база данных, обычно используется для простых приложений или встраивается в мобильные приложения. Несмотря на это, она поддерживает большинство стандартных SQL-возможностей.

2. **Производительность и масштабируемость:**

- **PostgreSQL:** Обеспечивает хорошую производительность и масштабируемость, особенно при правильной настройке индексов и оптимизации запросов. Может быть использован для крупных и сложных проектов.

- MySQL: Имеет хорошую производительность и масштабируемость, особенно на низкой и средней нагрузке. Однако при очень высоких нагрузках может потребоваться более тщательная настройка.
- SQLite: Часто используется для небольших приложений или для прототипирования из-за своей простоты и легковесности. Не подходит для высоконагруженных приложений или крупных баз данных.

3. Поддержка стандартов и соответствие SQL:

- PostgreSQL: Стремится к полному соответствию стандартам SQL и предоставляет обширные возможности для разработчиков.
- MySQL: Хорошо соответствует основным стандартам SQL, но может отличаться в некоторых аспектах от PostgreSQL.
- SQLite: Также соответствует основным стандартам SQL, но, как и в случае с MySQL, есть некоторые различия в функциональности и возможностях.

4. Распространенность и экосистема:

- PostgreSQL: Стремится к полному соответствию стандартам SQL и предоставляет обширные возможности для разработчиков.
- MySQL: Хорошо соответствует основным стандартам SQL, но может отличаться в некоторых аспектах от PostgreSQL.
- SQLite: Также соответствует основным стандартам SQL, но, как и в случае с MySQL, есть некоторые различия в функциональности и возможностях.

В целом, PostgreSQL представляет собой мощную и гибкую базу данных, особенно подходящую для крупных и сложных проектов, требующих богатый набор функциональных возможностей и высокую надежность. MySQL и SQLite также имеют свои преимущества и подходят для различных типов приложений и сценариев использования.

Безопасность базы данных

Безопасность баз данных — это критически важный аспект в области информационной технологии. Она охватывает различные аспекты, включая аутентификацию, авторизацию, шифрование, аудит и другие меры для защиты данных от несанкционированного доступа, изменений и утечек.

1. Аутентификация

Аутентификация — это процесс проверки подлинности пользователей и устройств перед предоставлением доступа к базе данных. Включает в себя методы, такие как:

- Имя пользователя и пароль: Самый распространенный метод аутентификации, который требует от пользователей предоставить учетные данные для доступа к базе данных.

- Многофакторная аутентификация (MFA): Дополнительный слой защиты, который требует от пользователя предоставить не только пароль, но и другой аутентификационный фактор, такой как одноразовый код, биометрические данные или аппаратный ключ.
- Интеграция с внешними системами аутентификации: Возможность использовать сторонние системы аутентификации, такие как LDAP, OAuth или SAML, для централизованного управления доступом.

2. Авторизация

Авторизация определяет права доступа пользователей к данным и операциям в базе данных. Ключевые концепции включают:

- Ролевая модель доступа: Определение ролей и привилегий для пользователей и групп пользователей. Назначение ролей облегчает управление доступом и обеспечивает принцип наименьших привилегий (Principle of Least Privilege).
- Гибкие политики доступа: Возможность определения детализированных прав доступа на уровне таблиц, столбцов, процедур и других объектов базы данных.

3. Шифрование данных

Шифрование данных — это процесс преобразования данных в нечитаемый формат с использованием криптографических алгоритмов.

Основные виды шифрования включают:

- Шифрование в покое: Защита данных в хранилище базы данных путем шифрования файлов данных и журналов транзакций.
- Шифрование в движении: Обеспечение безопасной передачи данных между клиентами и серверами с использованием протоколов шифрования, таких как SSL / TLS.

4. Аудит и мониторинг

Аудит и мониторинг — это процесс записи и анализа действий пользователей и изменений данных в базе данных. Это позволяет выявлять несанкционированные действия и угрозы безопасности. Основные аспекты включают:

- Журналирование событий: Запись всех действий пользователей, выполненных запросов и изменений данных для последующего анализа и отслеживания.
- Анализ журналов: Использование специализированных инструментов для обнаружения аномалий, внутренних угроз и других подозрительных действий.

1.4 Инструменты разработки

Выбор правильных инструментов разработки играет ключевую роль в эффективной разработке программного обеспечения, обеспечивая

комфортную рабочую среду для разработчиков и оптимизируя процессы создания, тестирования и развертывания приложений.

При рассмотрении инструментов разработки необходимо учитывать требования проекта, особенности команды разработки и конкретные потребности разработчиков. Это включает в себя выбор интегрированных сред разработки (IDE), систем управления версиями кода, инструментов для непрерывной интеграции и развертывания (CI/CD), а также других инструментов для управления проектом и коммуникации в команде.

Целью данного раздела является обсуждение ключевых инструментов разработки, которые могут быть использованы для создания нашего проекта, а также их преимуществ и сферы применения.

Visual Studio Code

Visual Studio Code (VS Code) — это мощное, легкое и гибкое интегрированное средство разработки (IDE), созданное Microsoft. Вот несколько ключевых особенностей:

1. Многоплатформенность: VS Code поддерживает операционные системы Windows, macOS и Linux, что делает его универсальным инструментом для разработчиков, работающих на различных платформах.
2. Расширяемость: VS Code предлагает обширный репозиторий расширений, который позволяет разработчикам настраивать среду разработки под свои потребности. От поддержки различных языков программирования до инструментов управления версиями и отладки, расширения обеспечивают широкий спектр функциональности.
3. Интеграция с Git: VS Code имеет встроенную поддержку Git, что делает управление версиями и совместную работу в проектах легкой и интуитивно понятной.
4. Мощный редактор кода: Редактор кода в VS Code обладает множеством возможностей, включая подсветку синтаксиса, автоматическое завершение кода, быструю навигацию, интегрированный поиск и замену текста, а также поддержку различных видов файлов.
5. Отладка: Интегрированная система отладки в VS Code облегчает процесс обнаружения и исправления ошибок в коде.
6. Интеграция с различными фреймворками и средами разработки: VS Code поддерживает множество популярных языков программирования, фреймворков и инструментов разработки, таких как JavaScript, Python, Node.js, .NET и многие другие.
7. Автоматические обновления: VS Code регулярно обновляется, предоставляя пользователям новые функции и улучшения без необходимости установки новой версии.

Эти особенности делают Visual Studio Code одним из самых популярных и мощных инструментов разработки для широкого круга разработчиков.

Git

Git — это распределенная система управления версиями, разработанная Линусом Торвальдсом. Вот некоторые ключевые характеристики и преимущества Git:

1. Многоплатформенность: VS Code поддерживает операционные системы Windows, macOS и Linux, что делает его универсальным инструментом для разработчиков, работающих на различных платформах.
2. Расширяемость: VS Code предлагает обширный репозиторий расширений, который позволяет разработчикам настраивать среду разработки под свои потребности. От поддержки различных языков программирования до инструментов управления версиями и отладки, расширения обеспечивают широкий спектр функциональности.
3. Интеграция с Git: VS Code имеет встроенную поддержку Git, что делает управление версиями и совместную работу в проектах легкой и интуитивно понятной.
4. Мощный редактор кода: Редактор кода в VS Code обладает множеством возможностей, включая подсветку синтаксиса, автоматическое завершение кода, быструю навигацию, интегрированный поиск и замену текста, а также поддержку различных видов файлов.
5. Отладка: Интегрированная система отладки в VS Code облегчает процесс обнаружения и исправления ошибок в коде.
6. Интеграция с различными фреймворками и средами разработки: VS Code поддерживает множество популярных языков программирования, фреймворков и инструментов разработки, таких как JavaScript, Python, Node.js, .NET и многие другие.
7. Автоматические обновления: VS Code регулярно обновляется, предоставляя пользователям новые функции и улучшения без необходимости установки новой версии.

Git стал стандартом для управления версиями в различных проектах благодаря своей гибкости, мощным возможностям и широкому распространению.

pgAdmin

pgAdmin - это бесплатное кроссплатформенное программное обеспечение с открытым исходным кодом, предназначенное для администрирования PostgreSQL и связанных баз данных. Оно обеспечивает удобный графический интерфейс для управления базами данных PostgreSQL и выполнения различных административных задач.

Вот некоторые основные черты и возможности pgAdmin:

1. Интерфейс с поддержкой многих окон: pgAdmin предоставляет удобный пользовательский интерфейс, который позволяет администраторам

работать с несколькими окнами и вкладками, что облегчает управление несколькими базами данных одновременно.

2. **Обозреватель объектов:** В pgAdmin есть обозреватель объектов, который позволяет администраторам легко просматривать, создавать, изменять и удалять различные объекты базы данных, такие как таблицы, представления, индексы, функции и т.д.
3. **Редактор SQL-запросов:** pgAdmin включает мощный SQL-редактор с подсветкой синтаксиса, автозавершением и другими полезными функциями, что делает написание и выполнение SQL-запросов более удобным.
4. **Административные инструменты:** pgAdmin предоставляет различные административные инструменты для управления базой данных, такие как мониторинг активности, администрирование безопасности, настройка параметров и т.д.
5. **Интеграция с сервером:** pgAdmin обеспечивает интеграцию с сервером PostgreSQL, что позволяет администраторам легко подключаться к удаленным серверам и управлять ими из графического интерфейса.
6. **Поддержка расширений и плагинов:** pgAdmin расширяем и поддерживает плагины, что позволяет пользователю добавлять новые функции и возможности в программу.

В целом, pgAdmin является мощным и гибким инструментом для администрирования баз данных PostgreSQL, который обеспечивает широкий спектр функций и возможностей для работы с данными и объектами базы данных.

2 Проектирование базы данных

Web-приложению необходима база данных для хранения записей авторизованных пользователей и информации, необходимой для функционирования web-приложения, такой как названия SQL, доступные на нашем ресурсе.

База данных будет играть ключевую роль в обеспечении безопасности и управлении доступом к нашим ресурсам. Она будет содержать информацию о пользователях, их учетных данных, а также о том, какие SQL-запросы доступны на web-приложении.

Эта база данных будет служить основой для аутентификации пользователей, контроля доступа и обеспечения целостности данных на web-приложении. Благодаря ей, мы сможем эффективно управлять информацией, предоставлять пользователям необходимые функции и обеспечивать безопасность данных.

Разработка и поддержка такой базы данных требует внимательного проектирования, чтобы обеспечить оптимальную производительность, безопасность и масштабируемость нашего веб-ресурса.

Определим сущности и атрибуты базы данных.

Сущность представляет собой объект или концепцию в вашей системе, который вы хотите отслеживать и хранить в базе данных. Это может быть что угодно, от конкретного объекта (например, человек, товар, заказ) до абстрактной концепции (например, категория товара, роль пользователя). Сущности обычно представлены в виде таблиц в базе данных.

Атрибуты — это свойства или характеристики сущности, которые определяют её. Например, если рассматривается сущность "пользователь", то её атрибутами могут быть имя, фамилия, адрес электронной почты и т.д. Атрибуты помогают описать каждую сущность более детально и хранятся в виде столбцов в таблицах базы данных их можно увидеть в таблице 1.

Табл.1 – Сущности и атрибуты

Сущность	Атрибуты	
Пользователи	Логин	Пароль
Список SQL	Название	
Таблица	Поле	Название

Далее нормализуем данные.

Нормализация данных — это процесс организации структуры базы данных таким образом, чтобы минимизировать избыточность информации и предотвратить аномалии при вставке, обновлении и удалении данных. Цель нормализации состоит в том, чтобы разделить данные на небольшие логически связанные таблицы, уменьшив повторение информации и обеспечив согласованность данных.

Основные преимущества нормализации данных включают уменьшение избыточности информации, облегчение поддержки и модификации базы данных, а также предотвращение аномалий при манипуляции данными.

Однако следует помнить, что слишком агрессивная нормализация может привести к усложнению запросов и ухудшению производительности, поэтому важно достигать баланса между нормализацией и денормализацией в зависимости от конкретных потребностей приложения.

Чтобы обеспечить нормальное функционирование и редактирование данных из сущности таблица было выделенных целых три таблицы, которые связаны между собой внешними ключами. Смотрите на рисунке 2

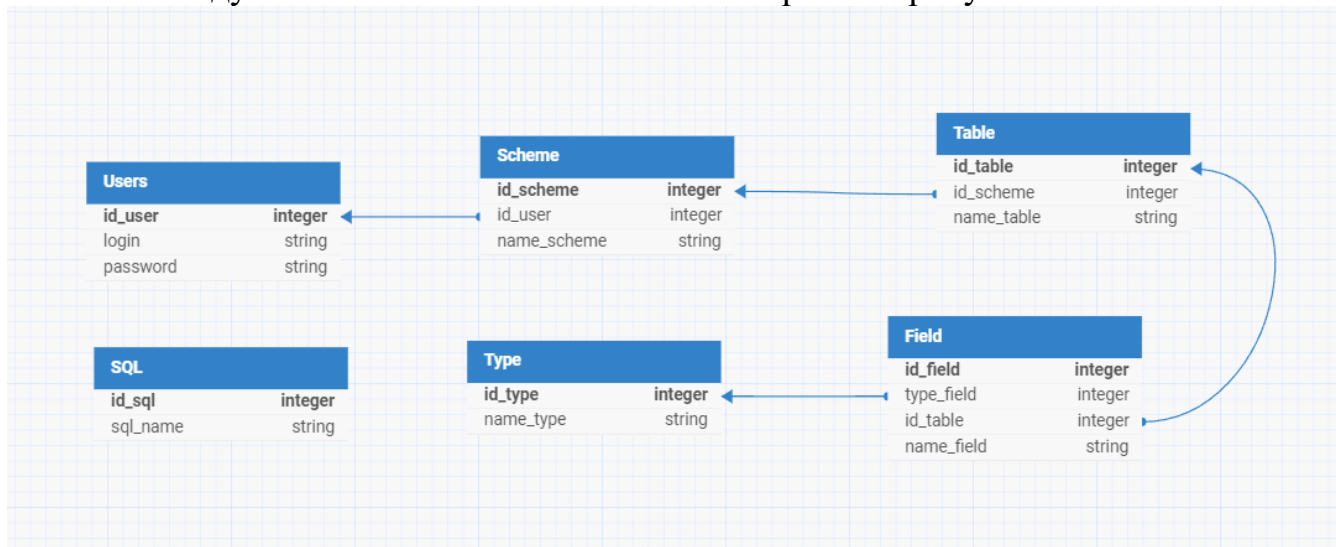


Рисунок 2 – Схема данных

Начнём с разбора первой таблицы Users, которую можно увидеть на рисунке 3, в ней находятся данные, которые понадобятся для авторизации соответственно логин и пароль.

Users	
id_user	integer
login	string
password	string

Рисунок 3 – Таблица Users

В следующей табличке хранится название схемы, и внешний ключ id пользователя, который позволит узнать какой пользователь создал данную схему. Её можно увидеть на рисунке 4.

Scheme	
id_scheme	integer
id_user	integer
name_scheme	string

Рисунок 4 – Таблица Scheme

Далее идёт таблица, которую можно увидеть на рисунке 5, где хранятся данные о таблицах, которые хранятся в этой схеме. Это осуществляется с помощью внешнего ключа id схемы.

Table	
id_table	integer
id_scheme	integer
name_table	string

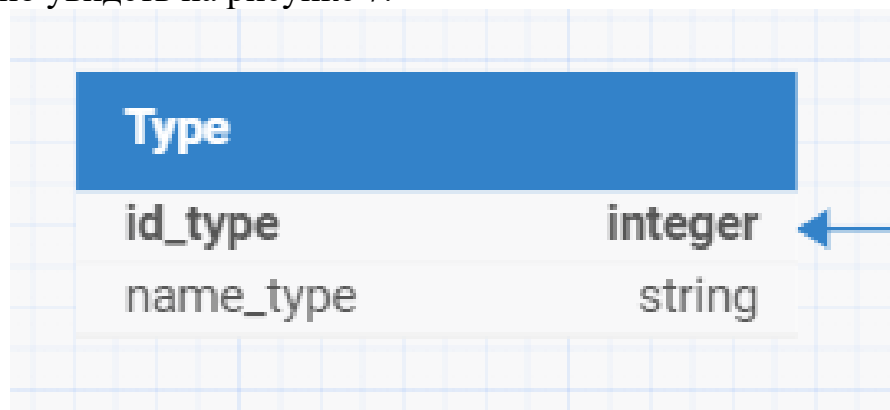
Рисунок 5 – Таблица Table

Ещё правее находится таблица со строками нашей таблицы, там хранится тип строки и её имя, а еще внешний ключ, который связывает её с таблицей в которой находятся эти строки. Её можно увидеть на рисунке 6.

Field	
id_field	integer
type_field	integer
id_table	integer
name_field	string

Рисунок 6 – Таблица Field

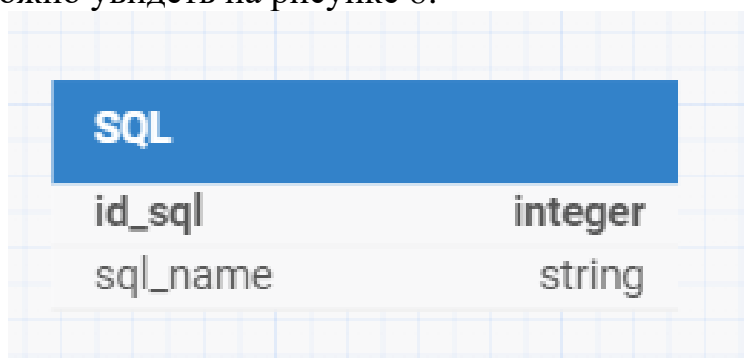
Далее идёт табличка, в которой хранятся типы данных наших строк, которую можно увидеть на рисунке 7.



Type	
id_type	integer
name_type	string

Рисунок 7 – Таблица Type

Отдельной не связанной с другими таблицами внешними ключами является таблица с типами SQL, которые поддерживает наше web-приложение. Её можно увидеть на рисунке 8.



SQL	
id_sql	integer
sql_name	string

Рисунок 8 – Таблица SQL

Такая структура данных позволит эффективно сохранять записи пользователей и хранить их длительный срок.

В таблице 2 представлен перевод и название полей таблиц нашей базы данных.

Табл.2 – Назначение таблиц

Таблица	Поле	Назначение
Users (Пользователи)	id_user (ид пользователя)	Уникальный идентификатор каждого пользователя
	login(логин)	Уникальное имя пользователя
	password(пароль)	Пароль от аккаунта
Scheme (Схема)	id_scheme (ид схемы)	Уникальный идентификатор каждой схемы
	id_user (ид пользователя)	Уникальный идентификатор каждого пользователя
	name_scheme (название схемы)	Название схемы
Table (Таблица)	id_table (ид таблицы)	Уникальный идентификатор каждой таблицы
	id_scheme (ид схемы)	Уникальный идентификатор каждой схемы
	name_table (название таблицы)	Название таблицы
Field (Поле)	id_field	Уникальный идентификатор каждого поля
	type_field (тип поля)	Тип поля
	id_table (ид таблицы)	Уникальный идентификатор каждой таблицы
	name_field (название поля)	Название поля
Type (тип)	id_type (ид типа)	Уникальный идентификатор каждого типа
	name_type (название типа)	Название типа
SQL (эскьюэль)	id_sql (ид sql)	Уникальный идентификатор каждого SQL
	sql_name (название sql)	Название SQL

3 Проектирования интерфейса

Проектирование интерфейса играет фундаментальную роль в разработке веб-приложений, определяя пользовательский опыт и взаимодействие с приложением. Этот раздел посвящен обсуждению процесса проектирования интерфейса, включая исследование и анализ, прототипирование. Рассмотрение данных аспектов поможет создать удобный, интуитивно понятный и привлекательный интерфейс, способствующий успешной реализации задач приложения и удовлетворению потребностей пользователей.

3.1 Исследование и анализ

Идентификация целевой аудитории:

1. Студенты и обучающиеся:
 - Эта категория пользователей включает студентов университетов и колледжей, изучающих базы данных, а также людей, проходящих курсы и обучение по данной тематике.
 - Они ищут инструменты, которые помогут им понять основы проектирования баз данных и практиковаться в создании собственных схем.
2. Разработчики программного обеспечения:
 - Разработчики, создающие приложения и сервисы, которые используют базы данных в качестве хранилища данных.
 - Они нуждаются в инструментах для проектирования и моделирования структуры баз данных перед началом разработки.
3. Баз данных администраторы:
 - Специалисты, отвечающие за управление и поддержку баз данных в предприятии.
 - Им требуются инструменты для анализа существующих баз данных, создания новых схем и оптимизации их производительности.
4. Аналитики данных:
 - Специалисты, занимающиеся анализом данных и созданием отчетов на основе информации из баз данных.
 - Они могут использовать инструменты для создания моделей данных и определения требований к структуре баз данных для оптимального анализа.
5. Начинающие пользователи:
 - Люди, не имеющие опыта в проектировании баз данных, но имеющие потребность в создании простых структур для своих проектов или идей.
 - Им нужен интуитивно понятный и легко осваиваемый инструмент для создания баз данных без необходимости в глубоких знаниях теории баз данных.

Анализ требований для приложения проектирования баз данных:

1. Создание и редактирование таблиц:
 - Пользователи должны иметь возможность создавать новые таблицы для хранения данных.
 - Требуется возможность добавления и удаления столбцов в таблицах, а также изменения их типов данных и других свойств.
2. Определение отношений между таблицами:
 - Приложение должно позволять пользователям определять связи между различными таблицами, такие как один-к-одному, один-ко-многим, многие-ко-многим.
 - Должна быть возможность управления связями, включая их создание, изменение и удаление.
3. Генерация SQL-кода:
 - Пользователи должны иметь возможность генерировать SQL-код для создания базы данных на основе созданных ими таблиц и связей.
 - Код должен быть совместим с распространенными СУБД, такими как MySQL, PostgreSQL, SQLite и другими.
4. Визуализация структуры базы данных:
 - Приложение должно предоставлять графический интерфейс для визуализации структуры базы данных, включая таблицы и связи между ними.
 - Требуется возможность масштабирования и перемещения элементов для удобного просмотра.
5. Поддержка различных типов данных:
 - Приложение должно поддерживать широкий спектр типов данных, используемых в различных СУБД, включая числовые, текстовые, даты, времена и другие.
 - Должна быть возможность настройки дополнительных параметров для каждого типа данных, таких как размер поля, ограничения на значения и т. д.
6. Импорт и экспорт данных:
 - Пользователи должны иметь возможность импортировать данные из внешних источников, таких как CSV-файлы или другие базы данных.
 - Требуется возможность экспорта структуры базы данных и ее содержимого для обмена данными с другими приложениями или для создания резервных копий.
7. Удобный интерфейс пользователя:
 - Интерфейс приложения должен быть интуитивно понятным и легко освоенным даже для пользователей без опыта в проектировании баз данных.
 - Требуется обеспечить удобство взаимодействия с элементами интерфейса, такими как контекстные меню, кнопки быстрого доступа и т. д.

8. Поддержка коллаборации:

- Приложение должно предоставлять возможность совместной работы нескольких пользователей над одной базой данных, в том числе с возможностью совместного редактирования и комментирования структуры.
- Требуется механизм управления доступом и версионирования изменений.

3.2 Прототипирование

Начнём с базового прототипа нашего интерфейса. Прототип интерфейса можно увидеть на рисунке 9.



Рисунок 9 – Прототип интерфейса

Далее разберем его отдельные элементы.

В низу находится Footer с кнопками изменения масштаба, который можно увидеть на рисунке 10.

Рисунок 10 – Прототип Footer



Рисунок 11 – Прототип кнопок масштабирования

Кнопки масштабирования нужны, чтобы отдалять или приближать рабочее пространство и не взирая на количество таблиц комфортно работать с ними. Их можно увидеть на рисунке 11.

В центре располагается рабочее пространство, которое можно увидеть на рисунке 12.

Рисунок 12 – Прототип рабочего пространства

Наверху располагается Header на котором находятся иконки выполняющие различные функции. Его можно увидеть на рисунке 13.

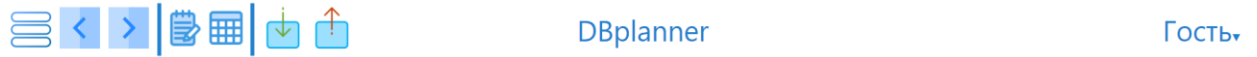


Рисунок 13 – Прототип header

Слева располагается иконка открывающегося меню. Смотрите рисунок 14.



Рисунок 14 – Прототип иконки открывающегося меню

В открывающемся меню, которое вы можете увидеть на рисунке 15, находятся три пункта создать, загрузить и сохранить.

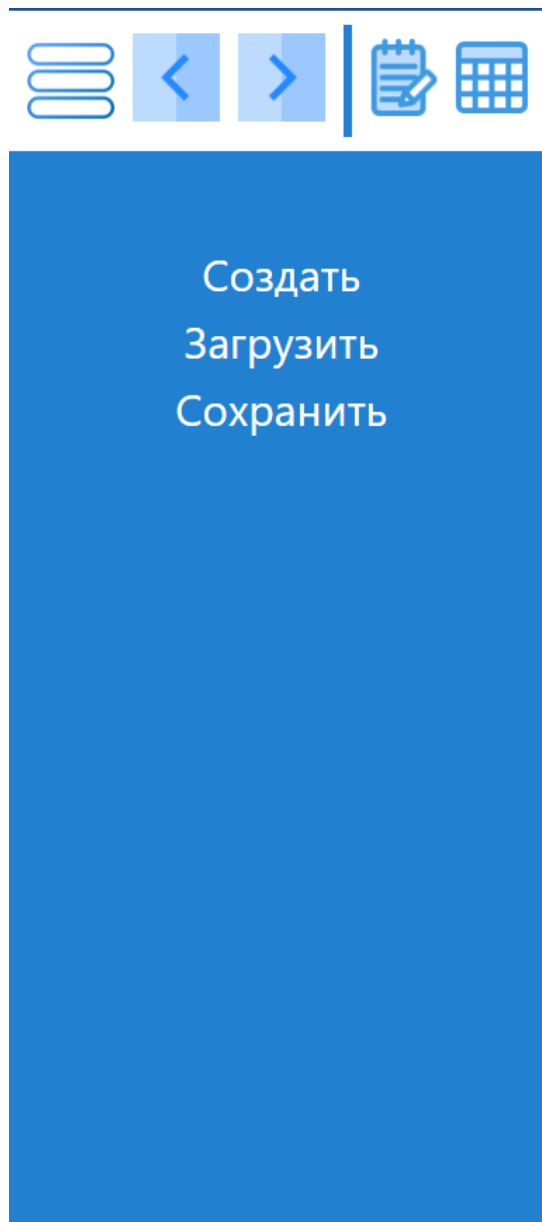


Рисунок 15 – Прототип открывающегося меню

Правее находятся иконки для отмены действий и перемещения между ними. Их вы можете увидеть на рисунке 16.



Рисунок 16 – Прототип иконок действий

Правее находятся иконки для вызова стикера для заметок и вызов окна создания таблицы. Они находятся на рисунке 17.



Рисунок 17 – Прототип иконок заметок и таблицы

На стикере мы можем написать какую-то информацию необходимую для нас. Это можно увидеть на рисунке 18.

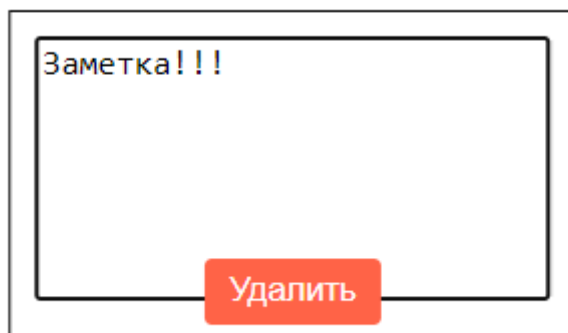


Рисунок 18 – Прототип стикера с заметками

На прототипе окна для создания таблиц, который можно посмотреть на рисунке 19, мы можем ввести название и выбрать тип данных, основной ключ, уникальное поле, автоинкремент и выбрать внешний ключ.

Название таблицы							
Структура таблицы							
Имя	Тип	Основной ключ	Уникальное поле	Автоинкремент	Внешний ключ	Внешняя таблица	Внешнее поле
<input type="text"/>	<input type="text" value="Select"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		<input style="color: blue;" type="button" value="+"/>
<input type="button" value="Сохранить"/>							
<input type="button" value="Заккрыть"/>							

Рисунок 19 – Прототип окна создания таблицы

Далее идут иконки импорта и экспорта. Вы можете увидеть их на рисунке 20.



Рисунок 20 – Иконки импорта и экспорта.

В окне импорта, которое можно посмотреть на рисунке 21, мы можем выбрать тип SQL и сгенерировать скрипт.

Выберите тип SQL

Select... ▼

Сгенерировать SQL

Заккрыть

Рисунок 21 – Окно экспорта.
В центре находится название нашего приложения.

DBplanner

Рисунок 22 – Название приложения
В крайнем правом углу находится система регистрации

Гость▼

Рисунок 23 – Система регистрации до входа
После наведения на надпись гость нам откроется два варианта входа и регистрации

Войти

Зарегистрироваться

Рисунок 24 – Система авторизации
После выбора одного из двух вариантов нам откроется окно входа и регистрации соответственно. Смотрите на рисунках 25 и 26 соответственно.

Email

Пароль

Зарегистрироваться

Закреть

Есть аккаунт? [Войдите](#)

Рисунок 25 – Окно регистрации

Email

Пароль

Войти

Закреть

Нет аккаунта? [Зарегистрируйтесь](#)

Рисунок 26 – Окно входа

Заключение

В ходе данной работы был представлен проект, направленный на разработку приложения с фокусом на упрощении процесса создания SQL скриптов для работы с базами данных. Основной целью проекта было создание интуитивно понятного пользовательского интерфейса, который позволил бы разработчикам визуально создавать, модифицировать и управлять структурой баз данных, минуя необходимость ручного написания SQL кода.

Процесс разработки охватывал несколько ключевых этапов, начиная с анализа требований и заканчивая оценкой качества. В рамках анализа требований детально изучили потребности пользователей и основные задачи, которые приложение должно было решать. Это позволило определить основные функциональные возможности и параметры конфигурации, которые должны были быть включены в приложение.

Проектирование архитектуры играло ключевую роль в создании приложения. Разработали общую архитектуру, учитывая особенности работы с базами данных и требования к пользовательскому интерфейсу. Важным аспектом проектирования была гибкость и расширяемость приложения, чтобы обеспечить возможность дальнейшего развития и добавления новых функциональных возможностей.

В ходе реализации функциональности уделяли особое внимание разработке удобного и интуитивно понятного пользовательского интерфейса. Также создали мощный движок для работы с базами данных, который позволял выполнять различные операции без необходимости в написании SQL кода вручную.

Тестирование и оценка качества играли важную роль в обеспечении надежности и производительности приложения. Провели обширное тестирование для обнаружения и исправления возможных ошибок, а также оценили общее качество приложения с учетом его производительности и удобства использования.

Проект направлен на предоставление разработчикам мощного инструмента для работы с базами данных, способствующего повышению их производительности и качества работы. Стремались создать приложение, которое было бы не только эффективным инструментом для создания и управления базами данных, но и интуитивно понятным и легким в использовании.

Список использованных источников

1. Коннолли Т., Бегг К. Базы данных: проектирование, реализация и сопровождение. 2009.
2. Дорофеев А.С. Базы данных: учебное пособие. ИрГТУ, 2008. 80 с.
3. Лесников М. React.js: Приложения для интернета вещей.
4. Дюккер Э. Node.js для профессионалов.
5. Фармаковский Д. PostgreSQL. Эффективное использование.
6. Макбрайд С.Б., Шевченко А.М. SQL. Полное руководство.
7. Степин А. Основы SQL. Практическое руководство по манипулированию данными. 2019.
8. Фейерштейн С. Oracle PL/SQL. Справочник разработчика. 2014.
9. Дольников Д. PostgreSQL. Администрирование и разработка. 2019.
10. Бек А. SQL для чайников. Полное руководство. 2016.
11. Маллен К.С. Базы данных: от реляционных к NoSQL. Введение в базы данных. 2013.
12. Бобруйко В. Учебник по Microsoft SQL Server 2016. 2017.
13. PostgreSQL Documentation. [Электронный ресурс]. URL: <https://www.postgresql.org/docs/> (дата обращения 17.05.2024).
14. React Documentation. [Электронный ресурс]. URL: <https://reactjs.org/docs/getting-started.html> (дата обращения 17.05.2024).
15. Node.js Documentation. [Электронный ресурс]. URL: <https://nodejs.org/en/docs/> (дата обращения 17.05.2024).