# **Введение**

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.

В данном выпускном квалификационном проекте (ВКР) мы обращаем внимание на разработку приложения, аналогичного DB Designer, который предоставляет интуитивно понятный и удобный интерфейс для создания SQL скриптов. Наше приложение будет стремиться облегчить процесс проектирования баз данных, предоставляя разработчикам инструменты для создания, визуализации и модификации структуры баз данных без необходимости написания SQL кода вручную.

Цель нашего проекта состоит в том, чтобы создать инструмент, который позволит разработчикам легко проектировать и управлять базами 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.

В данной работе мы рассмотрим процесс разработки такого приложения, включая анализ требований, проектирование архитектуры, реализацию функциональности, тестирование и оценку качества. Мы также рассмотрим существующие инструменты и подходы к разработке приложений для работы с базами данных, чтобы выявить лучшие практики и улучшить наш продукт.

Ключевыми аспектами нашего проекта являются удобство использования, гибкость конфигурации и возможность расширения функциональности в соответствии с потребностями пользователей. Наша цель - предоставить разработчикам мощный инструмент для работы с базами данных, который поможет им повысить производительность и качество своей работы

1. **Основная часть**
   1. **История языка SQL**

История языка SQL (Structured Query Language) берет свое начало в 1970-х годах, когда IBM разрабатывал систему управления базами данных (СУБД) под названием System R. В это время проектировался язык запросов, который мог бы облегчить взаимодействие с данными в базе. Это привело к созданию языка SEQUEL (Structured English QUEry Language), который позднее был переименован в SQL из-за проблем с торговыми марками.

SQL был официально стандартизирован в 1986 году Американским Национальным Институтом Стандартов (ANSI) и в 1987 году Международной Организацией по Стандартизации (ISO). Эти стандарты были затем доработаны и дополнены в последующие годы.

С течением времени SQL стал широко распространенным языком запросов для реляционных баз данных. Различные вендоры СУБД, такие как Oracle, Microsoft, IBM, PostgreSQL и другие, развивали свои собственные реализации SQL, иногда с небольшими вариациями стандарта.

Важным моментом в истории SQL стало появление стандарта SQL-92, который включал множество новых функций и расширений языка. Это включало в себя поддержку хранимых процедур, транзакций, агрегатных функций и многих других возможностей.

В последующие годы SQL продолжал развиваться, появлялись новые стандарты, такие как SQL:1999, SQL:2003, SQL:2008 и т. д., каждый из которых добавлял новые функции и улучшения. Одним из важных направлений развития стало расширение возможностей SQL для работы с различными типами данных, включая XML, JSON и географические данные.

Сегодня SQL является одним из самых популярных и широко используемых языков запросов в мире. Он применяется в различных областях, включая веб-разработку, аналитику данных, бизнес-анализ, финансы, медицину и многое другое. SQL остается основным инструментом для работы с данными в реляционных базах данных и продолжает развиваться, чтобы удовлетворять растущие потребности современных информационных технологий.

* + 1. **Oracle**

История языка SQL в контексте Oracle включает в себя ряд ключевых моментов, начиная с момента создания самой компании.

Oracle Corporation была основана в 1977 году Ларри Эллисоном, Бобом Майнером и Эдом Остеном. Изначально компания занималась разработкой системы управления базами данных, которая впоследствии стала известной как Oracle Database.

Oracle была одним из первых вендоров, которые предложили поддержку языка SQL в своей СУБД. В начале 1980-х годов Oracle выпустила свою первую версию базы данных, которая включала поддержку SQL. Этот язык запросов стал ключевым элементом взаимодействия с данными в Oracle Database.

С развитием компании Oracle и ее продуктов SQL продолжал развиваться и совершенствоваться. Oracle активно участвовала в процессе стандартизации SQL, и многие возможности, которые стали стандартом в SQL, были первоначально внедрены именно в продуктах Oracle.

Одним из важных моментов в истории SQL Oracle было выпуск нескольких значительных версий своей базы данных, каждая из которых вносила новые функции и улучшения в язык SQL и его возможности. Например, Oracle Database 8i в конце 1990-х годов представила поддержку объектно-ориентированных возможностей в SQL, а Oracle Database 10g в начале 2000-х годов внедрила различные улучшения производительности и администрирования, включая автоматическое управление ресурсами и диагностику.

Следует отметить, что Oracle также разработала свои собственные расширения SQL, которые расширяют стандартные возможности языка для удовлетворения специфических потребностей пользователей и приложений.

Сегодня Oracle Database остается одной из самых популярных реляционных баз данных в мире, и SQL продолжает быть основным языком запросов для работы с данными в этой системе. Стандарт SQL, развиваемый международными организациями по стандартизации, и дальнейшие инновации Oracle в области SQL, продолжают формировать эволюцию языка запросов и его применение в современном мире информационных технологий.

* + 1. **Microsoft SQL Server (MS SQL)**

История Microsoft SQL Server (MS SQL) связана с развитием компании Microsoft и их стремлением предоставить клиентам полноценное решение для управления базами данных под управлением их операционных систем.

Первая версия SQL Server была выпущена в 1989 году под названием SQL Server 1.0 для операционной системы OS/2, разработанной компанией IBM. Эта версия SQL Server была ориентирована на использование в клиент-серверных приложениях.

Следующая важная точка в истории MS SQL пришла в 1993 году с выпуском SQL Server 4.2, который был первой версией, выпущенной для платформы Windows NT. Это значительно расширило аудиторию продукта и укрепило его позиции на рынке.

За последующие годы SQL Server развивался, добавляя новые функции и улучшения. В 1995 году вышла версия SQL Server 6.0, в которой было внедрено множество новых возможностей, таких как хранимые процедуры, транзакции и триггеры.

С 2000 года Microsoft стала выпускать версии SQL Server с годовыми обновлениями. SQL Server 2000 был важным выпуском, который представил такие функции, как XML-поддержка, OLAP и аналитические возможности.

SQL Server 2005, выпущенный в 2005 году, был важным релизом, в котором были внедрены множество новых функций, включая улучшенную поддержку XML, встроенные отчеты и возможности для работы с большими объемами данных.

Дальнейшее развитие MS SQL продолжалось с выпуском SQL Server 2008, 2012, 2014, 2016 и последующих версий. Каждый релиз вносил новые функции и улучшения, такие как улучшенная производительность, расширенные возможности аналитики данных, поддержка облачных технологий и многое другое.

Одним из ключевых моментов в истории MS SQL был запуск SQL Server в облаке - SQL Azure, который предоставляет возможность использовать SQL Server в качестве облачной услуги.

Сегодня Microsoft SQL Server остается одним из ведущих реляционных СУБД на рынке. Он используется в широком спектре сфер, включая предприятия, веб-разработку, бизнес-аналитику и многое другое. SQL Server продолжает развиваться, внедряя новые технологии и функции, чтобы удовлетворить растущие потребности клиентов в области управления данными.

* + 1. **MySQL**

История MySQL — это история создания одной из самых популярных и распространенных открытых реляционных баз данных в мире.

MySQL была разработана в 1994 году шведскими программистами Монти Виденусом и Давидом Акерсоном. Их целью было создание СУБД с открытым исходным кодом, которая была бы простой в использовании, масштабируемой и доступной для широкого круга пользователей.

Первая версия MySQL была выпущена в 1995 году. Она предлагала базовые функции реляционной базы данных, такие как хранение данных в таблицах, поддержку индексов и простые операции SELECT, INSERT, UPDATE и DELETE.

С течением времени MySQL быстро приобрела популярность благодаря своей простоте, производительности и надежности. Большой вклад в популяризацию MySQL внесла решительная поддержка открытого исходного кода и распространение под лицензией GPL (General Public License).

Ключевыми моментами в истории MySQL были выпуски новых версий с расширенным функционалом и улучшениями производительности. В 2000 году была выпущена версия MySQL 3.23, которая внесла значительные улучшения в производительность и функциональность запросов. MySQL 4.0, выпущенная в 2003 году, добавила поддержку хранимых процедур и триггеров, а также другие расширенные возможности.

Однако, возможно, наиболее значимым событием в истории MySQL было приобретение компанией Sun Microsystems в 2008 году. Это придавало MySQL дополнительные ресурсы и укрепляло ее позиции на рынке.

В 2010 году компания Oracle приобрела Sun Microsystems, включая MySQL. Это вызвало опасения в сообществе открытого исходного кода относительно будущего MySQL под управлением корпорации Oracle. В ответ на это была создана отдельная организация под названием MariaDB, форк MySQL, который продолжает развиваться независимо.

MySQL по-прежнему активно развивается, и ее последние версии включают в себя множество новых функций и улучшений, таких как поддержка JSON, улучшенные возможности репликации и масштабируемости, а также интеграция с облачными платформами.

Сегодня MySQL остается одной из самых популярных баз данных в мире и широко используется веб-разработчиками, компаниями и организациями различных масштабов. Ее открытый исходный код и богатый функционал продолжают привлекать новых пользователей и обеспечивать ее популярность и востребованность в сообществе разработчиков.

* + 1. **PostgreSQL**

История PostgreSQL (или просто Postgres) весьма интересна и связана с академическими исследованиями, разработкой исходного кода с открытым доступом и постоянным развитием сообщества.

Всё началось в 1986 году, когда проект Postgres был начат Майклом Стоунбрейкером, профессором информатики в Берклиском университете, и его командой студентов. Их целью было создание следующего поколения СУБД, которая была бы наследником управления данными в стиле Ingres (предшественника Postgres) и одновременно расширяла бы его функциональность.

Первая версия Postgres (Post Ingres) была выпущена в 1989 году. Это была объектно-реляционная база данных, предлагавшая множество инновационных функций, таких как поддержка пользовательских типов данных, подзапросов, правил и многое другое.

В 1996 году, после выпуска версии 6.0, проект был переименован в PostgreSQL, чтобы подчеркнуть его продолжающееся развитие и сближение с SQL-стандартами.

Одним из важных моментов в истории PostgreSQL было создание PostgreSQL Global Development Group в 1996 году, объединившей разработчиков и пользователей PostgreSQL со всего мира в единое сообщество. Это способствовало активному обмену знаниями и опытом, а также совместной разработке и совершенствованию СУБД.

В последующие годы PostgreSQL продолжала развиваться, добавляя новые функции и улучшения, такие как поддержка транзакций, сохраняемые процедуры, триггеры, репликация и многое другое. Важными моментами стали выпуск версий 7.0 в 2000 году, 8.0 в 2005 году и 9.0 в 2010 году, каждая из которых вносила значительные улучшения и инновации.

Сегодня PostgreSQL остается одной из самых мощных и функциональных открытых реляционных баз данных в мире. Ее использование распространено в различных областях, включая веб-разработку, аналитику данных, геоинформационные системы, финансовые приложения и многое другое. Сообщество PostgreSQL продолжает активно разрабатывать и совершенствовать СУБД, обеспечивая ее актуальность и конкурентоспособность в современном мире информационных технологий.

* + 1. **Сравнение**

MySQL

* Особенности: MySQL известен своей простотой и быстротой. Он обеспечивает широкий набор функций, включая поддержку транзакций, индексацию и хранимые процедуры. MySQL также хорошо масштабируется и поддерживает высокие нагрузки.
* Применение: MySQL широко используется веб-разработчиками, начинающими компаниями и небольшими проектами, где важны простота использования и низкая стоимость.

PostgreSQL

* Особенности: PostgreSQL известен своей мощностью и расширяемостью. Он поддерживает множество расширений, включая геоданные, полнотекстовый поиск, JSON и многое другое. PostgreSQL также имеет продвинутые функции репликации и масштабируется хорошо.
* Применение: PostgreSQL часто используется в крупных предприятиях, где важны надежность, масштабируемость и расширяемость. Он также популярен в академических и научных исследованиях.

MS SQL

* Особенности: Microsoft SQL Server известен своей интеграцией с другими продуктами Microsoft и предоставлением различных инструментов для управления данными и аналитики. Он поддерживает широкий спектр функций, таких как хранимые процедуры, триггеры, OLAP и многое другое.
* Применение: Microsoft SQL Server часто используется в корпоративных средах и компаниях, работающих в экосистеме Microsoft. Он также популярен в разработке приложений на платформе .NET.

Oracle

* Особенности: Oracle Database славится своей надежностью, производительностью и масштабируемостью. Она обеспечивает широкий набор функций, включая продвинутые возможности управления транзакциями, средства мониторинга и управления данными, а также мощные средства аналитики.
* Применение: Oracle Database широко используется в крупных предприятиях, финансовом секторе, телекоммуникациях, государственном секторе и других областях, где требуется масштабируемая и высоконагруженная база данных с высокими требованиями к надежности и производительности.
  + 1. **Использование СУБД в реальной жизни**

Системы управления базами данных (СУБД) широко используются в реальной жизни в различных областях и сферах деятельности. Вот некоторые примеры использования СУБД

Электронная коммерция и интернет-торговля:

* СУБД используются для хранения информации о продуктах, заказах, клиентах и транзакциях в онлайн-магазинах. Например, данные о товарах, их описания, ценах и наличии хранятся в базе данных, а также информация о заказах, оплате и доставке.

Финансовая сфера:

* Банки, финансовые учреждения и компании используют СУБД для хранения и обработки финансовых данных, включая данные о клиентах, счетах, операциях, транзакциях и т. д. Это позволяет эффективно управлять финансовыми операциями, вести отчетность и анализировать данные.

Здравоохранение:

* Медицинские учреждения используют СУБД для хранения и управления медицинской информацией пациентов, историями болезней, результатами анализов, назначениями и прочими медицинскими данными. Это помогает ведению электронных медицинских записей, обеспечению конфиденциальности данных и повышению качества медицинского обслуживания.

Производство и складское хозяйство:

* Компании, занимающиеся производством и хранением товаров, используют СУБД для учета и управления запасами, отслеживания производственных процессов, управления поставками и т. д. Это помогает оптимизировать производственные операции, сокращать затраты и улучшать обслуживание клиентов.

Образование и наука:

* В учебных заведениях и исследовательских учреждениях СУБД используются для хранения и анализа образовательных данных, результатов исследований, библиотечных коллекций и других информационных ресурсов. Это позволяет эффективно управлять образовательными процессами, проводить анализ данных и делать научные открытия.

Телекоммуникации и информационные технологии:

* Компании, работающие в сфере телекоммуникаций и информационных технологий, используют СУБД для хранения и обработки информации о клиентах, сетях, устройствах, трафике и других аспектах своей деятельности. Это помогает обеспечивать качественное предоставление услуг связи, а также управлять и поддерживать информационные системы.

Это лишь несколько примеров использования СУБД в реальной жизни. Существует множество других отраслей и областей, где СУБД играют важную роль в организации и управлении данными, обеспечивая эффективное функционирование бизнеса и повышая его конкурентоспособность.

* 1. **Выбор стека технологий**

Наш проект представляет собой веб-приложение для создания SQL скриптов.

Требования к проекту:

* 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.

Выбранный стек технологий

Backend:

* Язык программирования: JavaScript (Node.js)
* Фреймворк: Express.js
* База данных: PostgreSQL

Frontend:

* Языки программирования: HTML, CSS, JavaScript
* Фреймворк: React.js

Другие технологии:

* Инструменты разработки: Git, VS Code, Postman

Обоснование выбора:

* Node.js и Express.js: Обеспечивают высокую производительность и масштабируемость.
* PostgreSQL: Предоставляет мощные средства для работы с данными и обеспечивает высокий уровень безопасности.
* React.js: Обеспечивают быструю и динамичную разработку пользовательского интерфейса, а также удобное управление состоянием приложения.

Этот стек технологий был выбран исходя из требований проекта, опыта команды разработчиков и с учетом лучших практик разработки веб-приложений.

* + 1. **Backend**

Node.js позволяет использовать JavaScript как язык программирования как на клиентской, так и на серверной стороне. Это обеспечивает единый язык программирования на всех уровнях приложения, что упрощает его разработку и поддержку.

С использованием Node.js мы создаем серверную часть приложения, которая обрабатывает запросы от клиентской стороны, выполняет логику приложения, взаимодействует с базой данных и возвращает ответы клиенту.

Мы используем фреймворк Express.js для упрощения создания API и обработки маршрутов в нашем приложении. Express.js предоставляет набор готовых инструментов для обработки HTTP-запросов, маршрутизации и управления обработкой запросов и ответов.

Node.js использует менеджер пакетов npm для управления зависимостями проекта. Мы можем легко устанавливать, обновлять и удалять пакеты через npm, что упрощает добавление нового функционала и поддержку проекта.

Мы используем Node.js для взаимодействия с базой данных PostgreSQL. С помощью библиотеки для работы с PostgreSQL, таких как pg или Sequelize, мы можем выполнять запросы к базе данных, получать и обрабатывать данные, а также сохранять изменения.

Мы используем Node.js для настройки и обработки CORS (Cross-Origin Resource Sharing) с помощью пакета cors. Это обеспечивает безопасное взаимодействие между клиентской и серверной сторонами приложения, предотвращает возможные атаки и обеспечивает соблюдение стандартов безопасности.

Использование Node.js в этом проекте позволяет нам создавать масштабируемое, производительное и безопасное веб-приложение, обеспечивая удобное взаимодействие между клиентской и серверной частями.

Сравним с аналогами.

Сравнение с Java (Spring Boot):

* Преимущества: Java обладает высокой производительностью и масштабируемостью. Spring Boot предоставляет много готового функционала и инструментов для создания веб-приложений.
* Недостатки: Сложность в настройке и развертывании. Могут потребоваться более долгие сроки разработки.

Сравнение с Java (Spring Boot):

* Преимущества: Простота и быстрота разработки. Обширная экосистема библиотек и фреймворков. Django предоставляет готовые инструменты для решения множества задач.
* Недостатки: Python может иметь некоторые ограничения в производительности. Некоторые фреймворки (например, Django) могут быть более "тяжеловесными" по сравнению с Express.js.

В целом, наш бэкенд на основе Node.js и Express.js предоставляет гибкое и эффективное решение для создания веб-приложений, обеспечивая хорошую производительность и надежность.

* + 1. **Frontend**

React.js выбран для разработки пользовательского интерфейса, так как это мощный и популярный фреймворк, обеспечивающий создание модульных и масштабируемых компонентов. React также предлагает виртуальный DOM и эффективный механизм обновления интерфейса, что повышает производительность приложения.

React Router используется для управления маршрутизацией в приложении. Это позволяет создавать одностраничные приложения с множеством маршрутов и предоставлять пользователям плавный и интуитивно понятный опыт использования.

Axios используется для работы с API и выполнения HTTP-запросов. CSS Modules и Styled Components обеспечивают удобный подход к стилизации компонентов. Jest и React Testing Library используются для тестирования компонентов и функциональности приложения. JSON Web Tokens (JWT) используются для безопасной аутентификации пользователей.

CSS используется для стилизации HTML-элементов и создания визуального оформления пользовательского интерфейса. CSS определяет внешний вид элементов, их размеры, цвета, шрифты, расположение и другие аспекты дизайна.

HTML используется для создания структуры и содержимого веб-страниц. Он определяет различные элементы, такие как заголовки, параграфы, списки, ссылки, изображения и другие, которые составляют внешний вид и функциональность вашего веб-приложения.

HTML5 представляет собой последнюю версию HTML со стандартными и расширенными элементами, атрибутами и API. Он предоставляет новые возможности, такие как аудио и видео элементы, формы с валидацией на стороне клиента, семантические элементы и другие, что упрощает разработку и улучшает пользовательский опыт.

Давайте сравним React с некоторыми его аналогами.

Vue.js:

* React: Основное внимание уделяется компонентному подходу и виртуальному DOM, что делает его мощным для разработки крупных и сложных приложений.
* Vue.js: Имеет более простой синтаксис и обучение, что делает его хорошим выбором для быстрого создания малых и средних проектов. У Vue.js более гибкая система реактивности.

Angular:

* React: Фокусируется на простоте и гибкости, позволяя разработчикам выбирать инструменты и библиотеки, которые им больше нравятся. Обладает большим сообществом и экосистемой плагинов.
* Angular: Предоставляет все необходимые инструменты из коробки, что делает его более полным решением для крупных и корпоративных проектов. Обладает встроенной системой маршрутизации и управления состоянием.

Svelte:

* React: Работает на основе виртуального DOM, что может потреблять больше памяти и ресурсов по сравнению с Svelte. Однако, React обладает большим сообществом и богатой экосистемой.
* Svelte: Предлагает компиляцию в чистый JavaScript во время сборки проекта, что обеспечивает более быструю загрузку и выполнение кода. Svelte более компактен и быстр в работе, но имеет меньшую экосистему и сообщество, чем React.

Ember.js:

* React: Предоставляет гибкий и модульный подход к разработке приложений, что делает его хорошим выбором для множества проектов, от небольших до крупных. React обладает большим сообществом и богатой экосистемой.
* Ember.js: Сосредоточен на конвенциях и нормах, что делает его отличным выбором для крупных и долгосрочных проектов. Ember.js имеет много встроенных функций и инструментов для быстрой разработки.

Выбор между React и его аналогами зависит от конкретных требований проекта, уровня опыта разработчика и предпочтений в используемых инструментах и подходах к разработке.

* + 1. **Инструменты разработки**

Visual Studio Code (VS Code) — это легкий, расширяемый и мощный текстовый редактор, разработанный Microsoft. Вот некоторые ключевые особенности и преимущества VS Code:

VS Code поддерживает множество языков программирования, включая JavaScript, TypeScript, Python, Java, C++, PHP, HTML, CSS и многие другие. Это делает его универсальным инструментом для разработки веб- и прикладного программного обеспечения.

VS Code предлагает широкий набор функций редактирования кода, включая подсветку синтаксиса, автодополнение, автоматическое выравнивание, быстрые исправления, рефакторинг и многое другое. Эти функции значительно улучшают процесс написания кода и увеличивают производительность разработчика.

VS Code интегрируется с популярными системами управления версиями, такими как Git, что позволяет легко работать с репозиториями, отслеживать изменения и совершать коммиты, не покидая редактор.

VS Code обладает огромным экосистемой расширений, которые позволяют настраивать и расширять функционал редактора под конкретные потребности разработчика. От поддержки языков программирования до интеграции с различными инструментами и сервисами - расширения делают VS Code универсальным инструментом разработки.

VS Code занимает небольшой объем памяти и ресурсов системы, что позволяет ему работать быстро и эффективно даже на слабых компьютерах. Он имеет низкий порог входа и не требует значительных ресурсов для установки и использования.

VS Code доступен для Windows, macOS и Linux, что делает его доступным для широкого круга разработчиков, работающих на различных платформах.

В целом, Visual Studio Code — это мощный и универсальный инструмент разработки, который обладает всеми необходимыми функциями для написания кода, управления проектами и повышения производительности разработчика.

Git — это распределенная система управления версиями, которая используется для отслеживания изменений в исходном коде программного обеспечения. Вот основные особенности и принципы работы с Git:

Git является распределенной системой, что означает, что каждый разработчик имеет полную копию репозитория на своем компьютере. Это позволяет работать в автономном режиме и управлять версиями кода даже без доступа к центральному серверу.

Одна из основных особенностей Git - это возможность создания легких и быстрых веток (branches) для разработки новых функций или исправления ошибок. После завершения работы над веткой изменения могут быть слиты (слияние) обратно в основную ветку (обычно это master или main).

Git отслеживает и сохраняет историю изменений каждого файла в репозитории. Это позволяет в любой момент вернуться к предыдущей версии файла или откатить все изменения в проекте до определенного момента.

Git облегчает совместную работу над проектом, позволяя разработчикам работать над разными частями кода независимо друг от друга. Каждый разработчик может создавать собственные ветки для работы над своими задачами, а затем объединять изменения в общий репозиторий.

Git поддерживает работу с удаленными репозиториями, такими как GitHub, GitLab или Bitbucket. Это позволяет разработчикам обмениваться изменениями, делать код доступным для других участников проекта и вести совместную работу даже удаленно.

Git может быть использован через командную строку (терминал) или графический интерфейс, такой как GitHub Desktop или Sourcetree. Командная строка обеспечивает большую гибкость и контроль, в то время как графический интерфейс может быть более удобным для начинающих разработчиков.

Git имеет обширную документацию и широкое сообщество пользователей, что делает его легким для изучения и использования. Есть множество ресурсов, учебных пособий и форумов, где можно найти ответы на вопросы и помощь при работе с Git.

Git — это мощный инструмент, который обеспечивает эффективное управление версиями кода, совместную работу над проектами и отслеживание изменений в исходном коде

* 1. **Проектирование приложения**
     1. **Проектирование базы данных**
     2. **Проектирование интерфейса**

Проектирование интерфейса веб-приложения — это процесс создания пользовательского опыта (UX) и пользовательского интерфейса (UI), который обеспечивает удобство использования, эффективность и привлекательность приложения для его целевой аудитории. Вот некоторые ключевые шаги проектирования интерфейса веб-приложения:

* Проведём исследование нашей целевой аудитории, чтобы понять их потребности, предпочтения, цели и поведение. Проанализируем данные пользовательских опросов, аналитики сайта, отзывов пользователей и конкурентов.
* Определим основные цели, которые пользователи будут достигать при использовании нашего приложения, а также задачи, которые им придется выполнить. Это позволит сосредоточиться на создании интерфейса, который поможет пользователям эффективно решать их задачи.
* Создадим описание типичных пользовательских персонажей (personas), которые представляют нашу целевую аудиторию, и опишите их основные характеристики, потребности и цели. Затем разработайте сценарии использования (user scenarios), которые показывают, как эти персонажи будут взаимодействовать с вашим приложением для достижения своих целей.
* Определим структуру и организацию информации в нашем приложении. Разработаем карту сайта, которая покажет основные разделы и страницы приложения, и построим дерево навигации, которое определит логическую последовательность переходов между ними.
* На основе информационной архитектуры создадим макеты (wireframes) и прототипы нашего приложения. Макеты помогут определить расположение элементов интерфейса на странице, а прототипы позволят проверить работу основных функций и взаимодействий.
* Разработаем дизайн пользовательского интерфейса (UI) с учетом стилей, цветовой палитры, типографии, иконок и других элементов дизайна. Убедимся, что дизайн соответствует нашему приложению и создает приятное визуальное впечатление.
* Добавим интерактивность к нашему интерфейсу, такую как анимации, переходы между страницами, всплывающие окна и элементы управления. Это поможет сделать наше приложение более привлекательным и удобным в использовании.
* Проведём тестирование нашего интерфейса с реальными пользователями, чтобы оценить его удобство использования и эффективность. Полученные обратные связи используем для оперативного улучшения интерфейса, исправления ошибок и оптимизации пользовательского опыта.
* Задокументируем все компоненты интерфейса, включая макеты, прототипы, дизайн и стилизацию. После завершения проектирования наш интерфейс, чтобы он был готов к использованию нашими пользователями.

Проектирование интерфейса веб-приложения требует внимательного анализа пользовательских потребностей, тщательного планирования и итеративной разработки, чтобы создать удобный и привлекательный пользовательский опыт.

* + - 1. **Опрос целевой аудитории**

Вопросы для нашего опроса:

1.Какой уровень знаний у вас в области баз данных и SQL?

2.С какими базами данных вы работали ранее?

3.Какую часть баз данных вы считаете самой сложной для проектирования?

4.Какие инструменты или программное обеспечение вы предпочитаете для проектирования баз данных?

5.Какие функции вы ожидаете от приложения для проектирования баз данных и генерации SQL скриптов?

6.Какие дополнительные инструменты или функции бы вы хотели видеть в таком приложении?

7.Какой формат вы предпочитаете для генерации SQL скриптов?

* + - 1. **Создание макетов**

В начале проанализируем дизайн сервиса конкурента, а конкретно DB Designer.

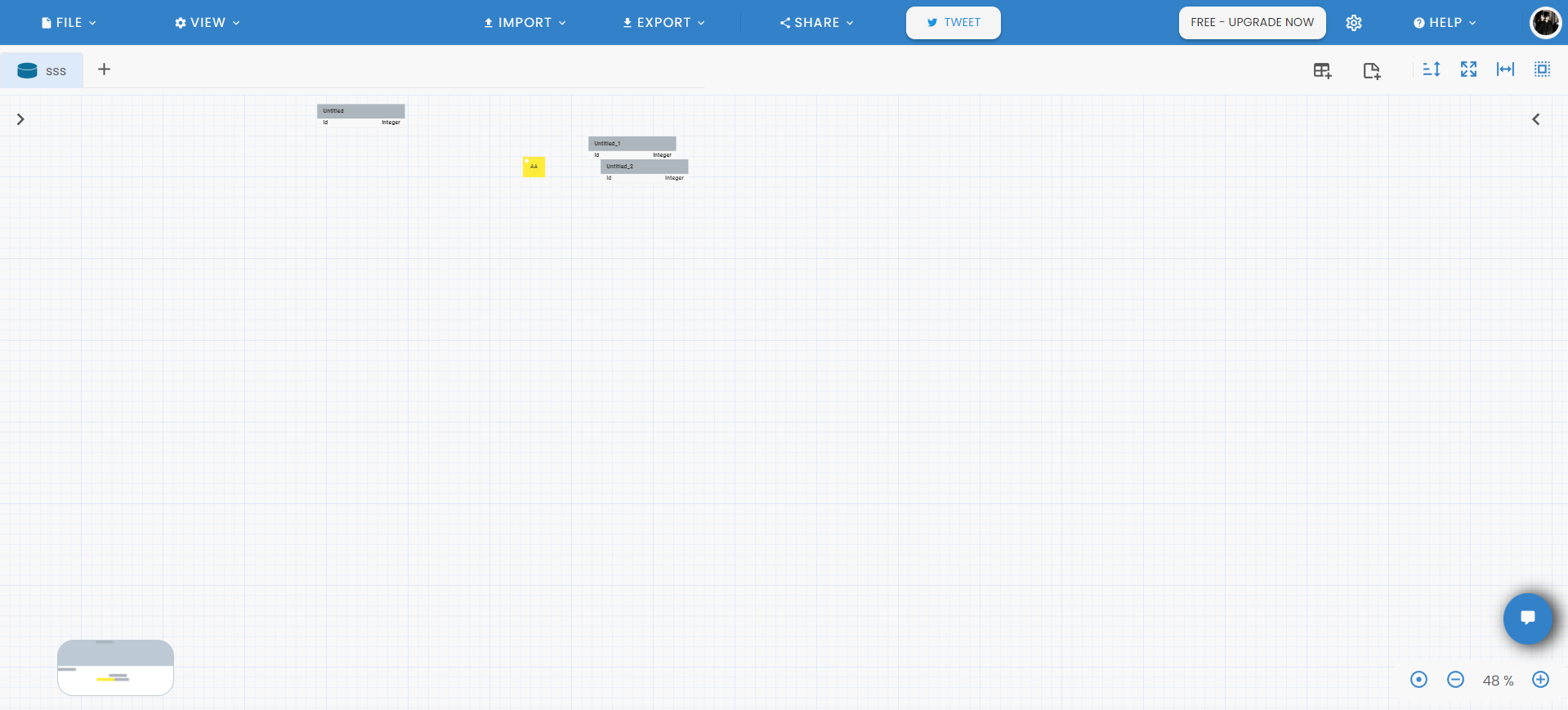


Рис. 1 – Интерфейс DB Designer

Попытаемся проанализировать интерфейс DB Designer, мы видим header на котором находятся несколько элементов file, import, export, share, tweet, setting, help, profile.



Рис. 2 – header DB Designer

Я решил удалить, пункт share и setting.

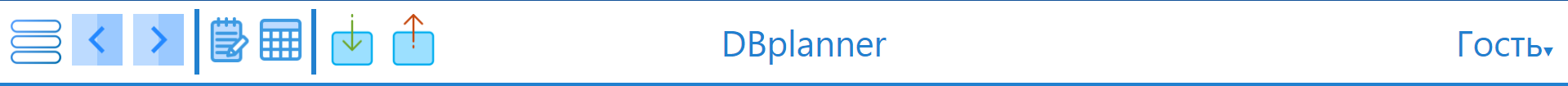


Рис. 3 – header DBplanner

Вот мой макет header, на нем мы видим 9 элементов, всплывающее меню, шаги назад и вперёд, стикер и таблица, импорт и экспорт, название, профиль.

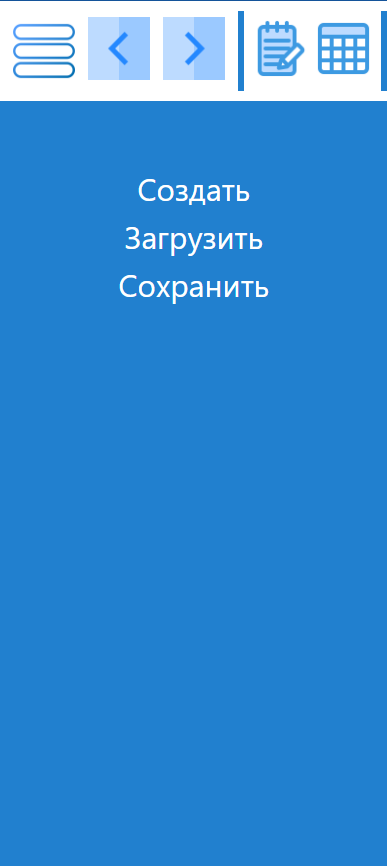


Рис. 4 – всплывающее меню

В нашем всплывающее меню имеется 3 пункта:

* Создать
* Загрузить
* Сохранить

Посмотрим на profile

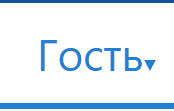


Рис. 5 – profile в режиме гость

Изначально он находится в гостевом режиме, но мы при наведении нам будет предложено зарегистрироваться или войти при наличие аккаунта.



Рис. 6 – profile при наведении

При нажатие войти или зарегистрироваться будет всплывать окно в зависимости от нашего выбора, где нам будет предложено войти или зарегистрироваться соответственно.

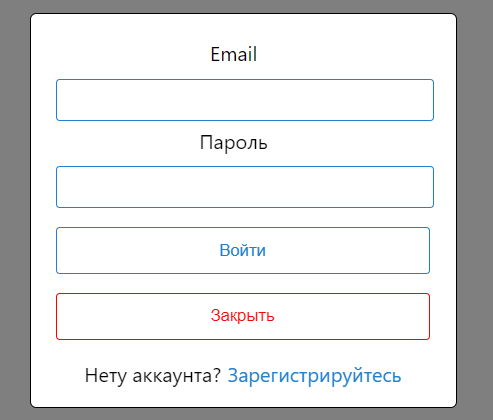


Рис. 7 – Окно авторизации

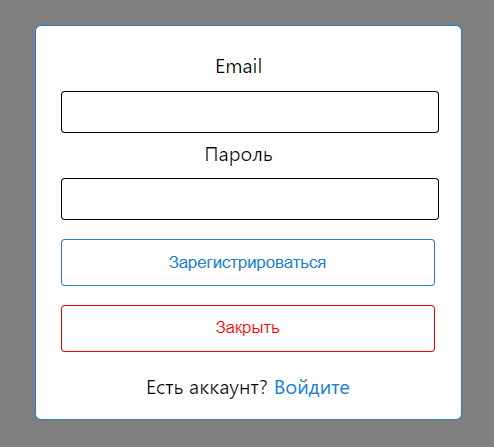


Рис. 8 – Окно регистрации

* + 1. **А**