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** |
| наименование института |

|  |  |  |
| --- | --- | --- |
|  |  | Допускаю к защите  Руководитель ООП |
|  |  | А.Ф. Аношко |
| подпись |  | И.О. Фамилия |

|  |
| --- |
| **Разработка программного средства для проектирования модели** |
| **данных** |
| наименование темы |

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к выпускной квалификационной работе бакалавра

Программа бакалавриата

Вычислительные машины, комплексы, системы и сети

по направлению подготовки

09.03.01 Информатика и вычислительная техника

|  |  |  |
| --- | --- | --- |
| 0.000.00.00 | | ПЗ |
|  | |  |
| Разработал студент  Группы ЭВМб-20-1 |  |  |  | Е.В. Прохоров | | |
|  |  | подпись |  | И.О. Фамилия | | |
| Руководитель |  |  |  | | А.С. Дорофеев | |
|  |  | подпись |  | | И.О. Фамилия | |
| Нормоконтроль |  |  |  | | А.С. Дорофеев | |
|  |  | подпись |  | | И.О. Фамилия | |

Иркутск 2024 г.

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** |
| наименование института |

УТВЕРЖДАЮ

Директор института ИТ и АД

\_\_\_\_\_\_\_\_\_\_\_\_\_А.С. Говорков

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2024 г.

ЗАДАНИЕ

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| на выпускную квалификационную работу студенту | | | | | Прохорову Евгению | | | |
| Викторовичу | | | | | | | | |
| группы | | ЭВМб-20-1 |  | | | | | |
| 1. Тема работы: Разработка программного средства для проектирования модели данных | | | | | | | | |
|  | | | | | | | | |
| Утверждена приказом по университету от | | | |  | | № |  |  |
| 1. Срок представления студентом законченной работы в ГЭК | | | | | | |  | |
| 1. Исходные данные | | | | | | | | |
|  | 3.1 СТО 005-2020 «Система менеджмента качества. Учебно- | | | | | | | |
|  | методическая деятельность. Оформление курсовых проектов (работ) и | | | | | | | |
|  | выпускных квалификационных работ технических направлений | | | | | | | |
|  | подготовки и специальностей» | | | | | | | |
|  | 3.2 Материалы преддипломной практики | | | | | | | |
| 1. Содержание расчетно-пояснительной записки (перечень подлежащих разработке вопросов): | | | | | | | | |
|  | 4.1 Анализ языка SQL | | | | | | | |
|  | 4.2 Выбор стека технологий | | | | | | | |
|  | 4.3 Проектирование интерфейса | | | | | | | |
|  | 4.4 Проектирование базы данных | | | | | | | |
|  | 4.5 Создание программного средства | | | | | | | |
|  | 4.6 Тестирование | | | | | | | |
| 5 | Перечень графического материала (с указанием обязательных чертежей) | | | | | | | |
|  | 5.1 Презентационные материалы | | | | | | | |

Календарный план

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Разделы | Месяцы и недели | | | | | | | | | | | | |
| апрель | | | | май | | | | | июнь | | | |
| Введение |  |  | **+** |  |  |  |  |  |  |  |  |  |  |
| 1. Анализ языка SQL |  |  |  | **+** |  |  |  |  |  |  |  |  |  |
| 1. Выбор стека технологий |  |  |  |  | **+** |  |  |  |  |  |  |  |  |
| 1. Проектирование интерфейса |  |  |  |  |  | **+** | **+** |  |  |  |  |  |  |
| 1. Проектирование базы данных |  |  |  |  |  | **+** | **+** |  |  |  |  |  |  |
| 1. Создание программного средства |  |  |  |  |  |  | **+** | **+** | **+** |  |  |  |  |
| 1. Тестирование |  |  |  |  |  |  |  |  |  | **+** |  |  |  |
| Заключение |  |  |  |  |  |  |  |  |  | **+** |  |  |  |
| Оформление пояснительной записки |  |  | **+** | **+** | **+** | **+** | **+** | **+** | **+** | **+** | **+** |  |  |
| Подготовка к защите ВКР |  |  |  |  |  |  |  |  |  | **+** | **+** | **+** | **+** |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Дата выдачи задания | | | « | 22 | | | » | | апреля 2024 г. | | | | | |  | | | |
|  | | | | | | | | | |  | | | | | | | | |
|  | Руководитель выпускной работы бакалавра | | | | | | | | |  | | | | | | А.С. Дорофеев | | |
|  |  | | | | | | | | | подпись | | | | | | И.О. Фамилия | | |
|  |  | | | | | | | | |  | | | | | | | | |
|  | Руководитель ООП | | | | | | | | |  | | | | | | А.Ф. Аношко | | |
|  |  | | | | | | | | | подпись | | | | | | И.О. Фамилия | | |
|  |  | | | | | | | | |  | | | | | | | | |
|  | Задание принял к исполнению студент | | | | | | | | | |  | | | | | Е.В. Прохоров | | |
|  |  | | | | | | | | | | подпись | | | | | И.О. Фамилия | | |
|  | План выполнен |  | | | | | | | | | | | | | | | | |
|  |  | (полностью, не полностью) | | | | | | | | | | | | | | | | |
|  | Руководитель работы | | | « |  | » | |  | | | | 2024 г. |  |  | | | А.С. Дорофеев |
|  |  | | |  |  |  | | дата | | | |  | подпись |  | | | И.О. Фамилия |

Оглавление

[Введение 5](#_Toc165379584)

[1 Анализ языка SQL 6](#_Toc165379585)

[1.1 История языка SQL 6](#_Toc165379586)

[1.2 Современные языки SQL 6](#_Toc165379587)

[1.2.1 Oracle 6](#_Toc165379588)

[1.2.2 Microsoft SQL Server (MS SQL) 7](#_Toc165379589)

[1.2.3 MySQL 8](#_Toc165379590)

[1.2.4 PostgreSQL 9](#_Toc165379591)

[1.3 Синтаксис языка SQL 10](#_Toc165379592)

[1.4 Скрипты для создания базы данных 11](#_Toc165379593)

[2 Выбор стека технологий 11](#_Toc165379594)

[2.1 Backend 12](#_Toc165379595)

[2.2 Frontend 13](#_Toc165379596)

[2.3 База данных 14](#_Toc165379597)

[2.4 Инструменты разработки 16](#_Toc165379598)

[2.4.1 Visual Studio Code 16](#_Toc165379599)

[2.4.2 Git 17](#_Toc165379600)

[3 Проектирования интерфейса 17](#_Toc165379601)

[3.1 Исследование и анализ 17](#_Toc165379602)

### 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.

### Анализ языка SQL

### 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.

### Современные языки SQL

### 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.

### 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.

### 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.

### 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.

### Синтаксис языка SQL

SQL, или язык структурированных запросов (Structured Query Language), это стандартный язык программирования для работы с реляционными базами данных. Он предоставляет набор инструкций для создания, изменения, управления и извлечения данных из базы данных. Вот основные компоненты синтаксиса SQL:

1. Команды:

* DDL (Data Definition Language): Команды для определения структуры базы данных, такие как CREATE, ALTER, DROP.
* DML (Data Manipulation Language): Команды для управления данными, такие как SELECT, INSERT, UPDATE, DELETE.
* DCL (Data Control Language): Команды для управления правами доступа к данным, такие как GRANT, REVOKE.

1. Ключевые слова:

* SELECT: Извлечение данных из базы данных.
* INSERT: Добавление данных в базу данных.
* UPDATE: Обновление существующих данных в базе данных.
* DELETE: Удаление данных из базы данных.
* CREATE: Создание новых объектов базы данных, таких как таблицы, индексы и т. д.
* ALTER: Изменение структуры объектов базы данных.
* DROP: Удаление объектов базы данных.
* И другие, в зависимости от конкретной реализации SQL.

1. Комментарии:

* SQL поддерживает комментарии для документирования кода. Однострочные комментарии начинаются с двойного дефиса --, а многострочные комментарии заключаются между /\* и \*/.

1. Таблицы:

* База данных состоит из таблиц, которые хранят данные в виде строк и столбцов.

1. Столбцы:

* Каждая таблица состоит из столбцов, которые определяют типы данных, которые могут храниться в каждой строке.

1. Строки:

* Строки представляют отдельные записи или элементы данных в таблице.

1. Условия:

* Условия используются для фильтрации данных в операторе SELECT с использованием ключевого слова WHERE.

1. Функции:

* SQL предоставляет множество встроенных функций, таких как арифметические функции (SUM, AVG), функции работы со строками (CONCAT, UPPER), функции даты и времени (NOW, DATE\_FORMAT) и т. д.

1. Соединения (JOIN):

* SQL позволяет объединять данные из нескольких таблиц по определенным условиям с помощью оператора JOIN.

1. Группировка и агрегация:

* SQL поддерживает группировку данных с использованием оператора GROUP BY и агрегацию данных с использованием функций, таких как SUM, AVG, COUNT, MIN, MAX.

1. Сортировка:

* Для упорядочивания результатов запроса SQL используется оператор ORDER BY.

Это лишь краткий обзор основного синтаксиса SQL. Существует много различных реализаций SQL, таких как MySQL, PostgreSQL, SQLite, и каждая из них может иметь свои особенности и дополнения к стандарту SQL.

### Скрипты для создания базы данных

1. Создание структуры базы данных:

Скрипты для создания баз данных необходимы для определения структуры хранения данных в проекте. Это включает в себя определение таблиц, их столбцов, типов данных, ограничений, а также взаимосвязей между таблицами (например, внешние ключи).

1. Обеспечение целостности данных:

При создании таблиц в базе данных можно определить различные ограничения, такие как уникальность значений, ограничения на внешние ключи и т. д. Это позволяет обеспечить целостность данных, предотвращая некорректные или недопустимые значения.

1. Упрощение развертывания и миграций:

Создание скриптов для создания баз данных также упрощает процесс развертывания вашего проекта на новых серверах или его миграции на другие базы данных. Путем выполнения этих скриптов можно быстро восстановить структуру базы данных в новом окружении.

1. Документация и понимание проекта:

Скрипты для создания баз данных также могут служить документацией к вашему проекту, описывая его структуру и отношения между данными. Это упрощает понимание проекта другими разработчиками или членами вашей команды.

### 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.

### Backend

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.

### Frontend

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.

### База 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.

1. 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.

1. 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.

1. 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.

### Инструменты разработки

### 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.

### Проектирования интерфейса

### 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 освоимым 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.

Изучение конкурентов:

1. MySQL Workbench:

Плюсы:

* + Мощный функционал для проектирования баз данных, включая создание схем, определение отношений, генерацию SQL-кода и т. д.
  + Интеграция с MySQL, что облегчает развертывание и управление базами данных.
  + Бесплатность и открытый исходный код.

Минусы:

* + Ограниченность использования только с MySQL, что может быть недостаточно для проектов, использующих другие СУБД.
  + Интерфейс может показаться сложным для новичков из-за большого количества функций.

1. Microsoft Visio:

Плюсы:

* + Широкий спектр возможностей для создания различных типов диаграмм, включая диаграммы баз данных.
  + Интеграция с другими продуктами Microsoft, такими как Microsoft Office.

Минусы:

* + Платная лицензия, что может быть проблемой для некоммерческих или индивидуальных пользователей.
  + Не является специализированным инструментом для проектирования баз данных, что может привести к ограничениям в функциональности.

1. Lucidchart:

Плюсы:

* + Онлайн-приложение с удобным интерфейсом, доступным из любого браузера.
  + Различные шаблоны и элементы для создания ER-диаграмм и других типов диаграмм.

Минусы:

* + Ограничения в бесплатной версии, включая ограниченное количество диаграмм и отсутствие некоторых функций.
  + Не такой широкий набор функций для проектирования баз данных, как у специализированных инструментов.

1. draw.io (diagrams.net):

Плюсы:

* + Бесплатное и открытое программное обеспечение.
  + Широкий набор элементов для создания диаграмм, включая ER-диаграммы.
  + Возможность экспорта в различные форматы.

Минусы:

* + Отсутствие интеграции с конкретными СУБД, что требует ручного создания SQL-кода.
  + Отсутствие онлайн-синхронизации и совместной работы в реальном времени в бесплатной версии.

1. Dbdiagram.io:

Плюсы:

* + Простота использования благодаря использованию языка разметки Markdown.
  + Онлайн-инструмент с возможностью доступа из любого браузера.

Минусы:

* + Ограниченные возможности по сравнению с более продвинутыми инструментами, такими как MySQL Workbench или Lucidchart.
  + Отсутствие некоторых функций, таких как генерация SQL-кода, в бесплатной версии.

### Прототипирование

Начнём с базового прототипа нашего интерфейса.

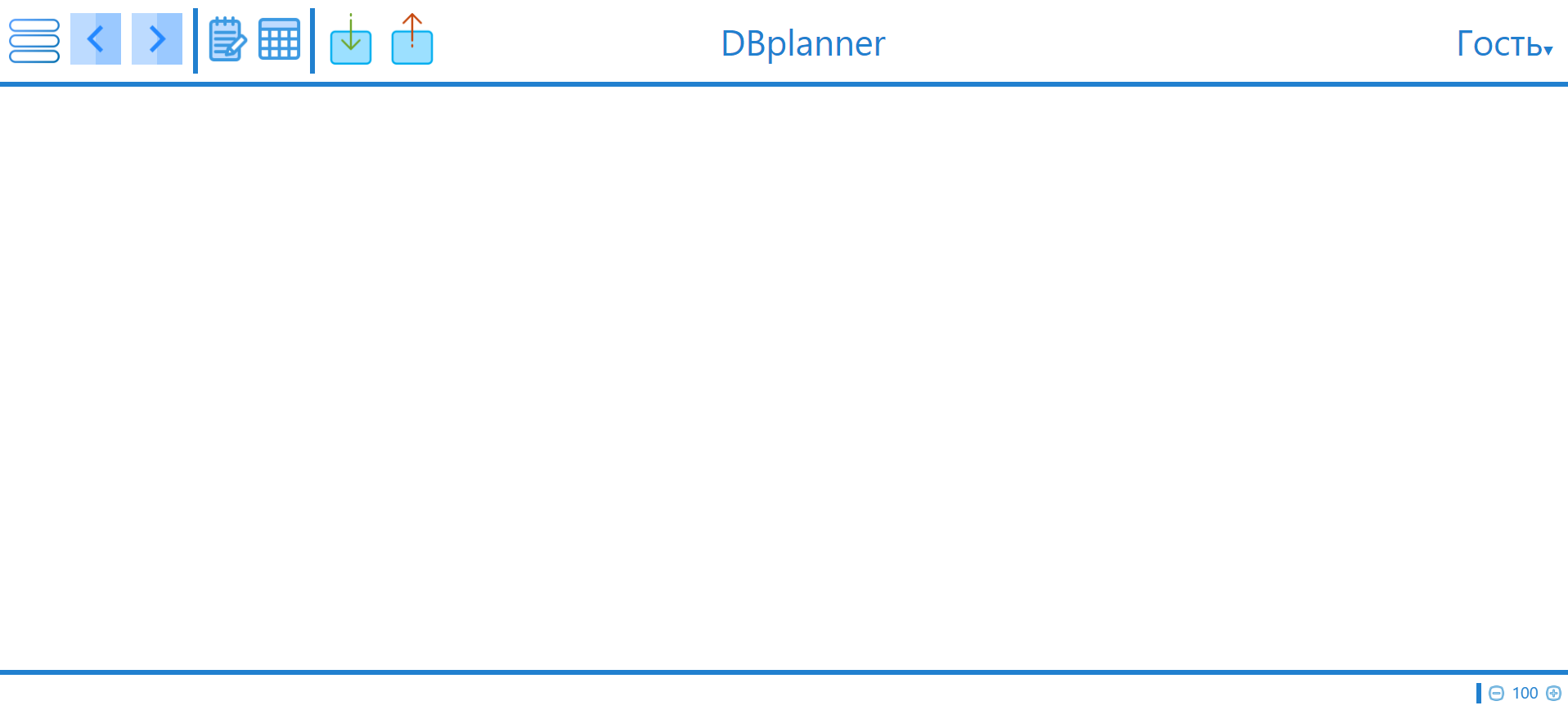


Рис. 1 – Прототип интерфейса

Далее разберем его отдельные элементы.

В низу находится Footer с кнопками изменения масштаба.



Рис. 2 – Прототип Footer



Рис. 3 – Прототип кнопок масштабирования

Кнопки масштабирования нужно, чтобы отдалять или приближать рабочее пространство и не взирая на количество таблиц комфортно работать с ними.

В центре располагается рабочее пространство



Рис. 4 – Прототип рабочего пространства

Наверху располагается Header на котором находятся иконки выполняющие различные функции.



Рис. 5 – Прототип header

Слева располагается иконка открывающегося меню



Рис. 6 – Прототип иконки открывающегося меню

В открывающемся меню находятся три пункта создать, загрузить и сохранить.

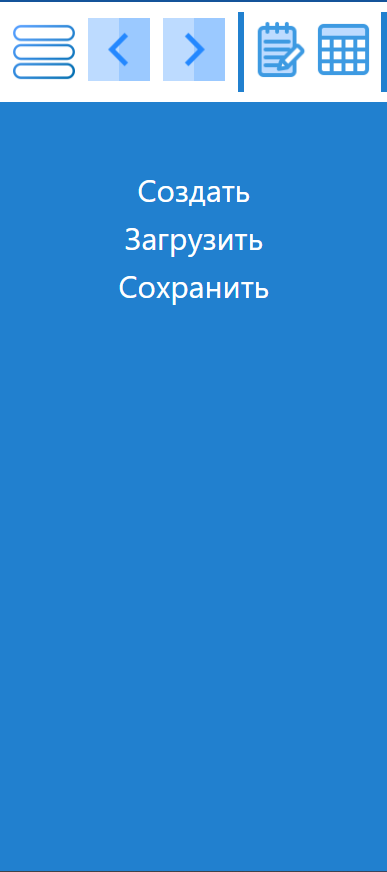


Рис. 7 – Прототип открывающегося меню

Правее находятся иконки для отмены действий и перемещения между ними.



Рис. 8 – Прототип иконок действий

Правее находятся иконки для вызова стикера для заметок и вызов окна создания таблицы.



Рис. 9 – Прототип иконок заметок и таблицы

На стикере мы можем написать какую-то информацию необходимую для нас.

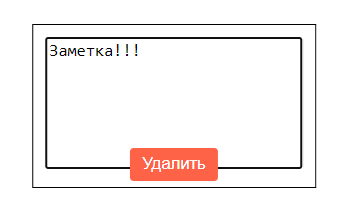


Рис. 10 – Прототип стикера с заметками

На прототипе окна для создания таблиц мы можем ввести название и выбрать тип данных, основной ключ, уникальное поле, автоинкремент и выбрать внешний ключ.

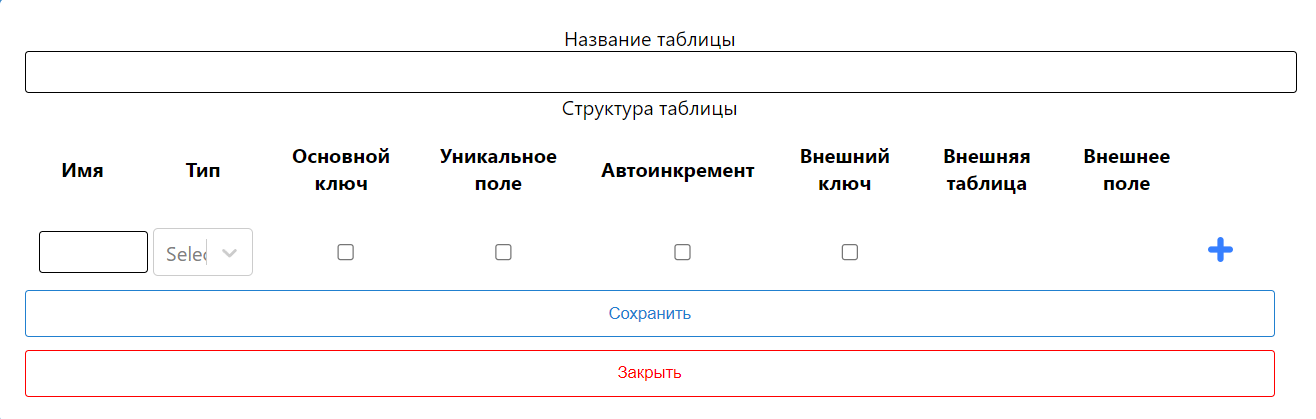


Рис. 11 – Прототип окна создания таблицы

Далее идут иконки импорта и экспорта.

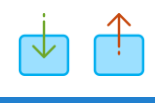


Рис. 12 – Иконки импорта и экспорта.

В окне импорта мы можем выбрать тип SQL и сгенерировать скрипт.

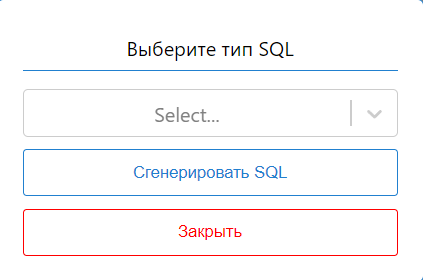


Рис. 13 – Окно экспорта.

В центре находится название нашего приложения.



Рис. 14 – Название приложения

В крайнем правом углу находится система регистрации



Рис. 15 – Система регистрации до входа

После наведения на надпись гость нам откроется два варианта входа и регистрации



Рис. 16 – Система авторизации

После выбора одного из двух вариантов нам откроется окно входа и регистрации соответственно.

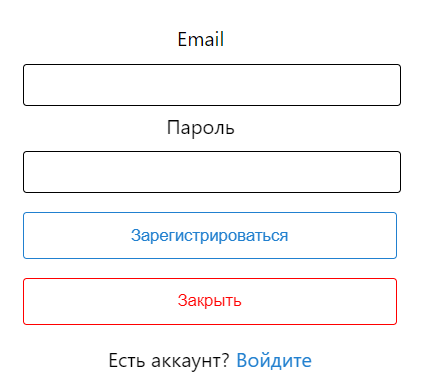


Рис. 17 – Окно регистрации

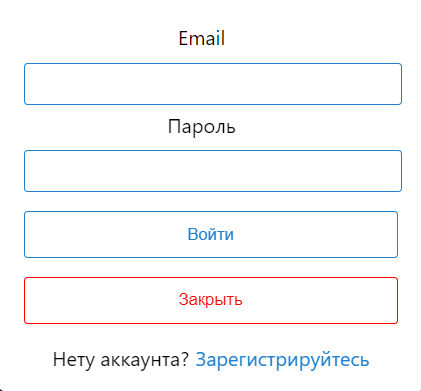


Рис. 17 – Окно входа

### Проектирование базы данных