СОДЕРЖАНИЕ

ВВЕДЕНИЕ

На сегодняшний день, мы все чаще записываем и сохраняем информацию, для дальнейшего изучения и использования. Это обусловлено большим количеством информации на просторах интернета и нашей жизни. И для большинства людей это является проблемой, так как требуется запоминать и записывать большое количество разного рода информацию на разных носителях.

Для того чтобы не потерять, не забыть самому и поделиться с другими информацией используются заметки.

Ведение заметок является неотъемлемой частью жизни современных людей. Однако не все могут позволить носить с собой записную книжку или не всегда она оказывается под рукой. Поиск нужных заметок в большом количестве данных занимает много времени, а также не всегда ты сможешь отыскать нужную тебе информацию. У записной книжки есть очень важный минус, что количество записей ограничено, также бумажную записную книжку можно забыть и потерять.

В связи с развитием интернета и создания большого количества веб-приложений, каждый пользователь может воспользоваться ими из любой части мира.

Одним из вариантов решения проблемы с ношением записных книжек и ограничением места, является использования специализированных веб-приложений, позволяющие оформлять и сохранять собственные заметки онлайн.

Существуют множество веб-приложений для ведения заметок, отличающиеся между собой: подходом формирования заметки, дизайном, удобством использования, а также ценой. Все они в каком-то виде пытаются заменить бумажные носители и предоставить удобства использования.

Целью данного дипломного проекта является сбор и анализ информации существующих аналогов в этой области и созданием собственного веб-приложения для управления и хранения пользовательских заметок. Данное программное обеспечение позволит сохранять свои заметки и легко искать информацию по заметкам, а также делиться этой информацией с другими людьми.

1 АНАЛИЗ ЛИТЕРАТУРЫ ПО ТЕМЕ ДИПЛОМНОГО ПРОЕКТА

1.1 Описание предметной области

В ходе увеличения объема информации и количества дел, появилась необходимость записывать всю эту информацию, для дальнейшего хранения упорядочивания и использования.

Для структурирования и хранения информации без использования интернета используются: меловые и маркерные доски, записные книжки.

Но с появлением интернета и его развития начали создавать различные программы для структурирования и записи информации.

Программы можно разделить на следующие виды по типу предоставляемого функционала:

– электронная доска;
– написание заметок;
– интеллект-карта.

К виду электронных досок можно отнести электронная интерактивная доска.

Интерактивная доска — это большой интерактивный экран в виде белой магнитно-маркерной доски. Интерактивная доска может быть представлена как автономным компьютером с большим сенсорным экраном, так и подключаемым к ноутбуку устройством, объединяющим проектор и сенсорную панель. Интерактивные доски используются в школьных кабинетах, переговорных, залах для групповых занятий, комнатах для дистанционного обучения и других помещениях [1].

К виду написание заметок можно отнести программные продукты, которые имитируют записную книгу и позволяют писать заметки и управлять ими.

К виду интеллект-карта можно отнести программные продукты, которые структурируют информацию в виде карты мыслей и методу интеллект-карт.

Интеллект-карта — это способ фиксации мыслей, наиболее похожий на то, как они рождаются и развиваются в нашей голове. Их также могут называть диаграммами связей, ментальными или ассоциативными картами и картами мыслей [2].

Программы для ведения заметок можно разделить на классы на основе предоставления сервисов в Интернет:

– offline-приложения – функционала – представляют собой приложения с частичным функционалом работающий через интернет;
– offline-приложения с элементами online-функционала – представляют собой приложения с частичным функционалом работающий через интернет;

– online-приложения – это приложения, функционал которого работает только с использованием интернета и представляют собой веб-приложения.

Веб-приложения – это интерактивные компьютерные приложения, разработанные для сети интернет, позволяющие пользователям вводить, получать и манипулировать данными с помощью взаимодействия. Такие программы обычно имеют очень тесную связь с сервером, отправляя на него множество запросов [2].

Также программы можно разделить на виды по их стоимости:

– бесплатные программы;

– бесплатные программы с дополнительными платными функциями;

– платные программы.

1.2 Анализ существующих аналогов

Сейчас создано много различных программных обеспечений для создания пользовательских заметок, включающие простые и сложные, а также бесплатных и платных. Но каждая программа содержит различные функции и распространяется по-разному.

1.2.1 Google Keep

Одним из известных сервисов для создания быстрых заметок, является сервис от компании Google. Google Keep — онлайн-сервис для создания заметок и их хранения.

Сервис обладает следующими возможностями:

– создавать заметки в виде карточек;

– отправить заметку в архив;

– выбирать цвет карточки;

– поиск по заметкам;

– добавлять теги.

На рисунке 1.1 представлен интерфейс приложения Google Keep.

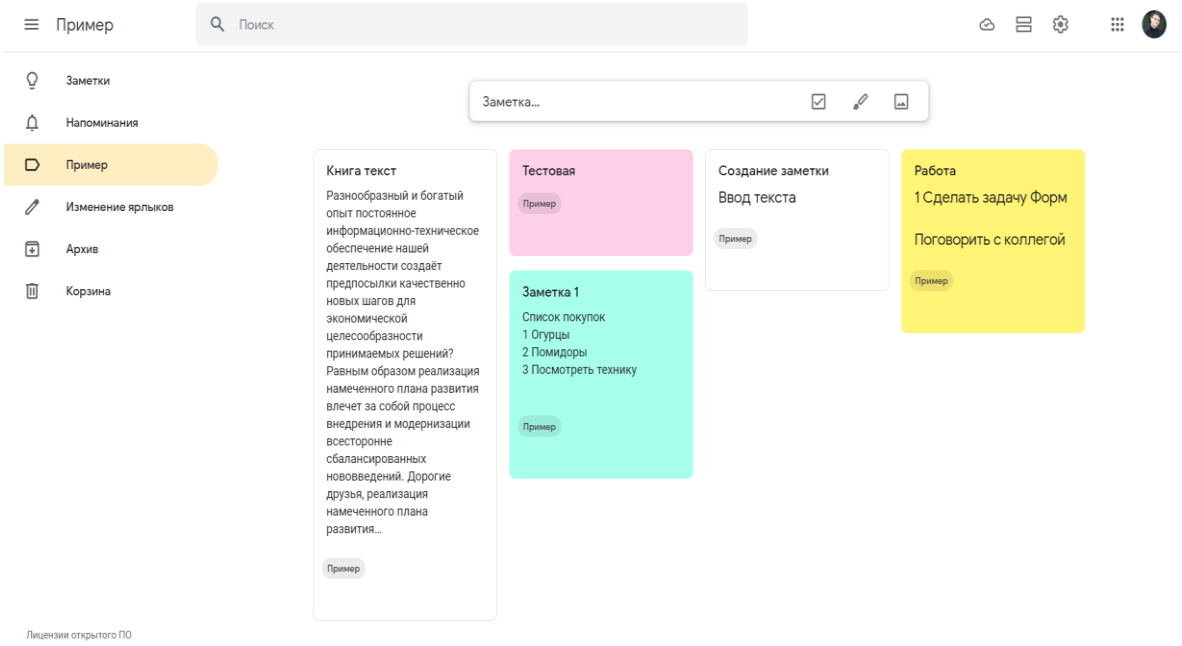

Рисунок 1.1 – Интерфейс программы Google Keep

Для создания заметки надо нажать поле заметка и тогда откроется небольшая форма для создания. Форма создания заметки представлена на рисунке 1.2.

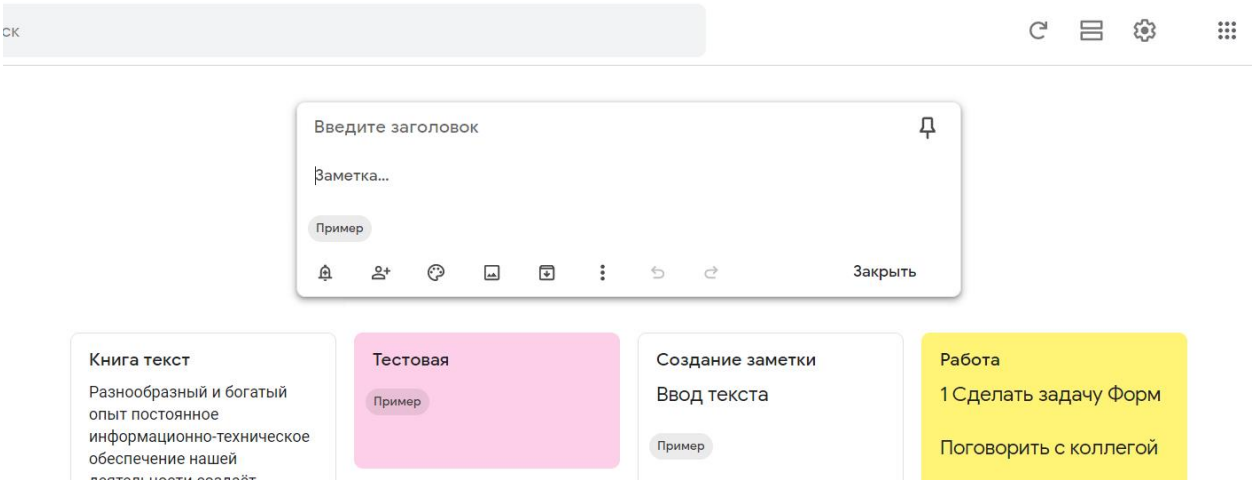

Рисунок 1.2 –Интерфейс формы по созданию заметки

При создании есть следующие возможности:

– изменить цвет карточки;
– добавить соавтора (только имеющего учетную запись Google);
– добавить тег;
– создание разного типа заметок.

Google keep поддерживает следующие виды заметок:

– текстовые
– в виде списка

– картинка (ссылка на картинку из интернета).

Заметки в виде списка позволяют отмечать выполнение данных задач. На рисунке 1.3 представлен пример создания заметки в виде списка с изменением цвета карточки.

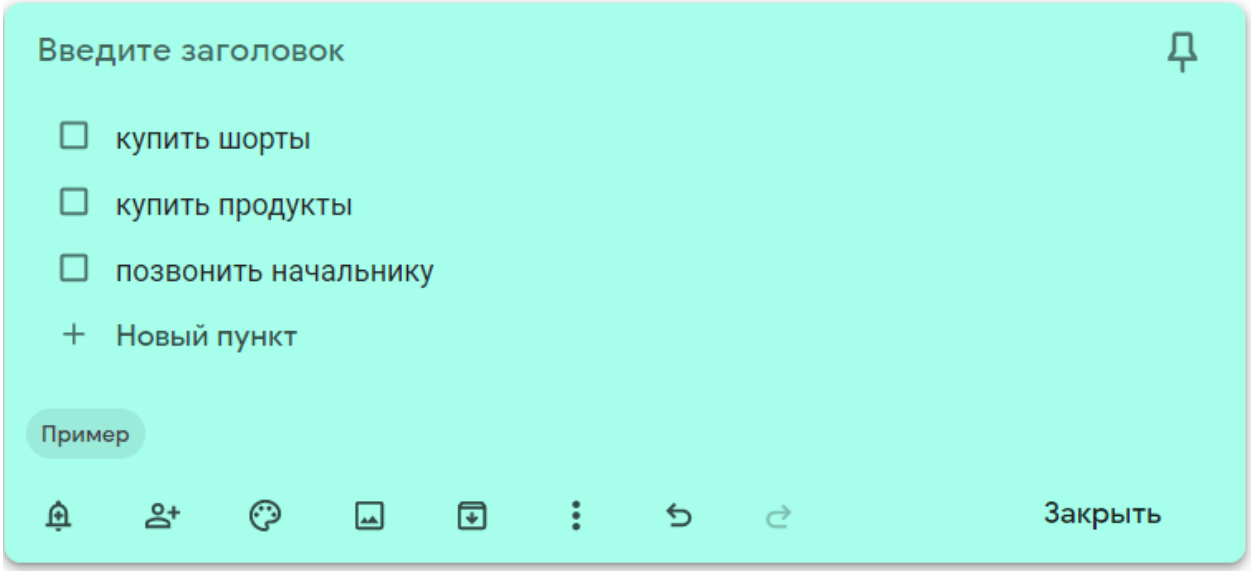

Рисунок 1.3 – Заметка в виде списка

Так как это сервис принадлежит к общей экосистеме google, он предоставляется на бесплатной основе только пользователям, у которых есть учетная запись Google. Это накладывает ограничения на пользователя, так как он должен создать себе учетную запись и контролировать ее.

Проанализировав сервис можно выделить следующие преимущества:

– простой дизайн;
– создавать быстрые заметки, которые нужны на короткий срок;
– создавать разные типы заметок;
– поиск по заметкам;
– использование тегов;
– поддерживает русский язык;
– является частью экосистемы google, на бесплатной основе.

Сервис обладает следующими недостатками:

– хаотичное расположение карточек и когда их большое количество сложность в поиске информации;
– привязан к экосистеме google;
– возможность использовать в одно время один тип заметок, что не позволяет группировать информацию в одной заметке.

1.2.2 Simplenote

Simplenote – это менеджер заметок. Simplenote позволяет работать только с текстами — поддержки иллюстраций, вложенных файлов и данных

прочих форматов здесь нет. Интерфейс программы представлен на рисунке 1.4.

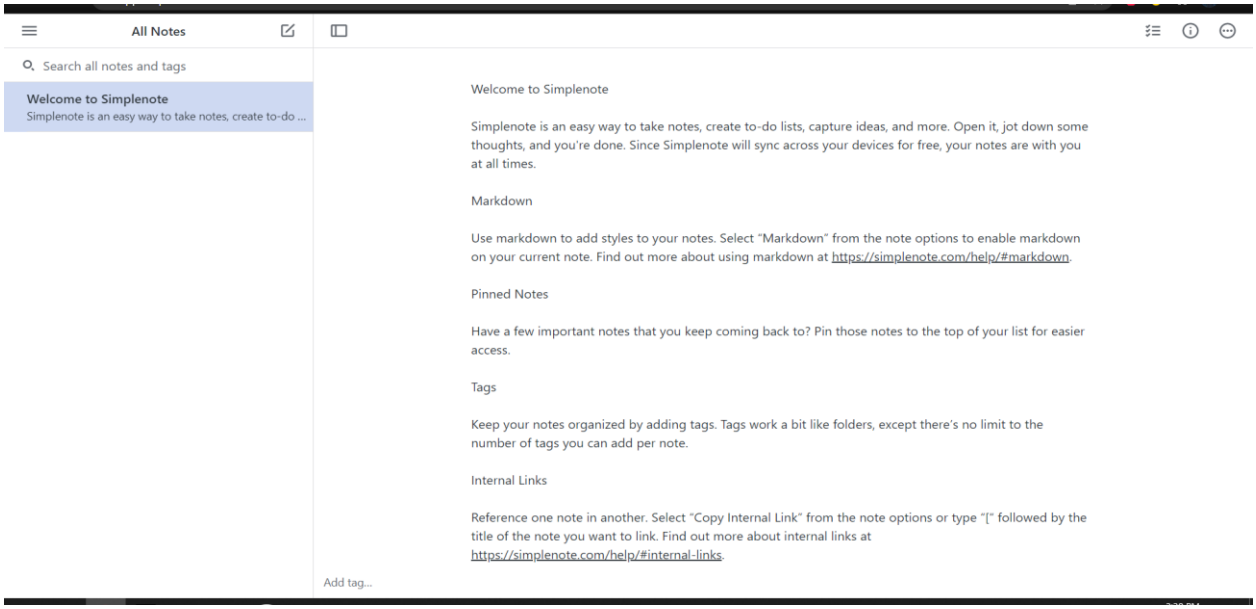

Рисунок 1.4 – Интерфейс программы

Программа предоставляет следующие возможности:

– создавать текстовые заметки, без возможности редактирования текста;
– искать заметки;
– добавлять теги.

Интерфейс позволяет добавлять заметки при нажатии на иконку редактирования и представлен на рисунке 1.5.

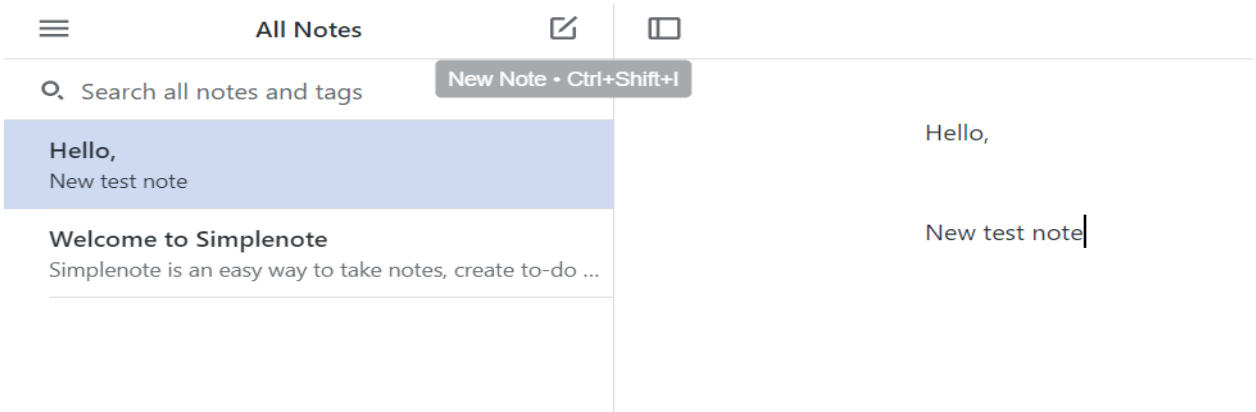

Рисунок 1.5 – Интерфейс создания заметки в программе SimpleNote

Для поиска используется поисковая строка, которая позволяет искать по названию, тегам и по содержанию заметки.

Это простое приложение имеет следующие преимущества:

– формировать текстовых заметок;

– добавление тегов;
– поиск по заметкам;
– бесплатное распространение.

Приложение обладает следующими недостатками:

– нет возможности оформлять другие типы заметок, только текстовые.
– интерфейс только на английском языке

По итогу можно сказать, что приложение позволяет, удобно не отвлекаясь записывать текстовые заметки, это удобно для студентов, которые ведут заметки лекций.

1.2.3 Notion

Notion — универсальная программа, которую в равной степени можно использовать и как приложение для хранения заметок, и в качестве платформы для создания баз знаний наподобие «Википедии», управления задачами и совместной работы с коллегами. Интерфейс программы Notion представлен на рисунке 1.6.

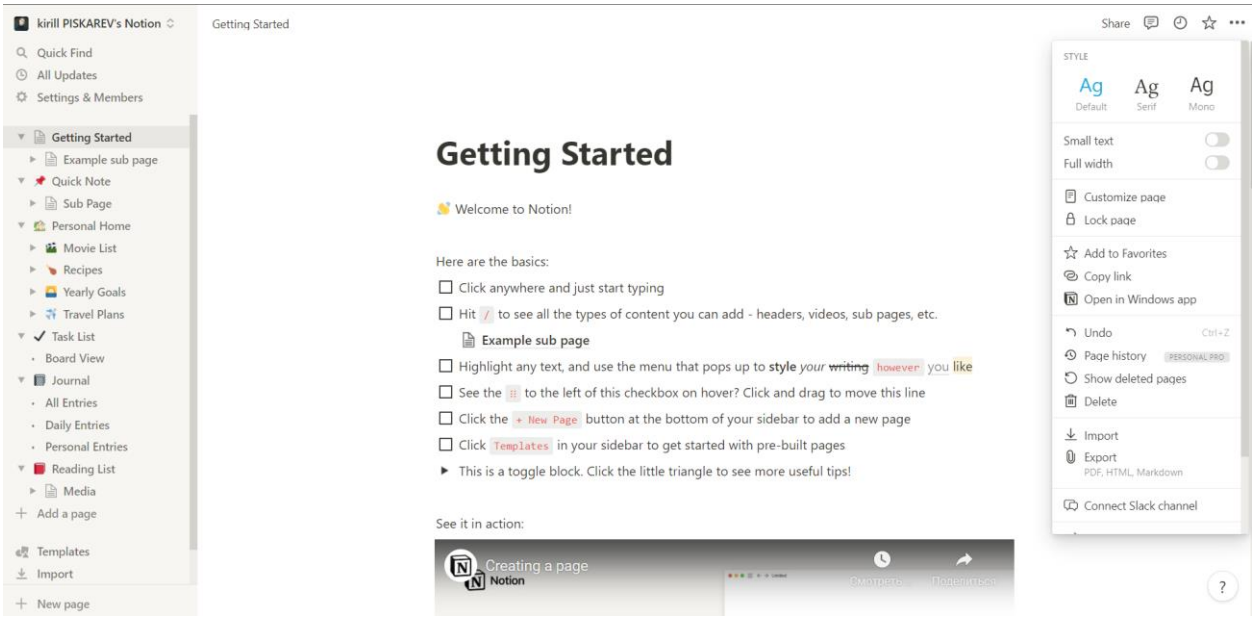

Рисунок 1.6 – Интерфейс программы Notion

В основной части функционал приложения предоставляет огромный выбор возможности подачи информации, начиная от форматируемых как угодно текстовых блоков, таблиц, чек-листов, канбан-досок или многоуровневых списков и заканчивая дизайнерскими шаблонами всевозможных документов.

Интерфейс создание новой заметки представлен на рисунке 1.7.

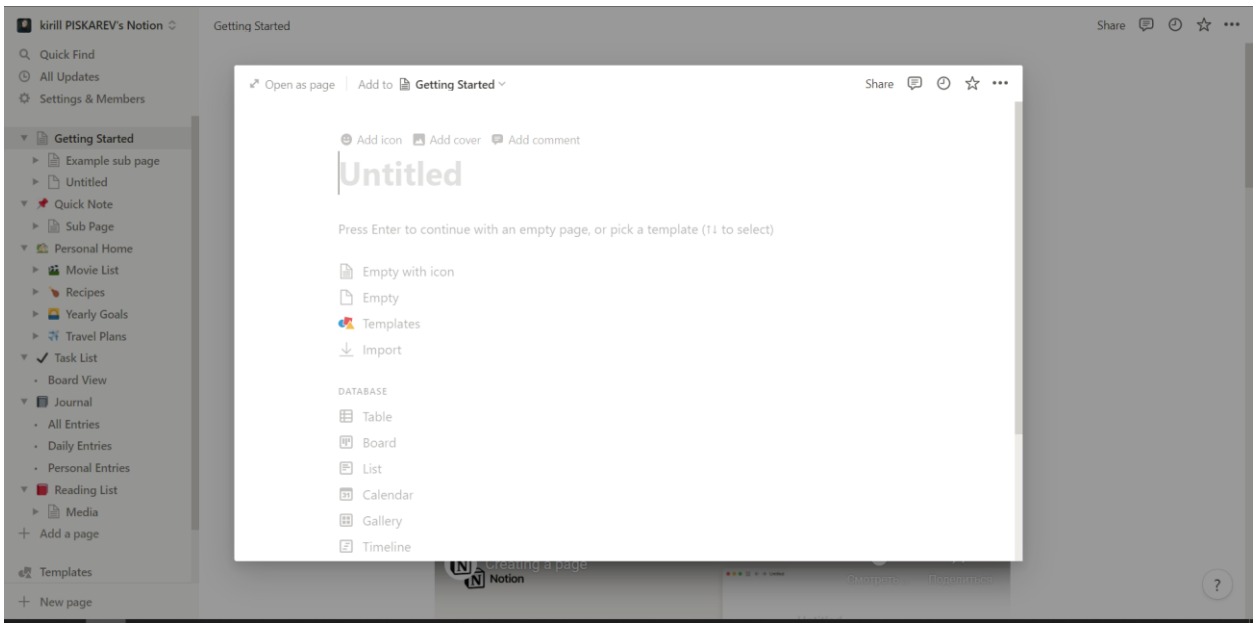

Рисунок 1.7 – Интерфейс создания заметки Notion

Урезанный функционал предоставляется бесплатно, и он больше предназначен для ознакомления, для полного использование, требуется оформлять подписку. Подписка составляет 4$ в месяц для личного использования и 8$ в месяц для командной работы.

Из-за большого набора функционала программа является сложной для освоения пользователя, без изучения предварительно документации по всем компонентам. Также большое количество функционала отвлекает от основной задачи записи заметок.

Программа имеет следующие преимущества:

– большое количество функция по созданию разных типов заметок;
– возможность добавлять теги;
– возможность работать с командой;
– создание разные виды досок.

Программа имеет следующее недостатки:

– не удобно использовать, из-за большого количества функций, требующее изучение документации перед использованием;
– интерфейс представлен только на английском, корейском.

1.3 Постановка цели и задач

Как показал анализ программных решений в области создания и хранения заметок они обладают следующими недостатками:

– нет возможности форматировать текст;
– не позволяют группировать разные типы заметок;

– не все позволяют вставлять ссылки на картинки из интернета с загрузкой картинки;
– у бесплатных решений не хватает функционала для полноценной работы с заметками;
– высокая стоимость;
– нет возможности поделиться заметкой с любым пользователем.

Целью данного дипломного проекта является разработка веб-приложения для управления и хранения пользовательских заметок, способных устранить вышеперечисленные недостатки и обладающими всеми необходимыми функциями, характерных для программных обеспечений в этой области.

Программа должна обеспечивать выполнение следующих функций:
– регистрация пользователя;
– авторизация пользователя;
– создавать, редактировать и удалять заметки;
– добавлять теги;
– редактирование текстовых заметок;
– добавление таблиц;
– оформлять в виде списка текст;
– добавлять ссылки на медиа.
– хранение заметок;
– поиск по заметкам;
– возможность управления пользователями администратору, чтобы сбросить пароль или удалить пользователя из системы со всеми данными;
– надежное шифрование паролей;
– возможность поделиться заметкой.

1.4 Входные данные

К входной информации мы будем относить все вводимые пользователем данные, конфигурационные данные по заметкам.

Пользователи при регистрации будут указывать свои персональные данные, для регистрации, а также при авторизации.

При создании заметки пользователи должны указать наименование заметки, сам текст заметки, теги относящиеся к заметкам. При поиске формировать поисковые запросы.

В качестве кода к входным данным мы отнесем запросы на сервер для работы с приложением, конфигурации и форматирование заметок.

Входной информацией будут является:
– данные для регистрации и авторизации;

– пользовательская информация для заметки;
– теги;
– добавление, редактирование и удаление пользовательских заметок;
– поисковые запросы;
– тип информации для формирования заметки (ссылки, текст, таблицы, код);
– методы запросов: GET, POST, PUT, DELETE.

1.5 Выходные данные

В качестве выходной информации будет выступать данные после сохранения заметок такие как: сама заметка в отформатированном виде, ссылки для возможности поделиться информацией.

Система будет отдавать в качестве выходной информации списки тегов, используемые в системе. Если пользователь добавляет тег, которого не было в системе, он автоматически добавляется для этого пользователя в список тегов.

К выходной информации также относятся системные данные: сессии и токены авторизации, ошибки системы.

К выходной информации мы относим:
– пользовательская информация;
– сессия и JWT;
– информация по заметкам;
– списки тегов;
– ссылки для просмотра пользовательских заметок;
– ошибки.

2 МОДЕЛИРОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ И РАЗРАБОТКА ФУНКЦИОНАЛЬНЫХ ТРЕБОВАНИЙ

2.1 Функциональные требования

Функциональным назначением разрабатываемого веб-приложения является предоставление возможностей управления и хранения пользовательских заметок.

В качестве пользователя программного обеспечения может выступать любой человек, который имеет доступ к персональному компьютеру с доступом к сети интернет. Для использования программного решения не требуется специальная подготовка или обучение пользователей.

Предполагается возможность одновременной эксплуатации разрабатываемого решения широким кругом пользователей.

Исходя из предполагаемого использования, можно заключить, что проектируемое программное решение должно реализовывать следующие группы функций:

- регистрация и авторизация пользователя;
- управление и хранение пользовательских заметок;
- поиск по заметкам;
- управление доступом к заметкам;
- управление пользователями.

2.2 Описание функциональности программного обеспечения

Диаграмма вариантов использования (use case diagram) – это наиболее общее представление функционального назначения системы.

Диаграмма вариантов использования призвана ответить на главный вопрос моделирования: «что делает система во внешнем мире?». На диаграмме применяются два типа основных сущностей: варианты использования и действующие лица, между которыми устанавливаются следующие основные типы отношений:

- ассоциация между действующим лицом и вариантом использования;
- обобщение между действующими лицами;
- обобщение между вариантами использования;
- зависимости (различных типов) между вариантами использования.

В системе можно выделить следующие действующие лица:

- анонимный пользователь;
- пользователь;
- администратор;

На рисунке 2.1 представлена общая диаграмма вариантов использования.

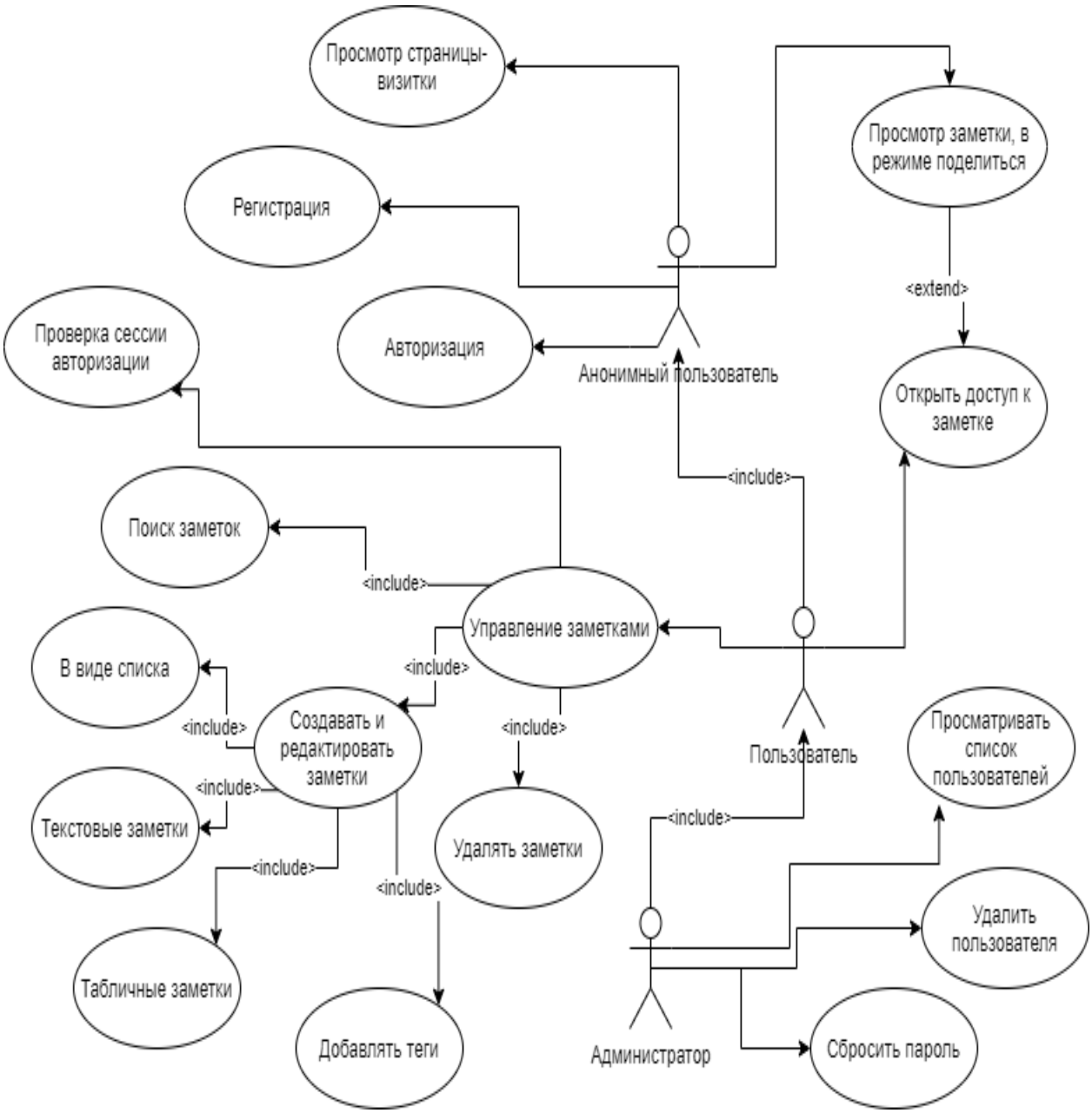

Рисунок 2.1 – Диаграмма вариантов использования

Анонимный пользователь – это пользователь, который не авторизован в системе. Данный пользователь может просмотреть главную страницу сайта, с описанием назначения веб-приложения. Анонимному пользователю доступны функции регистрации в приложении, а также авторизации.

Если Анонимному пользователю дали ссылку на доступ к режиму чтения заметки, ему доступна информация по заметке.

После авторизации анонимный пользователь получает роль «пользователь», после этого непосредственно перенаправляется в режим управления заметками, но у него остаются все возможности и анонимного

пользователя. В режиме управления заметками пользователю доступны следующие функции:

- создание заметки;
- редактирование заметки;
- удаление заметки;
- поиск заметок;
- открыть доступ к заметке;
- выход из системы;

Редактирование заметок включает в себя следующие возможности:

- работа с текстом (его форматирование и оформление);
- создание списка (числового, пунктами, переключателями);
- создание таблиц;
- добавление медийных ссылок (картинки и другие ссылки на источники из интернета).

В системе присутствует роль «Администратор», которая включает в себя возможности «Анонимного пользователя» и «Пользователя», а также расширяет возможности по управлению пользователями. У Администратора есть возможность просматривать пользователей, при необходимости сбрасывать пароль, а также удалять пользователя.

2.3 Разработка информационной модели

Информационный объект – это описание некоторой сущности предметной области реального объекта, процесса, явления или события. Информационный объект образуется совокупностью логически взаимосвязанных реквизитов, представляющих качественные и количественные характеристики сущности.

Проанализировав возможные пользовательские действия была спроектирована информационная модель БД, представленная на рисунке 2.2.

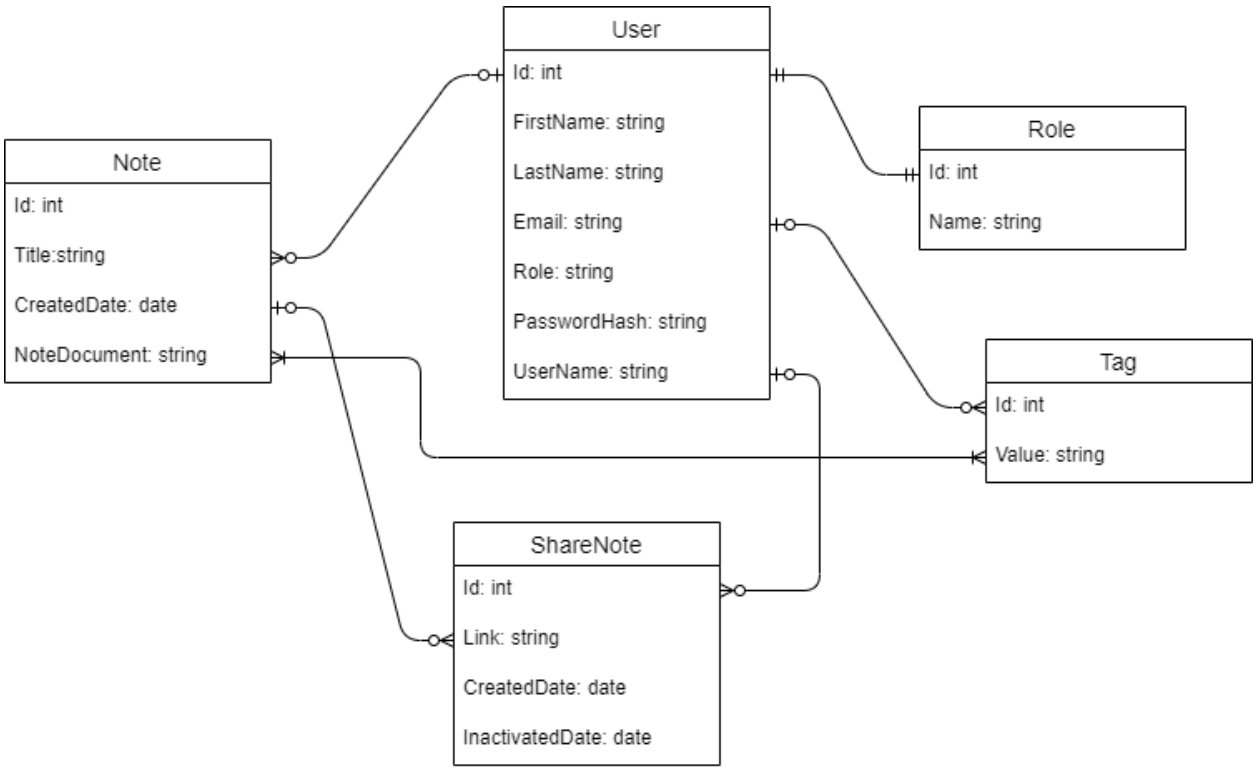

Рисунок 2.2 – Информационная модель БД

В процессе анализа предметной области были выделены следующие типы сущностей:

– User – сущность хранящая пользовательские учетные записи с требуемой пользовательской информацией. Атрибуты: Id – идентификатор пользователя; FirstName – имя пользователя; LastName – фамилия пользователя; Email – почтовый адрес пользователя; Role – роль пользователя в системе; PasswordHash – зашифрованный пароль пользователя; UserName – имя учетной записи в системе;

– Role – сущность для хранения ролей в системе. Атрибуты: Id – идентификатор роли; Name – наименование роли;

– Note – сущность для описание заметки. Атрибуты: Id – идентификатор заметки; Title – название заметки; CreatedDate – дата создания; NoteDocument – сами данные заметки, формат разметки html;

– Tag – сущность для описания тегов в системе. Атрибуты: Id – идентификатор тега; Value – значение тега;

– ShareNote – сущность для описания данных по режиму поделиться заметкой. Атрибуты: Id – идентификатор; Link – ссылка для возможности просмотреть заметку; CreateDate – дата создания ссылки; InactivedDate – дата инактивирования.

2.4 Разработка модели взаимодействия пользователя с интерфейсом

Проанализировав тенденции для реализации приложения будет использоваться подход одностраничных приложений.

SPA или Single Page Application — это одностраничное веб-приложение, которое загружается на одну HTML-страницу. Благодаря динамическому обновлению с помощью JavaScript, во время использования не нужно перезагружать или подгружать дополнительные страницы. На практике это означает, что пользователь видит в браузере весь основной контент, а при прокрутке или переходах на другие страницы, вместо полной перезагрузки нужные элементы просто подгружаются.

В процессе работы пользователю может показаться, что он запустил не веб-сайт, а десктопное приложение, так как оно мгновенно реагирует на все его действия, без задержек.

К преимущества одностраничным приложениям можно отнести:

– более быстрая загрузка страниц;

– улучшенное восприятие пользовательского интерфейса, поскольку загрузка данных с сервера происходит в фоновом режиме;

– нет необходимости писать код на сервере для визуализации страницы;

– разделение на Front-end и Back-end — разработку;

– упрощенная разработка под мобильные приложения; вы можете повторно использовать один и тот же серверный код для веб-приложения и мобильное приложение;

Данная технология также обладает следующеми недостатками

– тяжелые клиентские фреймворки, которые нужно загружать на каждый клиент;

– веб-формы не компилируются, а потому их сложнее отлаживать и они содержат больше уязвимостей, которыми могут воспользоваться злоумышленники;

– SEO (Search Engine Optimization) — SPA усложняет оптимизацию сайта под поисковые движки. Поскольку большая часть веб-страницы строится на стороне клиента, боты поисковых систем видят страницу совершенно иначе, чем пользователь;

На рисунке 2.3 представлен пример сравнения жизненного цикла многостраничных приложений и одностраничных.

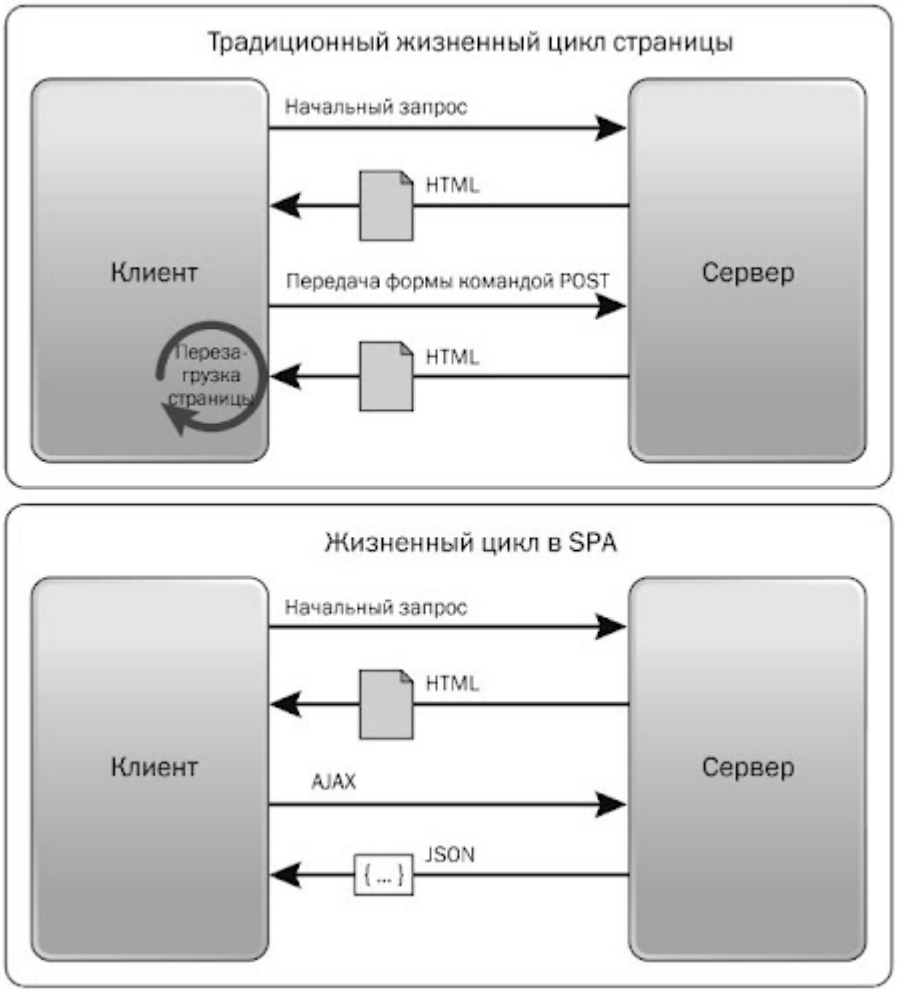

Рисунок 2.3 – Сравнение жизненного цикла многостраничных и одностраничных веб-приложений

2.5 Разработка технических требований к программному обеспечению

Разрабатываемое программное решение должно обеспечивать корректное функционирование при развертывании программных модулей на сервере со следующими техническими характеристиками:

– Intel 2.2 Ггц или более быстродействующий процессор;
– оперативная память 4 Гбайт или более;
– доступный объем дискового пространства 100 Гбайт.

Для нормального функционирования клиентской части программного комплекса должны выполняться следующие технические требования:

– Pentium 2 Ггц или более быстродействующий процессор;
– оперативная память 2 Гбайт или более;
– 32- или 64-битная версия операционных систем Windows 10, 11;
– браузер Google Chrome версии 95.x, Opera версии 81.x

3 ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

3.1 Обоснование выбора среды разработки

Процесс разработки веб-приложения делиться на следующие этапы: разработка серверной части, разработка клиентской части, верстка. Для этого были использованы следующие инструменты и технологии: C#, ASP.NET Core, Vue3, JS, HTML, CSS, MS SQL Server 2019, Visual Studio 2019 Community.

C# — современный объектно-ориентированный и типобезопасный язык программирования. C# позволяет разработчикам создавать разные типы безопасных и надежных приложений, выполняющихся в .NET

ASP.NET Core является кроссплатформенной, высокопроизводительной средой с открытым исходным кодом для создания современных облачных приложений, подключенных к Интернету. ASP.NET Core позволяет выполнять следующие задачи:

– создавать веб-приложения и службы, приложения IoT и серверные части для мобильных приложений;

– использовать избранные средства разработки в Windows, macOS и Linux;

– выполнять развертывания в облаке или локальной среде;

– работать в .NET Core или .NET Framework.

– ASP.NET Core предоставляет следующие преимущества:

– единое решение для создания пользовательского веб-интерфейса и веб-API;

– интеграция современных клиентских платформ и рабочих процессов разработки;

– облачная система конфигурации на основе среды;

– встроенное введение зависимостей;

– упрощенный высокопроизводительный модульный конвейер HTTP- запросов;

– возможность размещения в IIS, Nginx, Apache, Docker или в собственном процессе;

– параллельное управление версиями приложения, ориентированное на .NET Core;

– инструментарий, упрощающий процесс современной веб-разработки;

– возможность сборки и запуска в ОС Windows, macOS и Linux;

– открытый исходный код и ориентация на сообщество.

Vue 3 представляет современный прогрессивный фреймворк, написанный на языке TypeScript и предназначенный для создания веб-

приложений на уровне клиента. Основная область применения данного фреймворка – это создание и организация пользовательского интерфейса.

Vue 3 имеет довольно небольшой размер и при этом обладает хорошей производительностью по сравнению с такими фреймворками как Angular или React, а также по сравнению с предыдущей версией - Vue.js 2.x. Поэтому неудивительно, что данный фреймворк в последнее время набирает обороты и становится все более популярным.

Одним из ключевых моментов в работе Vue 3 является виртуальный DOM. Структура веб-страницы, как правило, описывается с помощью DOM (Document Object Model), которая представляет организацию элементов html на странице. Для взаимодействия с DOM (добавления, изменения, удаления html-элементов) применяется JavaScript. Но когда мы пытаемся манипулировать html-элементами с помощью JavaScript, то мы можем столкнуться со снижением производительности, особенно при изменении большого количества элементов. А операции над элементами могут занять некоторое время, что неизбежно скажется на пользовательском опыте. Однако если бы мы работали из кода js с объектами JavaScript, то операции производились бы быстрее.

Bootstrap – это инструментарий с открытым исходным кодом для разработки с помощью HTML (HyperText Markup Language), CSS (Cascading Style Sheets), JS (JavaScript). Он включает в себя множество разных компонентов для веб-сайта: типографику, веб-формы, кнопки, блоки навигации и другие. Основные преимущества Bootstrap:

- адаптивность – дизайн сайта будет корректно отображаться на экранах устройств разных размеров вне зависимости от диагонали;
- кросс-браузерность – одинаковое отображение во всех браузерах;
- легкость в использовании;
- понятный код – позволяет писать качественный и понятный код, что облегчает чтение кода для других разработчиков;

HTML – язык разметки (маркировки) гипертекста. HTML дает возможность производить переход от одной части текста к другой, и, что замечательно, эти части могут храниться на совершенно разных компьютерах. Он создан специально для разметки веб-страниц. Именно язык разметки дает браузеру необходимые инструкции о том, как отображать тексты и другие элементы страницы на мониторе.

CSS — язык таблиц стилей, который позволяет прикреплять стиль (например, шрифты и цвет) к структурированным документам (например, документам HTML и приложениям XML). Обычно CSS-стили используются для создания и изменения стиля элементов веб-страниц и пользовательских интерфейсов, написанных на языках HTML и XHTML, но также могут быть применены к любому виду XML-документа, в том числе XML, SVG и XUL. Отделяя стиль представления документов от содержимого документов, CSS упрощает создание веб-страниц и обслуживание сайтов.

JavaScript – это интерпретируемый язык программирования, разработанный для взаимодействия с веб-страницами. Он используется для описания сценариев для активных веб-страниц. Программа на JavaScript встраивается непосредственно в исходный текст HTML-документа и интерпретируется браузером по мере загрузки этого документа. С помощью JavaScript можно динамически изменять текст загружаемого HTML документа и реагировать на события, связанные с действиями посетителя или изменениями состояния документа или окна.

3.2 Разработка структурной схемы программного обеспечения

Для разработки веб-приложения используется одностраничная архитектура с использованием фреймворка Vue3 для разработки клиентской части и ASP.NET Core для разработки серверной части.

Разработка серверной части предполагает создание REST API.

REST API — это прикладной программный интерфейс (API), который использует HTTP-запросы для получения, извлечения, размещения и удаления данных. Аббревиатура REST в контексте API расшифровывается как «передача состояния представления» (Representational State Transfer).

REST API включает следующие принципы:

- единый интерфейс;
- разграничение клиента и сервера;
- нет сохранения состояния;
- кэширование всегда разрешено;
- многоуровневая система;
- код предоставляется по запросу.

Для передачи запросов используется протокол HTTPS.

HTTPS (от англ. HyperText Transfer Protocol Secure) — расширение протокола HTTP для поддержки шифрования в целях повышения безопасности.

На рисунке 3.1 представлена модель передачи данных использую REST API. На примере авторизации в системе.

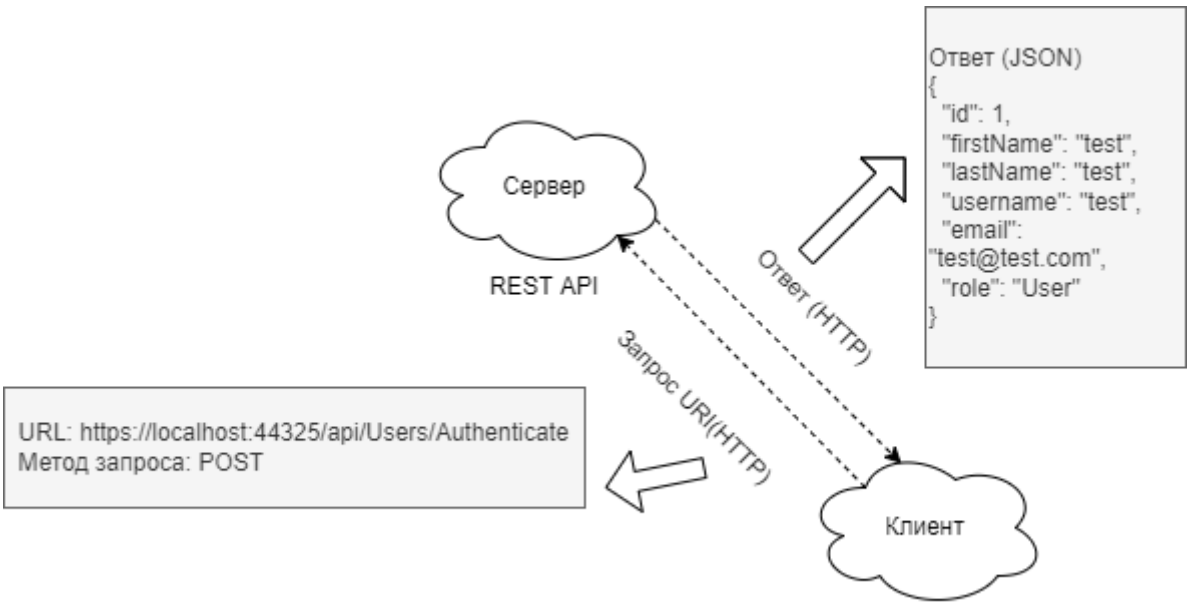

Рисунок 3.1 – Модель передачи данных с использованием REST API

Для разработки серверной части используется трехуровневая приложения. Данная модель архитектуры подразумевает разделение приложения на три уровня. На рисунке 3.2 представлена модель 3 уровневой архитектуры.

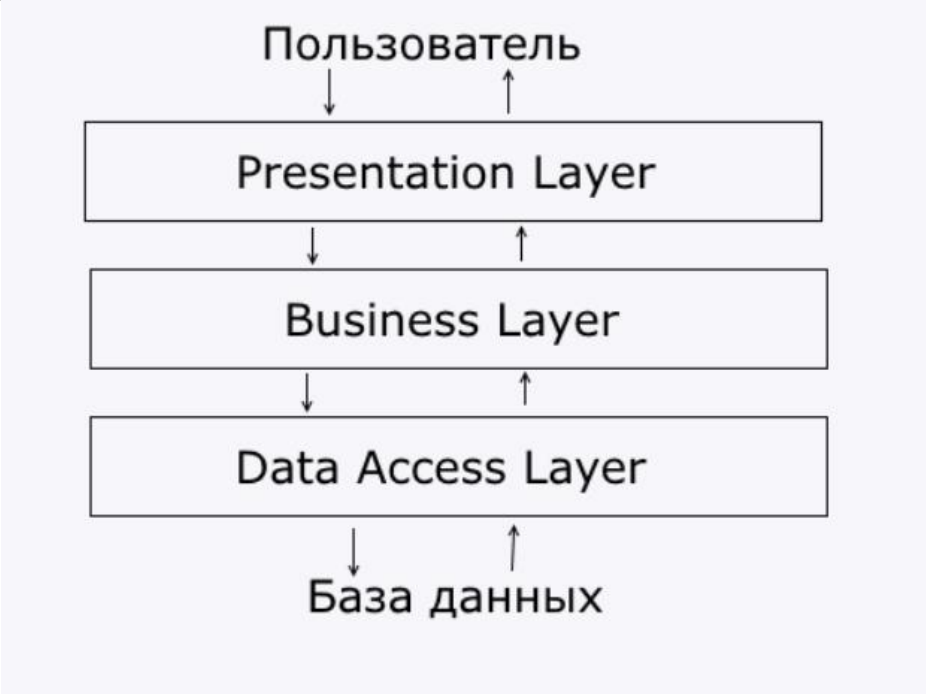

Рисунок 3.2 – Модель трехуровневой архитектуры

Presentation layer (уровень представления) – это тот уровень, с которым непосредственно взаимодействует пользователь. Этот уровень включает компоненты пользовательского интерфейса, механизм получения ввода от пользователя.

Business layer (уровень бизнес-логики) – содержит набор компонентов, которые отвечают за обработку полученных от уровня представлений данных, реализует всю необходимую логику приложения, все вычисления, взаимодействует с базой данных и передает уровню представления результат обработки.

Data Access layer (уровень доступа к данным) – хранит модели, описывающие используемые сущности, также здесь размещаются специфичные классы для работы с разными технологиями доступа к данным, например, класс контекста данных Entity Framework.

На рисунке 3.3 представлена трехуровневая архитектура приложения.

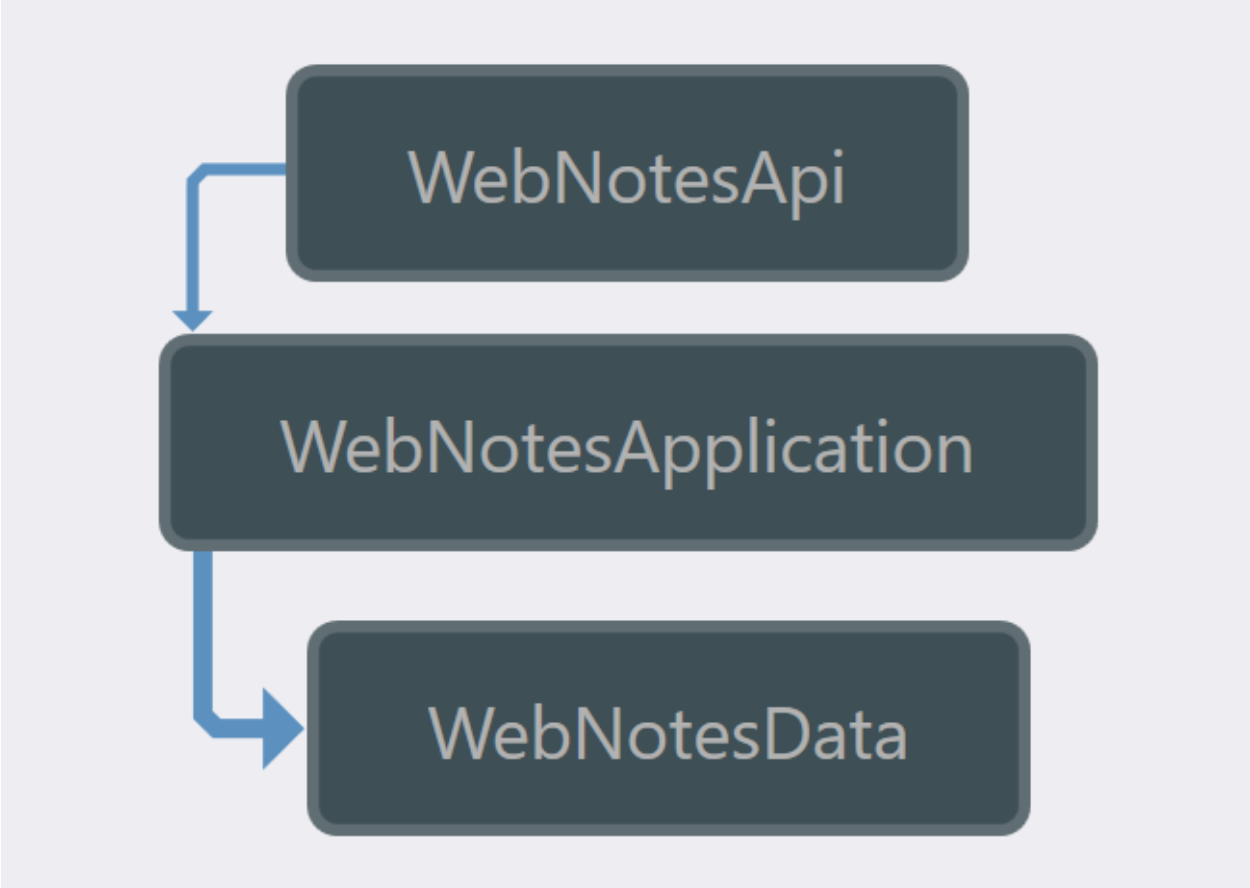

Рисунок 3.3 – Трехуровневая архитектура приложения

Для разработки клиентской части с использованием Vue3 подключены следующие модули:

– vue-router – официальная библиотека маршрутизации для. Она глубоко интегрируется с Vue и позволяет легко создавать SPA-приложения;

– vuex – паттерн управления состоянием и библиотека для приложений на Vue. Он служит централизованным хранилищем данных для всех компонентов приложения с правилами, гарантирующими, что состояние может быть изменено только предсказуемым образом.

На рисунке 3.4 представлена структурная схема работы Vuex.

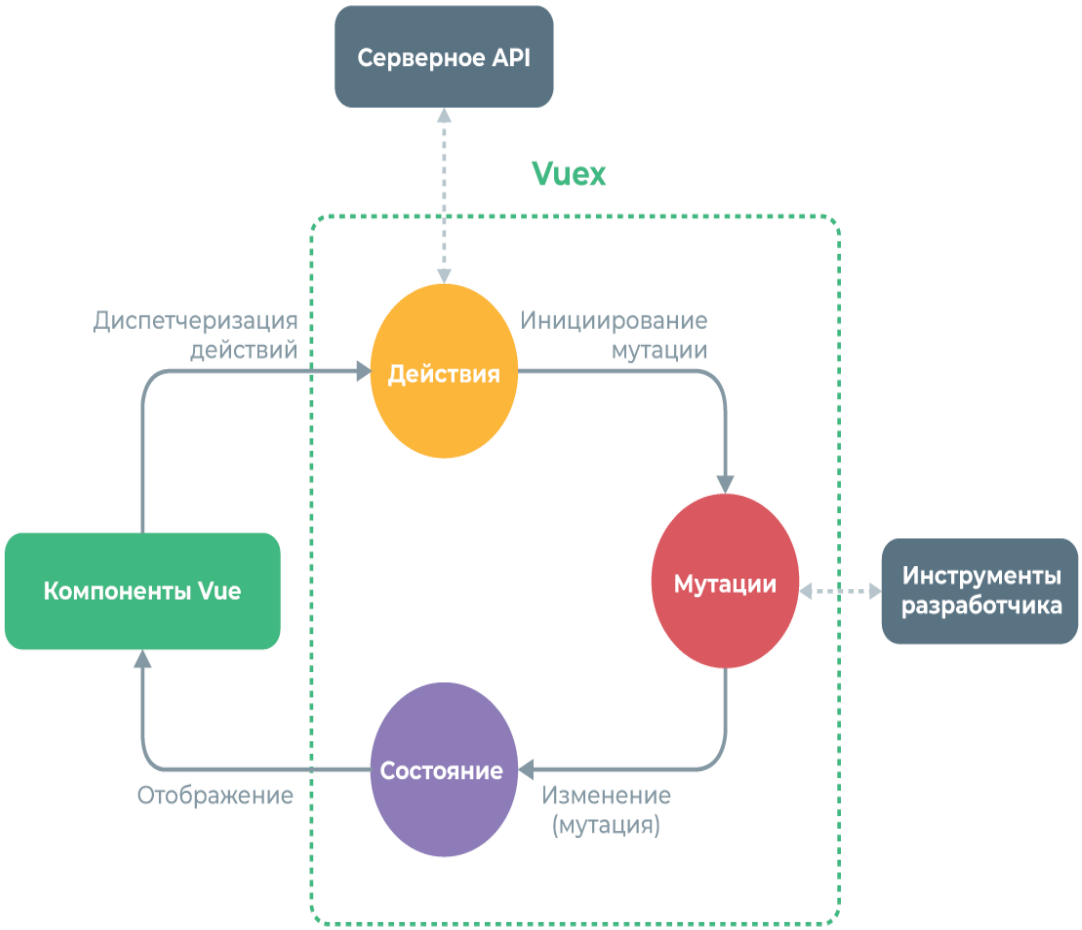

Рисунок 3.4 – Структурная схема работы Vuex

Структура клиентского приложения разделяется на следующие модули:

– хранилище состояний – в данном модуле будут хранится данные для использования глобально всеми компонентами;

– маршрутизация – данный модуль позволяет переходить и регистрировать различные страницы в системе и позволять управлять перенаправлением между страницами;

– представления – данный модуль, описывает сами страницы, которые будут представлены пользователю, данные страницы могут использовать компоненты, которые описывает небольшие функциональные блоки;

– провайдеры – сервисные модули, которые используются для переда данных между клиентской и серверной частью. Для этого используется библиотека Axios.

На рисунке 3.5 представлена клиентская структура взаимодействия компонентов.

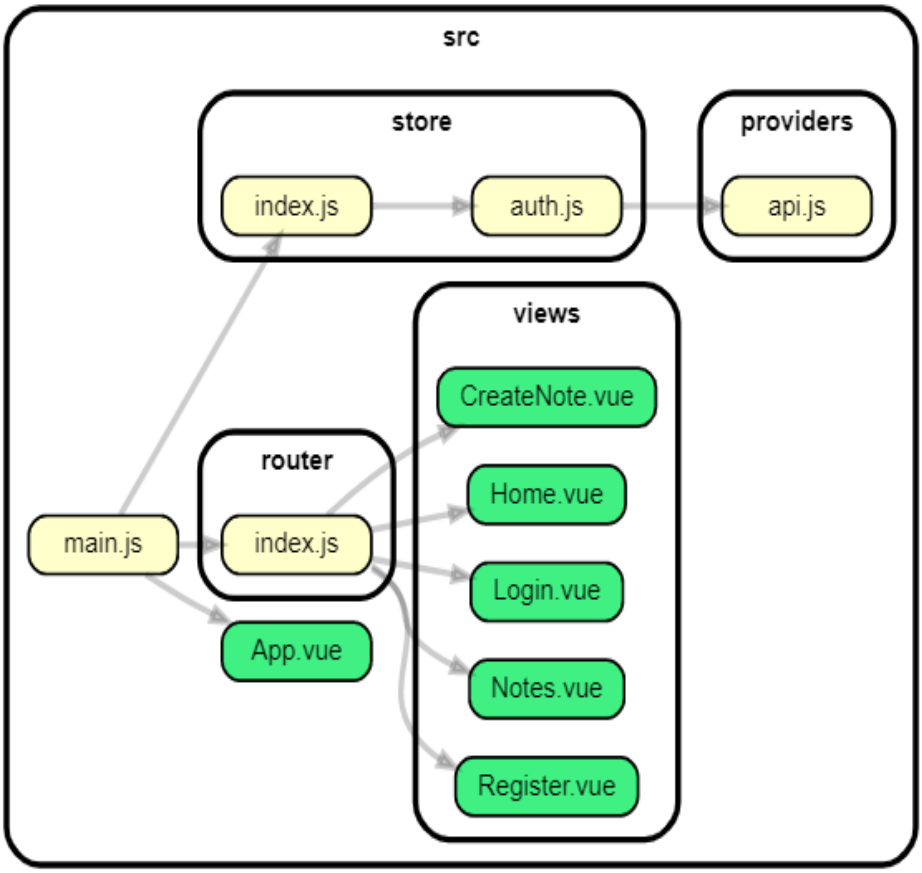

Рисунок 3.5 – Структура взаимодействия модулей клиентской части приложения

3.3 Разработка структуры классов

Диаграмма классов (class diagram) – основной способ описания структуры системы. На диаграмме классов применяется один основной тип сущностей: классы (включая многочисленные частные случаи классов: интерфейсы, примитивные типы, классы-ассоциации и многие другие), между которыми устанавливаются следующие основные типы отношений:

- ассоциация между классами;
- обобщение между классами;
- зависимости (различных типов) между классами и между классами и интерфейсами.

На рисунке 3.6 представлена диаграмма классов уровня представления.

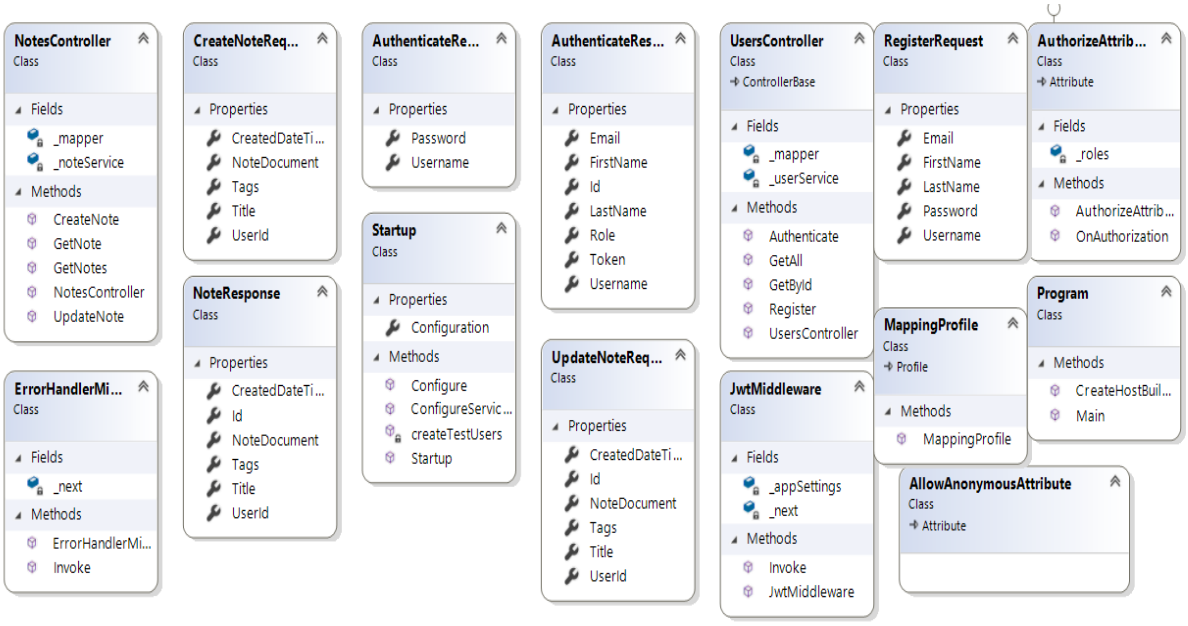

Рисунок 3.6 – Диаграмма классов уровня представления

Диаграмма классов уровня бизнес-логики представлена на рисунке 3.7.

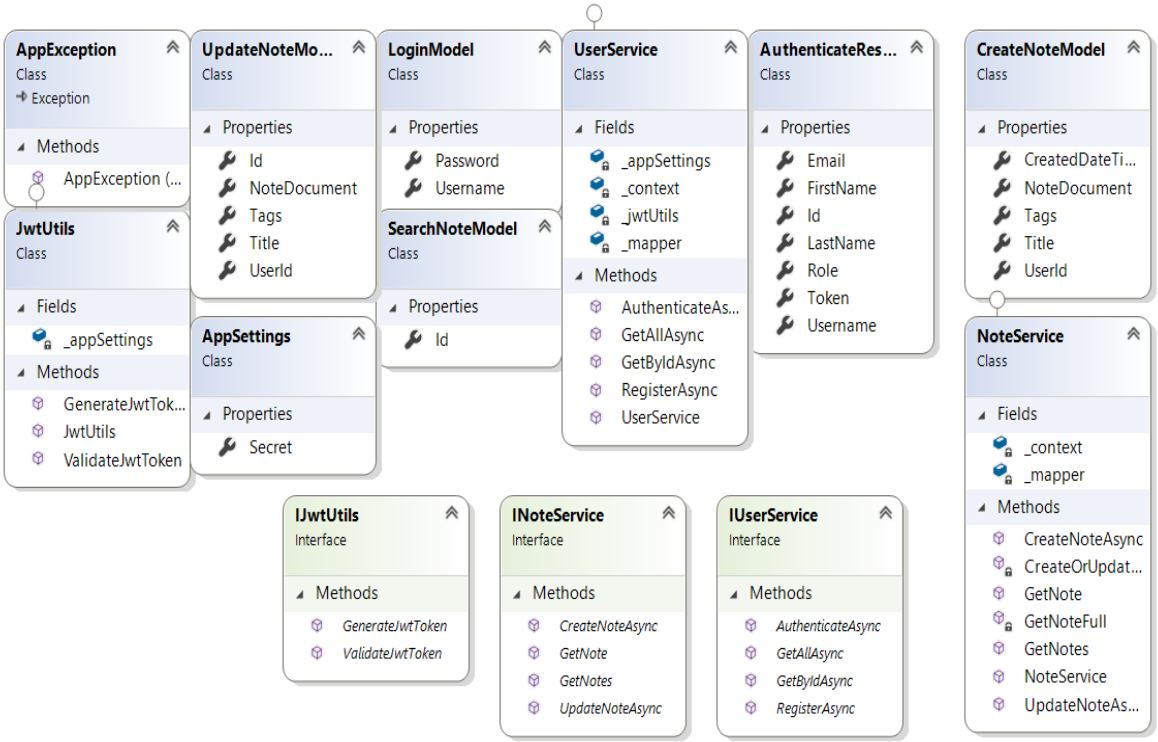

Рисунок 3.7 – Диаграмма классов уровня бизнес-логики

Диаграмма классов уровня доступа к данным представлена на рисунке 3.8.

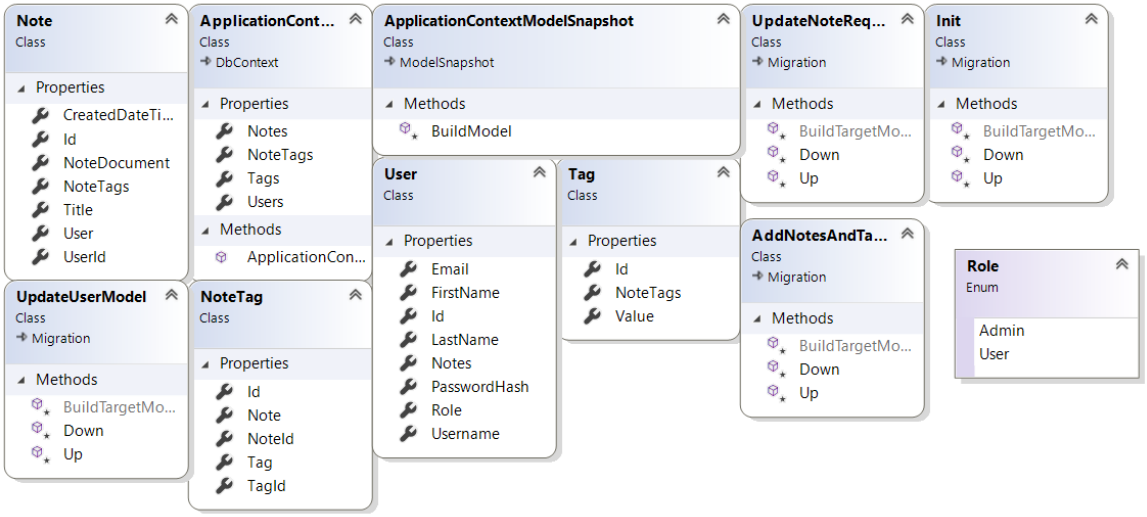

Рисунок 3.8 – Диаграмма классов уровня доступа к данным

3.4 Разработка физической модели данных

После проектирования информационной модели данных, она была реализована с использование реляционных баз данных MS SQL. И в ходе проектирования была проведена нормализация и не все сущности необходимы на уровне БД.

Достоинства реляционного подхода:

– наличие небольшого набора абстракций, которые позволяют сравнительно просто моделировать большую часть распространенных предметных областей и допускают точные формальные определения, оставаясь интуитивно понятными;

– наличие простого и в то же время мощного математического аппарата, опирающегося главным образом на теорию множеств и математическую логику и обеспечивающего теоретический базис реляционного подхода к организации БД;

– возможность ненавигационного манипулирования данными без необходимости знания конкретной физической организации баз данных во внешней памяти.

На рисунке 3.9 представлена физическая диаграмма базы данных.

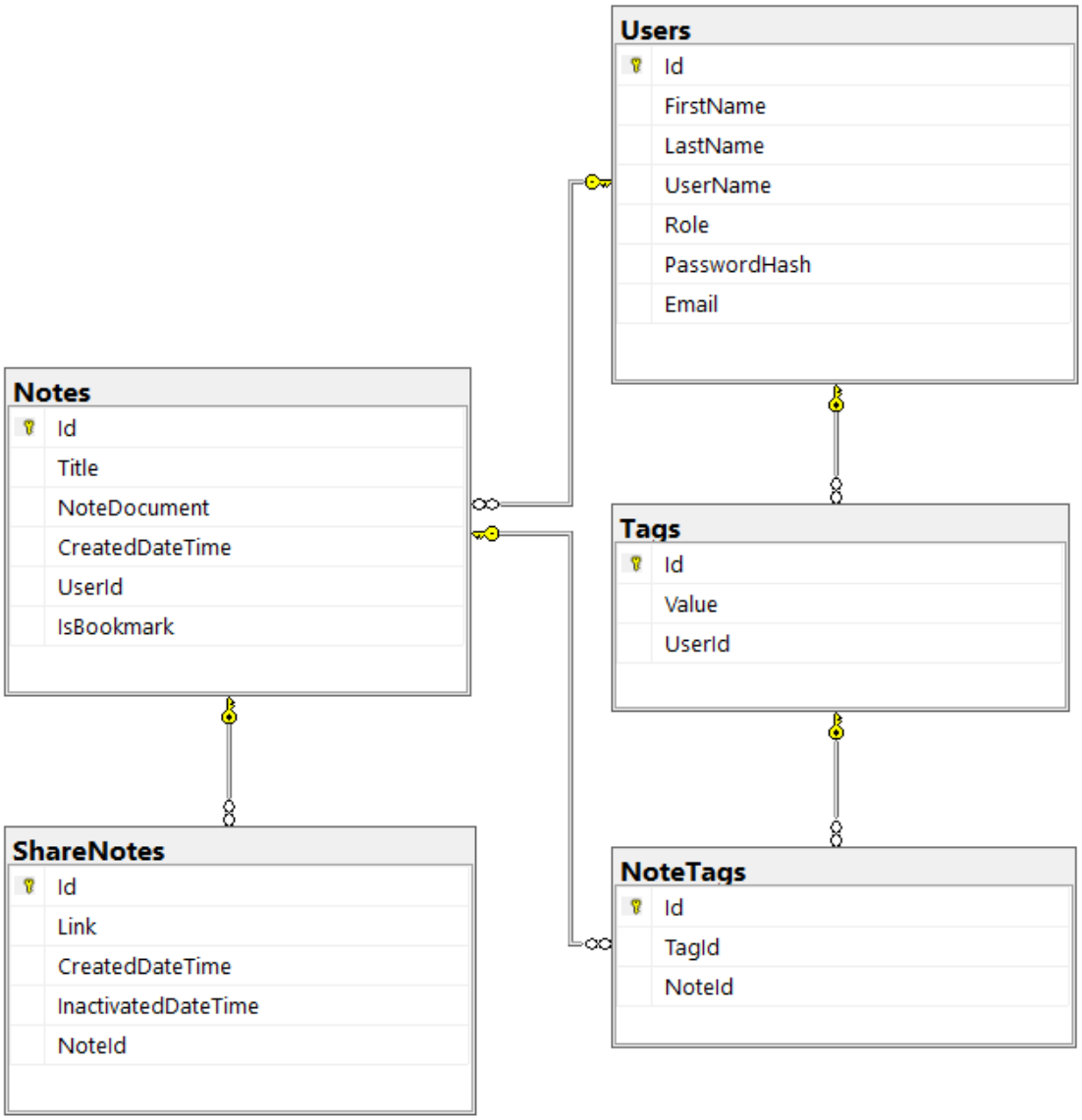

Рисунок 3.9 – Физическая модель данных БД

Формализованное описание объектов представлено в таблицах 3.1-3.5. Описание таблицы «Users» представлено в таблице 3.1.

Таблица 3.1 – Описание таблицы «Users»

Поле	Тип данных	Описание
Id	INT	Идентификатор пользователя
FirstName	NVARCHAR(MAX)	Имя пользователя
LastName	NVARCHAR(MAX)	Фамилия пользователя
UserName	NVARCHAR(MAX)	Имя учетной записи в системе
Role	INT	Роль пользователя

PasswordHash	NVARCHAR(MAX)	Хеш пароля
Email	NVARCHAR(MAX)	Почтовый адрес пользователя

Описание таблицы «Notes» представлено в таблице 3.1.

Таблица 3.2 – Описание таблицы «Notes»

Поле	Тип данных	Описание
Id	INT	Идентификатор заметки
Title	NVARCHAR(MAX)	Название заметки
NoteDocument	NVARCHAR(MAX)	Отформатированный текст заметки(в виде HTML
CreatedDateTime	DATETIME2(7)	Дата создания заметки
UserId	INT	Внешний ключ: Идентификатор пользователя
IsBookmark	BIT	Флаг для важных заметок

Описание таблицы «Tags» представлено в таблице 3.3.

Таблица 3.3 – Описание таблицы «Tags»

Поле	Тип данных	Описание
Id	INT	Идентификатор категории
Value	NVARCHAR(MAX)	Значение категории
UserId	INT	Внешний ключ: Идентификатор пользователя

Описание таблицы «NoteTags» представлено в таблице 3.4. Данная таблицы выступает связующей между таблицами «Notes» и «Tags», для образования связи многие ко многим

Таблица 3.4 – Описание таблицы «NoteTags»

Поле	Тип данных	Описание
Id	INT	Идентификатор
TagId	INT	Внешний ключ: Идентификатор категории

NoteId	INT	Внешний ключ: Идентификатор заметки

Описание таблицы «ShareNotes» представлено в таблице 3.4.

Таблица 3.5 – Описание таблицы «ShareNotes»

Поле	Тип данных	Описание
Id	INT	Идентификатор записи поделиться заметкой
Link	NVARCHAR(MAX)	Ссылка на просмотр заметки
CreatedDateTime	DATETIME2(7)	Дата создания
InactivedDateTime	DATETIME2(7)	Дата отмены работы ссылки
NoteId	INT	Внешний ключ: Идентификатор заметки

3.5 Проектирование алгоритмов работы программного обеспечения

Для определения логики написания программы были спроектированы следующие алгоритмы работы программного обеспечения.

3.5.1 Алгоритм регистрации на стороне сервера

Схема работы алгоритма регистрации на стороне сервера представлена на рисунке 3.10.

Данный алгоритм отображает схему работу регистрации на сервере приложения, после того как веб-приложение отправляет данные пользователя для регистрации. Результатом выполнения обработки запроса будет или ошибка, или успешный ответ с данными пользователями и JWT токеном.

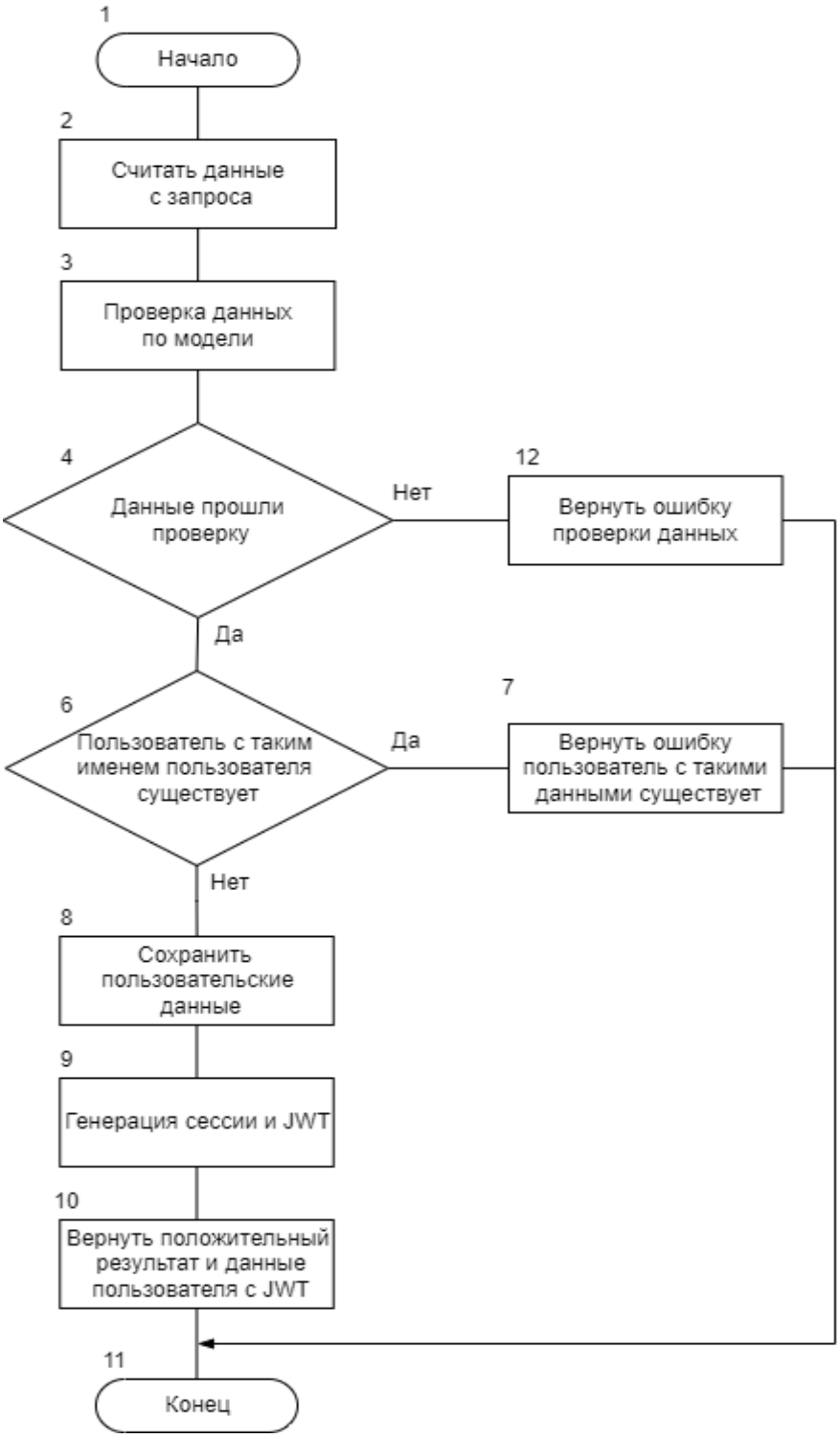

Рисунок 3.10 – Блок-схема работы алгоритма регистрации на стороне сервера

3.5.2 Алгоритм проверки уровня доступа на стороне сервера

Схема работы алгоритма регистрации на стороне сервера представлена на рисунке 3.11. Данный алгоритм демонстрирует как происходит проверка уровня доступа пользователя при отправке запроса на сторону сервера.

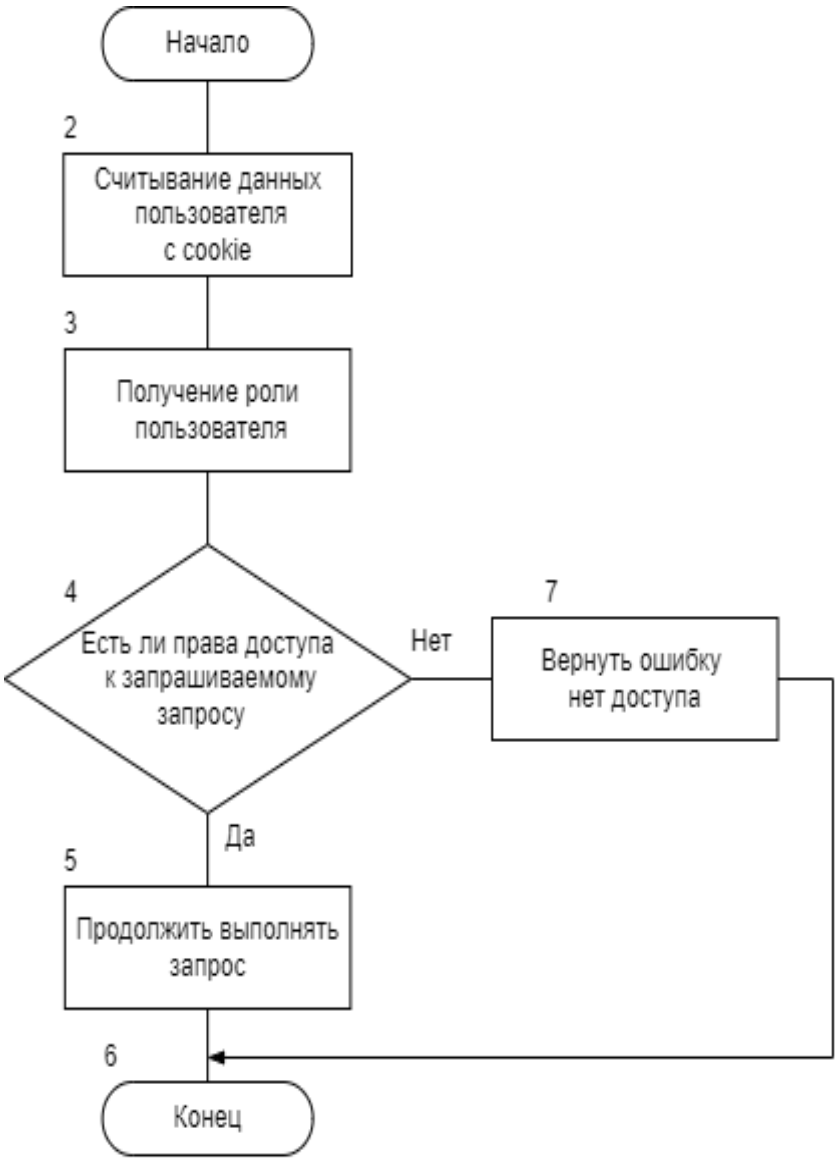

Рисунок 3.11 – Блок-схема алгоритма проверки уровня доступа на стороне сервера

3.5.3 Алгоритм от лица пользователя на стороне клиента

На рисунке 3.12 представлена блок схема алгоритма от лица пользователя на клиентской стороне программы. Данный алгоритм демонстрирует поведение системы на основе поведения пользователя, позволяя выдавать возможности работы с заметками после авторизации в системе.

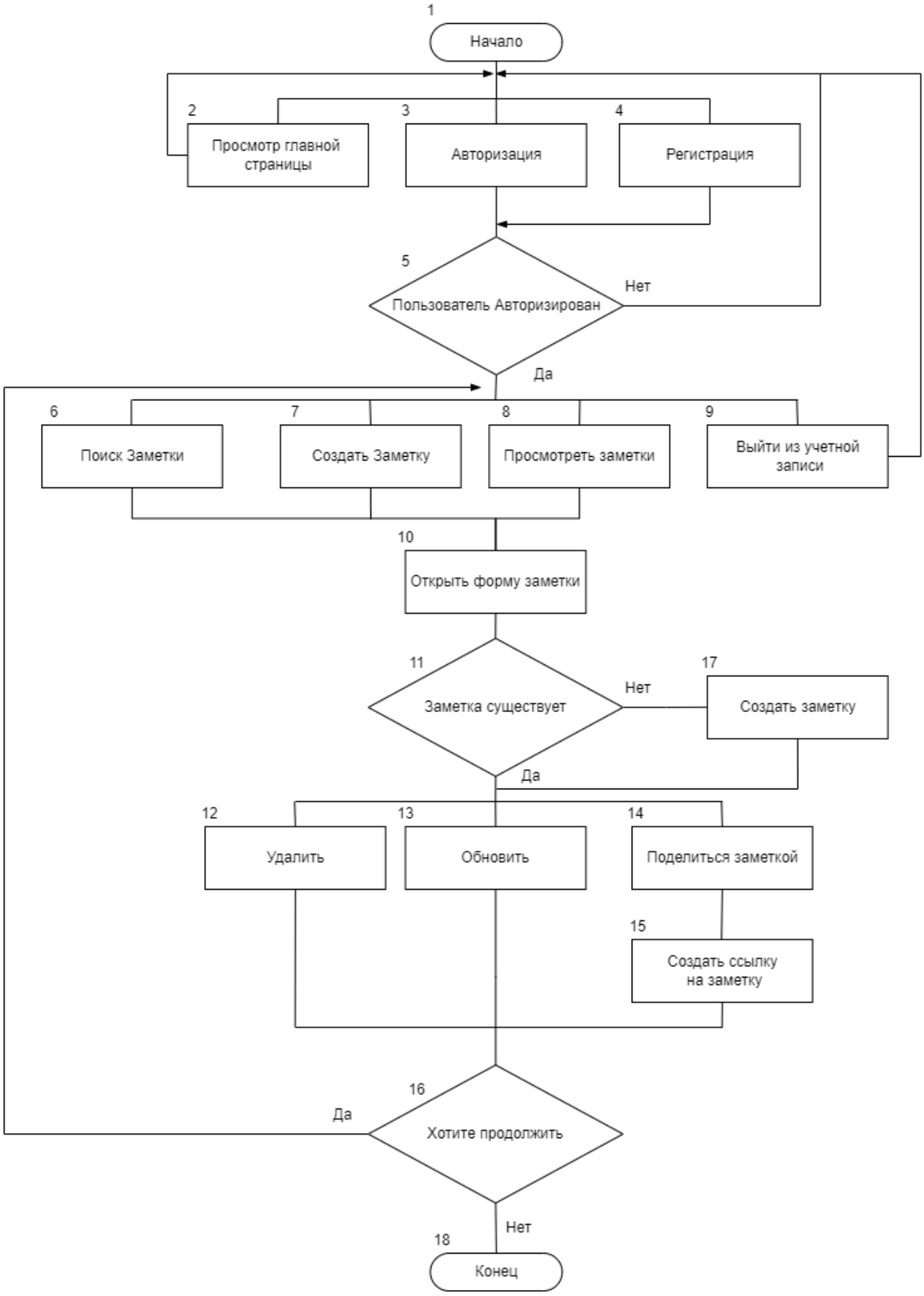

Рисунок 3.12 – Блок-схема алгоритма программы от лица пользователя на клиентской стороне

ЗАКЛЮЧЕНИЕ

В ходе прохождения преддипломной практики были выполнены все поставленные цели. Была разобрана предметная область и проанализированы готовые программные решения, определены достоинства и недостатки каждого приложения. На основе анализа, была поставлены цели и задачи.

После анализа и определения целей и задач, а также постановки входных и выходных данных, была проведено моделирование предметной области. В ходе моделирования предметной были определены функциональные требования. Также была смоделирована информационная модель и модели взаимодействия пользователя.

Завершив моделирования были спроектированы структурные схемы программы и алгоритмы работы программы.