Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К КУРСОВОМУ ПРОЕКТУ

НА ТЕМУ:

«База данных для просмотра и размещения развлекательного контента»

Студент _____ИУ7-62Б_____ _____ _____А.Д. Сучков_____
(Группа) (Подпись, дата) (И.О.Фамилия)

Руководитель курсового проекта _____ ____А.Л. Исаев_____
(Подпись, дата) (И.О.Фамилия)

Москва, 2021 г.

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

УТВЕРЖДАЮ
Заведующий кафедрой ___ИУ7____
(Индекс)
_____ _И.В. Рудаков_
(И.О.Фамилия)
« _____ » _____ 20 ____ г.

ЗАДАНИЕ
на выполнение курсового проекта

по дисциплине _____Базы данных_____

Студент группы _____ИУ7-62Б_____

_____Сучков Александр Дмитриевич_____
(Фамилия, имя, отчество)

Тема курсового проекта _____
_____База данных для просмотра и размещения развлекательного контента____

Направленность КП (учебный, исследовательский, практический, производственный, др.)
_____учебный_____
Источник тематики (кафедра, предприятие, НИР) __кафедра_____

График выполнения проекта: 25% к _4_ нед., 50% к _7_ нед., 75% к 11 нед., 100% к _14_ нед.

Задание

Спроектировать и реализовать базу данных для просмотра и размещения развлекательного контента. Разработать Web-приложение и реализовать интерфейс, который позволит работать с базой данных для обработки информации о пользователях, комментариях, оценках и записях. В записях могут содержаться текст, картинки и теги._____

Оформление курсового проекта:

Расчетно-пояснительная записка на 20-30 листах формата А4.

Расчетно-пояснительная записка должна содержать постановку введение, аналитическую часть, конструкторскую часть, технологическую часть, экспериментально-исследовательский раздел, заключение, список литературы, приложения.

Перечень графического (иллюстративного) материала (чертежи, плакаты, слайды и т.п.): на защиту проекта должна быть представлена презентация, состоящая из 15-20 слайдов. На слайдах должны быть отражены: постановка задачи, использованные методы и алгоритмы, расчетные соотношения, структура комплекса программ, диаграмма классов, интерфейс, характеристики разработанного ПО, результаты проведенных исследований.

Дата выдачи задания « ___ » _____ 20__ г.

Руководитель курсового проекта _____ _____А.Л. Исаев_____
(Подпись, дата) (И.О.Фамилия)

Студент _____ _____А.Д. Сучков_____
(Подпись, дата) (И.О.Фамилия)

РЕФЕРАТ

Курсовой проект представляет собой реализацию приложения для размещения развлекательного контента в записях.

Ключевые слова: web-приложение, блог, записи, развлекательный контент, PostgreSQL, React JS, Express.

Приложение реализуется на языке программирования TypeScript с использованием фреймворков React JS и Express.

Полученное в результате работы ПО является приложением для создания и просмотра записей, которые могут содержать изображения и текст, с возможностью их комментировать. Приложение может быть использовано любым человеком в повседневной жизни.

Отчёт содержит 34 страницу, 19 рисунков, 1 таблицу, 8 источников.

Оглавление

Введение

В настоящее время огромную популярность имеют социальные сети, форумы и, так называемые, имиджборды. Все они позволяют людям общаться между собой из разных точек земли, делится разного рода цифровым контентом, оценивать работы художников и писателей или обсуждать новости. Из этого следует, что общение в социальных сетях сегодня стало неотъемлемой частью жизни современного человека.

Цель курсовой работы – реализовать приложение для взаимодействия с записями, которые могут содержать изображения, текст, теги и комментарии.

Чтобы достигнуть поставленной цели, требуется решить следующие задачи:

1. формализовать задание, определить необходимый функционал;
2. провести анализ существующих СУБД;
3. описать структуру базы данных, включая объекты, из которых она состоит;
4. спроектировать приложение для доступа к БД;
5. разработать программное обеспечение, которое позволит пользователю взаимодействовать с записями.

1. Аналитическая часть

В данном разделе представлены постановка задачи, формализация данных и анализ существующих моделей баз данных.

1.1 Постановка задачи

Необходимо спроектировать и разработать приложение для взаимодействия с записями, которые содержат развлекательный контент. Пользователь должен иметь возможность просматривать, создавать, удалять и комментировать записи.

1.2 Формализация данных

В соответствии с поставленной задачей необходимо разработать клиент-серверное веб-приложение с возможностью аутентификации пользователей.

Для каждого типа пользователя предусмотрен свой набор функций.

Неавторизованный пользователь (гость):

- просмотр записей с контентом;
- зарегистрировать новый аккаунт или войти в существующий.

Авторизованный пользователь:

- просмотр записей с контентом;
- просмотр комментариев к записям;
- создание записей;
- создание комментариев под записями;
- выйти из аккаунта.

Администратор (модератор):

- весь набор функций авторизованного пользователя;
- удаление постов;
- удаление комментариев.

База данных должна хранить информацию о:

- записях;
- изображениях, текстах и тегах записей;

- комментариях к записям;
- учетной записи пользователя.

В таблице 1.1 приведены категории и сведения о данных.

Таблица 1 – категории и сведения о данных

Категория	**Сведения**
Запись	Автор, рейтинг, дата и время публикации
Комментарий	Запись, автор, содержание комментария, дата публикации
Теги	Имя
Изображение	Ссылка, теги
Текст	Содержание, теги
Пользователь	Имя, логин, адрес почты, соль, хеш, права доступа

1.3 Анализ моделей баз данных

Система управления базами данных (сокр. СУБД) — совокупность программных и лингвистических средств общего или специального назначения, обеспечивающих управление созданием и использованием баз данных [1].

1.3.1 Основные функции СУБД

Основными функциями СУБД являются:

- управление данными во внешней памяти;
- управление данными в оперативной памяти с использованием дискового кэша;
- журнализация изменений, резервное копирование и восстановление базы данных после сбоев;
- поддержка языков БД.

1.3.2 Классификация СУБД по модели данных

Модель данных — это абстрактное, самодостаточное, логическое определение объектов, операторов и прочих элементов, в совокупности составляющих абстрактную машину доступа к данным, с которой взаимодействует пользователь. Эти объекты позволяют моделировать структуру данных, а операторы — поведение данных [2].

На данный момент, существует 3 основных типа моделей организации данных:

- иерархическая;
- сетевая;
- реляционная.

В иерархической модели используется представление базы данных в виде древовидной структуры, которая состоит из объектов различных уровней. Между объектами существуют связи и каждый объект может включать в себя несколько объектов более низкого уровня. Такие объекты находятся в отношении предка к потомку, при этом возможна ситуация, когда объект-предок имеет несколько потомков, тогда как у объекта-потомка обязателен только один предок.

На рисунке 1 представлена структура иерархической модели данных.

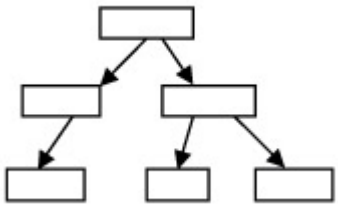

Рис. 1 – структура иерархической модели данных

В сетевой модели, потомок может иметь любое число предков. Такая БД состоит из набора экземпляров определенного типа записи и набора экземпляров определенного типа связей между этими записями. Главным недостатком сетевой модели данных являются жесткость и высокая сложность

схемы базы данных, построенной на основе этой модели. Так как логика процедуры выбора данных зависит от физической организации этих данных, то эта модель не является полностью независимой от приложения.

На рисунке 2 представлена структура сетевой модели данных.

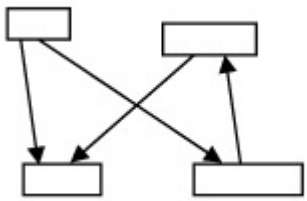

Рис. 2 – структура сетевой модели данных

Реляционная модель данных является совокупностью данных и состоит из набора двумерных таблиц. За счет табличной организации достигается отсутствие иерархии элементов. Таблицы состоят из строк – записей и столбцов – полей. На пересечении строк и столбцов находятся конкретные значения. Для каждого поля определяется множество его значений. За счет возможности просмотра строк и столбцов в любом порядке достигается гибкость выбора подмножества элементов. Реляционная модель, на данный момент, является наиболее удобной и широко используемой формой представления данных.

На рисунке 3 представлена структура реляционной модели данных.

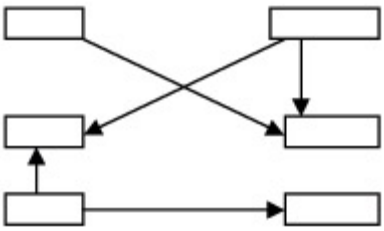

Рис. 3 – структура реляционной модели данных

В результате анализа моделей баз данных, в соответствии с поставленной задачей, наиболее оптимальным решением является использование реляционной модели базы данных, так как это позволит реализовать поставленные цели, не усложняя программную архитектуру.

Вывод

В разделе выше была поставлена задача реализации приложения для взаимодействия с записями с контентом и проведен анализ моделей баз данных, исходя из которого было решено использовать реляционную модель базы данных, так как это позволит намного эффективнее и быстрее реализовать поставленные задачи.

2. Конструкторская часть

В данном разделе рассматриваются структура приложения и базы данных.

2.1 Функциональная модель

На рисунке 4 изображена функциональная модель, отображающая структуру и функции системы.

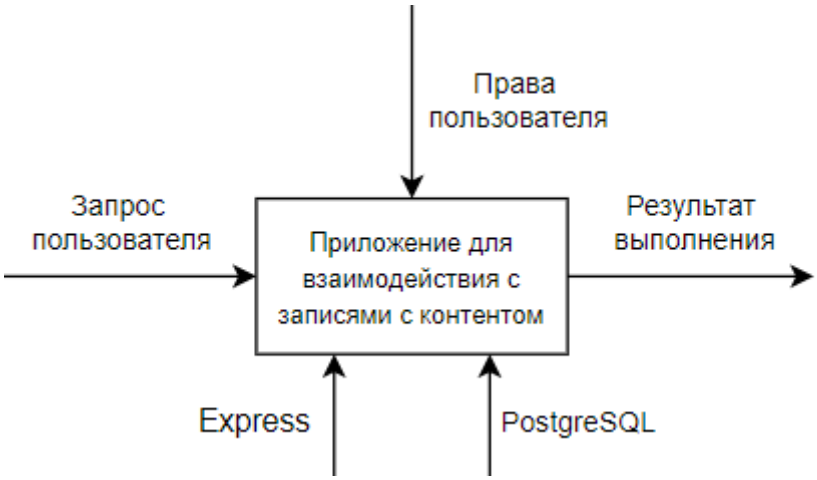

Рис. 4 – Функциональная модель.

2.2 Сценарий использования

В разделе необходимо построить диаграмму прецедентов (Use Case Diagram), состоящая из графической диаграммы, описывающей действующие лица и прецеденты – конкретные действия, которые выполняет пользователь при работе с системой.

Данная диаграмма предназначена для определения функциональных требований. В системе есть три типа пользователей:

- неавторизованный (гость);
- авторизованный;
- администратор (модератор).

На рисунке 5 представлена Use Case Diagram для действий всех трёх типов пользователей.

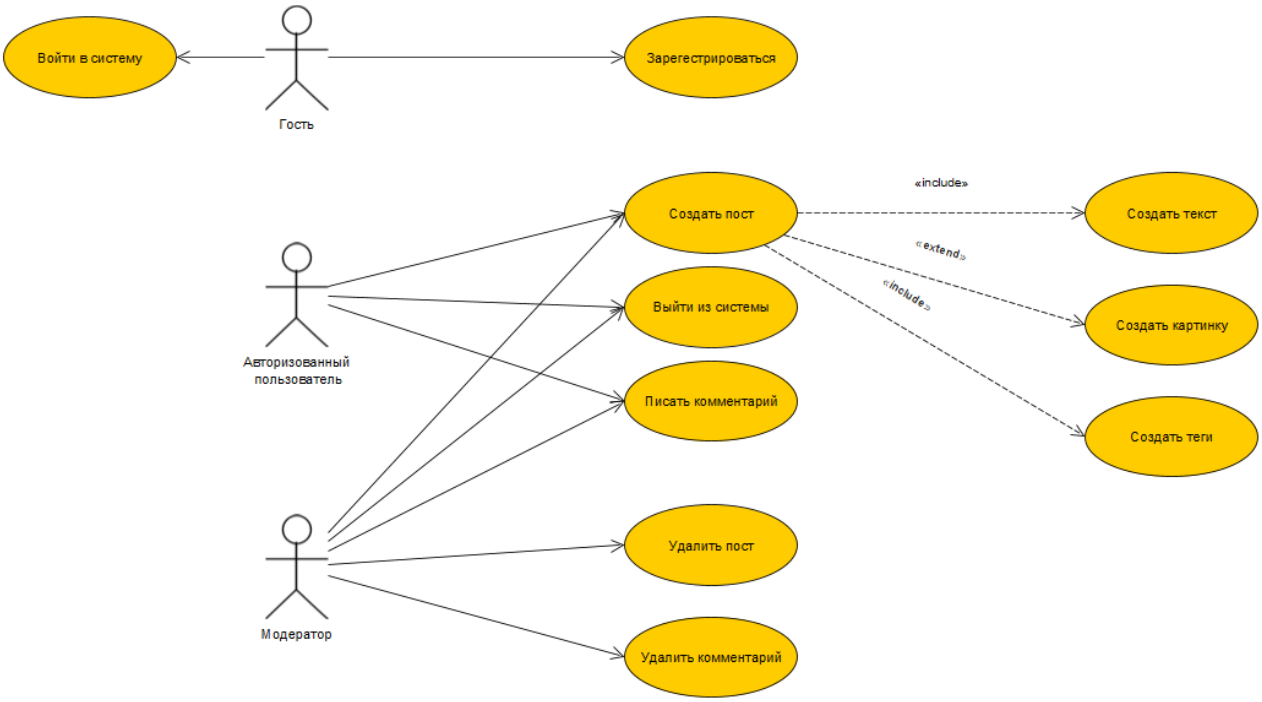

Рис. 5 – Use Case Diagram для трёх типов пользователей.

2.3 Проектирование базы данных

База данных должна хранить рассмотренные в таблице 1 данные. В соответствии с этой таблицей можно выделить следующие таблицы:

- записей – Post;
- комментариев к записям – Review;
- тегов для изображений и текстов – Tag;
- изображений – Post_pict;
- текстов – Post_text;
- пользователей – Account.

Таблица **Post** должна хранить информацию о записях:

- id – уникальный идентификатор записи, PK, serial;
- author_id – идентификатор автора, FK, date;
- rating – рейтинг записи, integer;
- public_date – дата и время публикации, date;

Таблица **Review** должна хранить информацию о комментариях под записями:

- id - уникальный идентификатор комментария, PK, serial;
- post_id – идентификатор записи, FK, integer;
- auth_id – идентификатор автора комментария, FK, integer;
- review_data – содержание комментария, text;
- public_date – время публикации комментария, date;

Таблица **Tag** должна хранить информацию о тегах изображений и текстов:

- id – уникальный идентификатор тега, PK, serial;
- name – название, varchar;

Таблица **Post_pict** должна хранить информацию о изображениях в записях:

- id – уникальный идентификатор изображения, PK, serial;
- path – путь до изображения, varchar;

Таблица **Post_text** должна хранить информацию о текстах в записях:

- id – уникальный идентификатор текста, PK, serial;
- data – содержание текста, varchar;

Таблица **Account** должна хранить информацию о пользователях:

- id – уникальный идентификатор пользователя, PK, serial;
- name – имя, varchar;
- login – логин, varchar, unique;
- email – адрес почты, varchar, unique;
- avatar – путь для аватара на сайте, varchar, unique;
- salt – соль для хранения пароля, varchar, unique;
- hash – хэш для проверки пароля, varchar, unique;
- role – права доступа пользователя, varchar

На рисунке 6 представлена ER – модель базы данных.

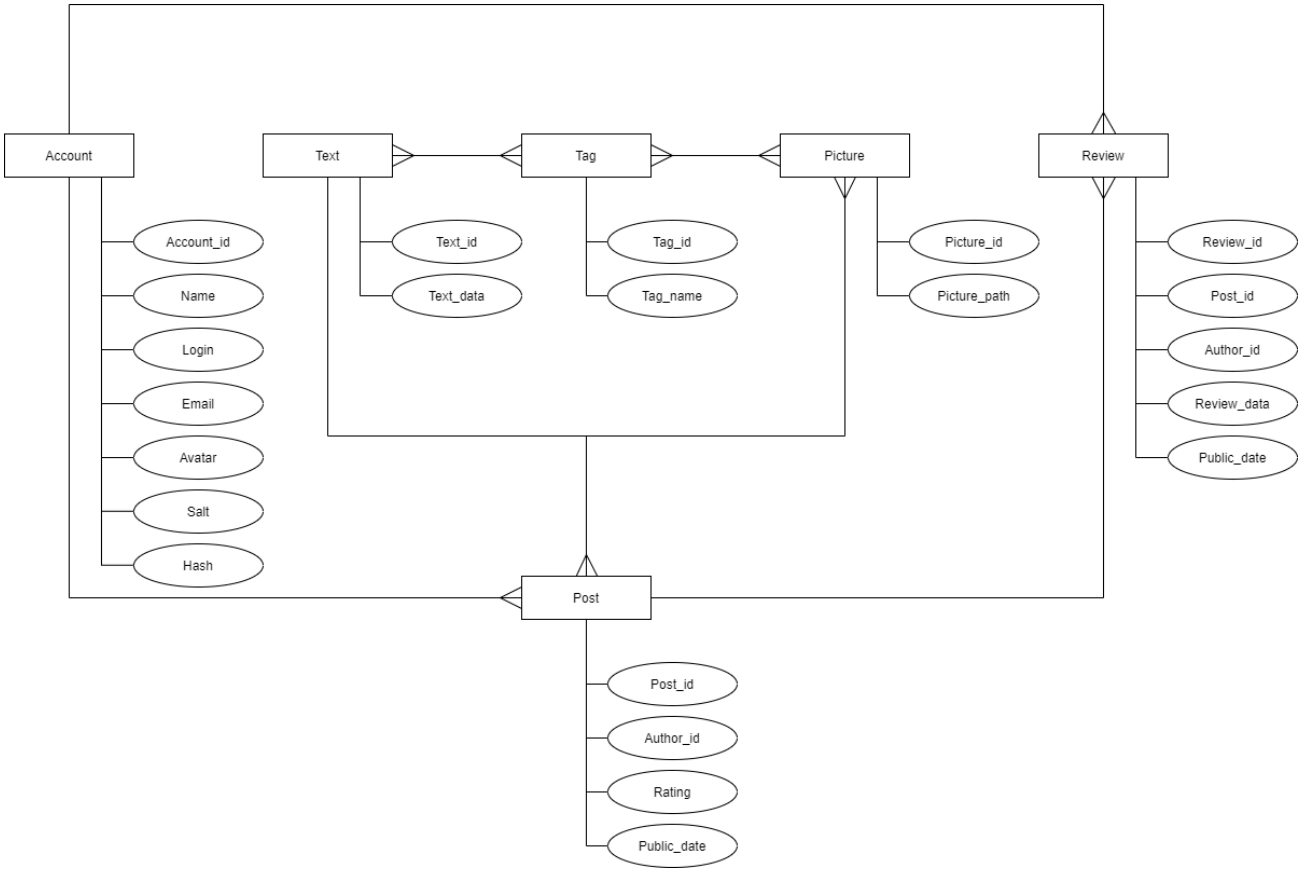

Рис. 6 – ER – модель базы данных.

2.4 Проектирование процедур и триггеров

Для более полного и корректного очищения данных при удалении записей с контентом необходимо организовать процедуры, которые будут очищать содержащиеся в удаляемой записи изображения, тексты и комментарии из соответствующих таблиц.

Схема процедуры для удаления изображений по входному идентификатору удаляемой записи, приведена на рисунке 7.

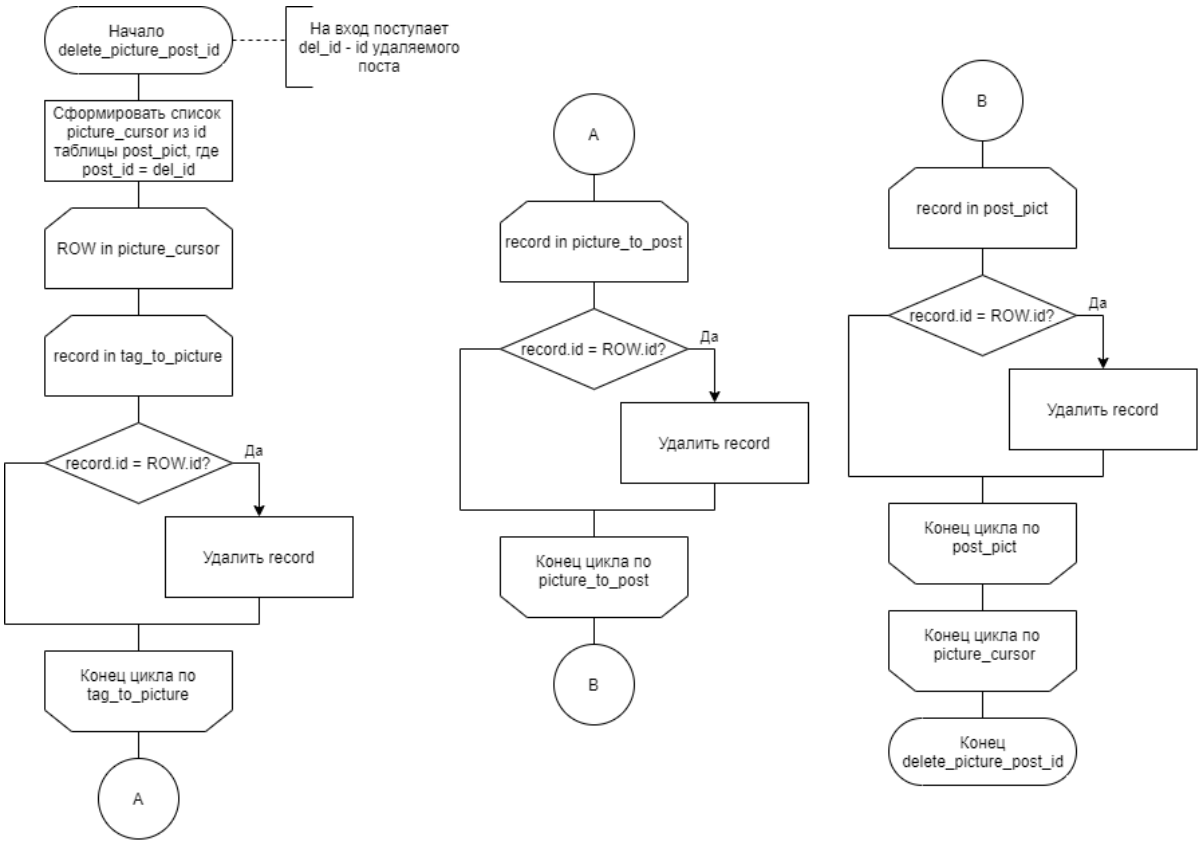

Рис. 7 – Схема процедуры удаления изображений по идентификатору.

Схема процедуры для удаления текстов аналогична схеме на рисунке 7.

Схема удаления записей и их содержимого по входному идентификатору приведена на рисунке 8.

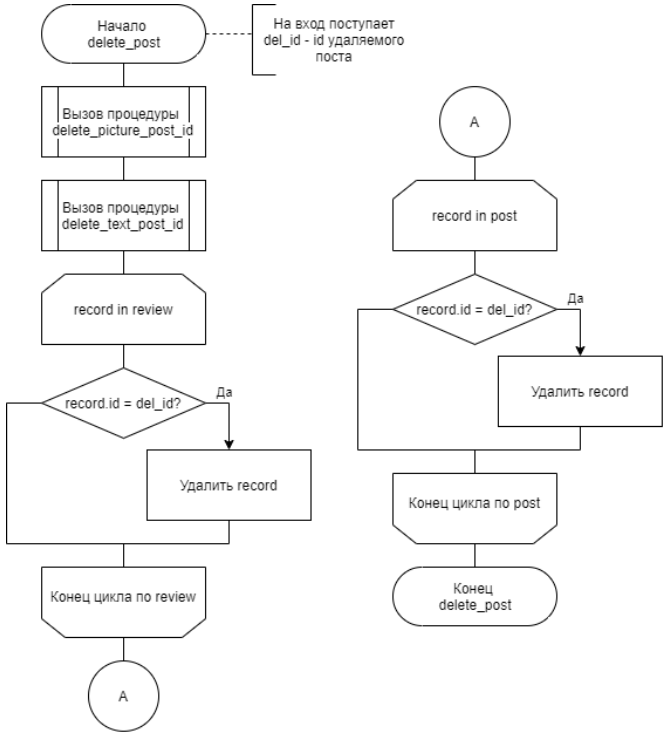

Рис. 8 – Схема процедуры удаления записи.

В случае добавления новых пользователей в базу данных, необходимо обеспечить валидацию минимального набора данных (логин и пароль, который хранится в виде соли и хеша), при которых пользователь сможет войти в систему даже при системной ошибке в приложении.

Схема триггера на добавление новых записей в таблицу пользователей приведена на рисунке 9.

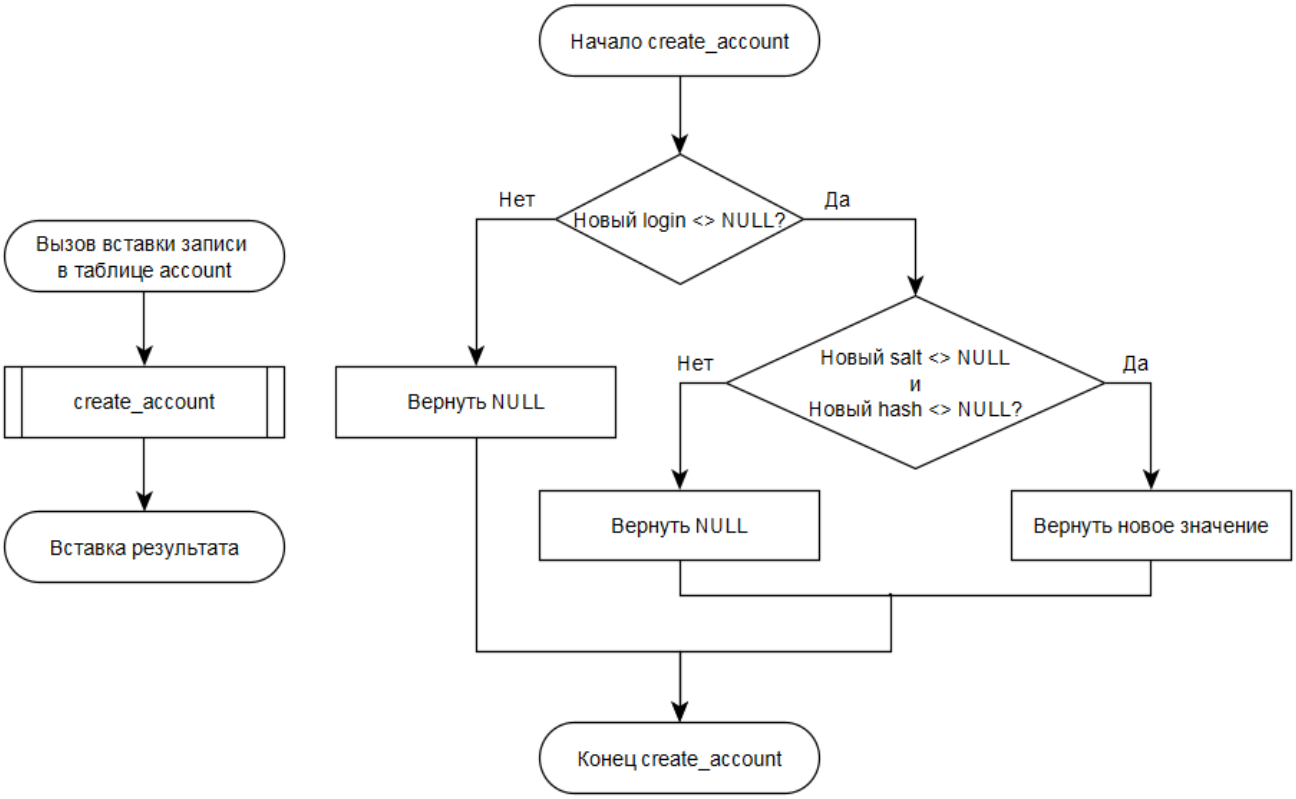

Рис. 9 – Схема триггера на добавление новых пользователей.

Также необходимо обеспечить хранение данных хотя бы оного администратора, во избежание проблем с дальнейшим администрированием системы. Для этого необходимо при удалении пользователей проверять количество администраторов.

Схема триггера на удаление записей из таблицы пользователей приведена на рисунке 10.

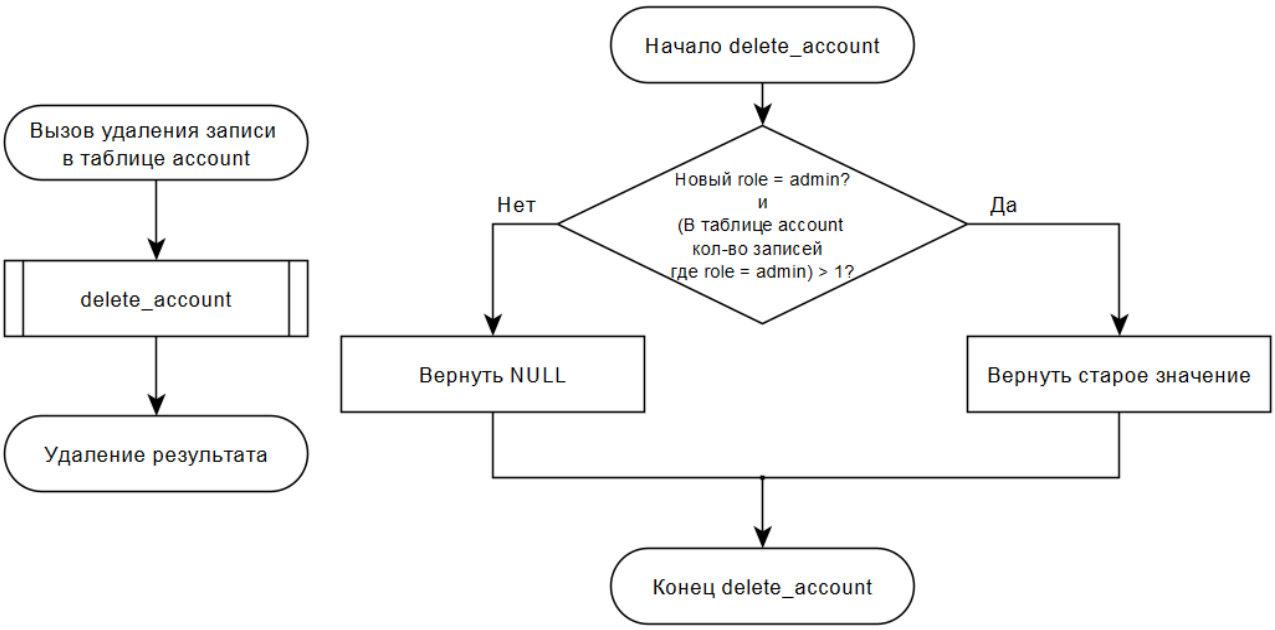

Рис. 10 – Схема триггера на удаление пользователей.

2.5 Проектирование архитектуры приложения

При построении архитектуры приложения был выбран подход к созданию клиент-серверного web-приложения.

В пользовательской части приложения, визуализируется интерактивный и понятный интерфейс, выделяются сервисы, отвечающие за обмен данных с серверной частью приложения.

За логику, работоспособность и правильное функционирование сайта отвечает серверная часть, которая скрыта от пользователя. Серверная часть web-приложения, где обрабатываются пользовательские запросы, включает в себя менеджер, сервисы и репозитории. В менеджере содержатся сервисы, с помощью которых запрос обрабатывается и формируется ответ. Сервисы содержат репозитории, которые включают в себя базовый класс хранилища, описывающий базовые методы для обращения к базе данных.

На рисунке 11 представлена архитектура приложения.

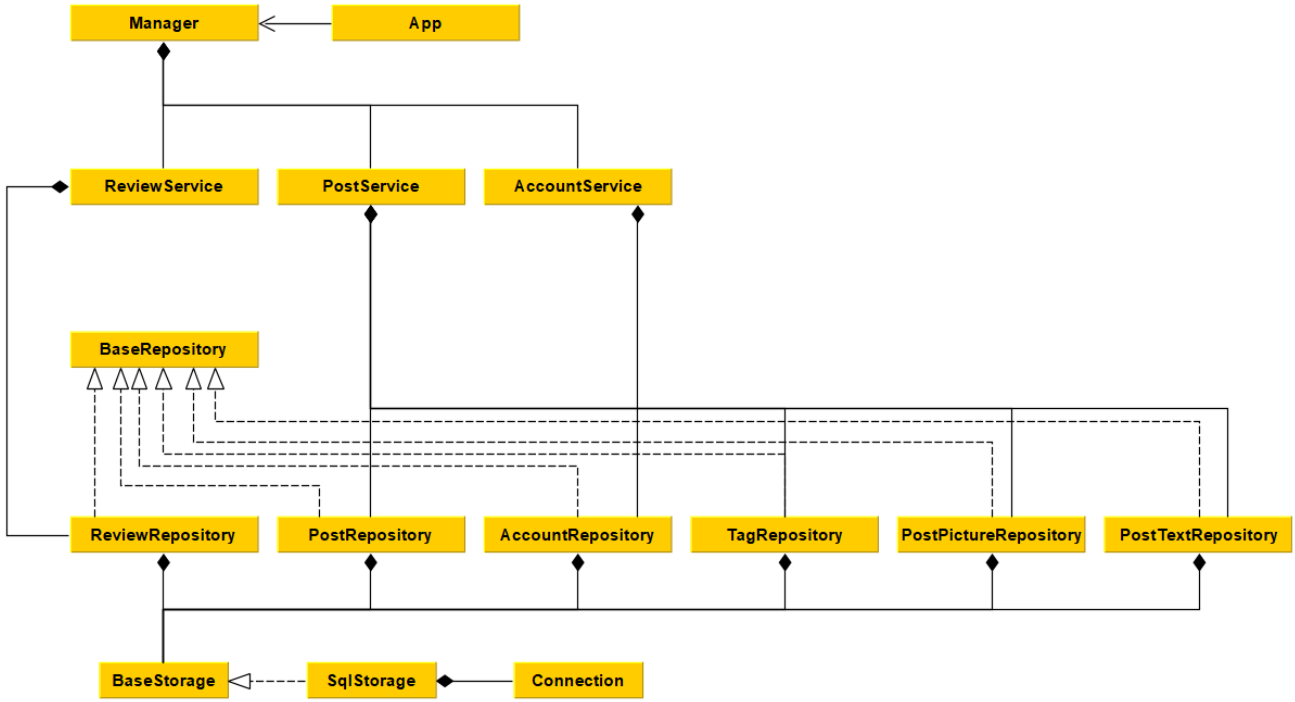

Рис. 11 – Архитектура приложения.

Вывод

В данном разделе были спроектированы база данных, процедуры, триггеры и архитектура приложения.

3. Технологическая часть

В данном разделе необходимо определить технологический стек для разработки программного обеспечения.

3.1. Выбор реляционной СУБД

Наиболее популярными РСУБД являются PostgreSQL, Microsoft SQL Server, MySQL и Oracle.

3.1.1 PostgreSQL

PostgreSQL — реляционная СУБД, которая отличается от других тем, что обладает объектно-ориентированным функционалом, в том числе полной поддержкой концепта ACID (Atomicity, Consistency, Isolation, Durability).

Достоинства:

- поддержка сторонними организациями;
- полная SQL-совместимость;
- является масштабируемой и способна обрабатывать терабайты данных;
- расширяемость за счёт использования хранимых процедур.

Недостатки:

- во время пакетных операций или выполнения запросов чтения, возможно уменьшение скорости работы может.

3.1.2 Microsoft SQL Server

Microsoft SQL Server — это СУБД, движок которой работает на облачных серверах, а также локальных серверах, с возможностью комбинировать типы применяемых серверов одновременно.

Достоинства:

- простота использования;
- текущая версия работает быстро и стабильно;
- движок предоставляет возможность регулировать и отслеживать уровни производительности, которые помогают снизить использование ресурсов;
- визуализация на мобильных устройствах.

Недостатки:

- высокая стоимость продукта;
- высокая ресурсоемкость SQL Server;
- возможны проблемы с использованием службы интеграции для импорта файлов.

3.1.3 MySQL

MySQL — реляционная СУБД с открытым исходным кодом с моделью клиент-сервер.

Достоинства:

- простота использования;
- поддерживает большую часть функционала SQL;
- поддерживает набор пользовательских интерфейсов;
- может работать с другими базами данных, включая Oracle.

Недостатки:

- ограничения функциональности, так как не полностью реализованы SQL-стандарты;
- некоторые операции реализованы менее надёжно, чем в других РСУБД.

3.1.4 Oracle

Oracle — объектно-реляционная СУБД.

Достоинства:

- поддержка огромных баз данных и большого числа пользователей;
- быстрая обработка транзакций.

Недостатки:

- высокая стоимость;
- требуются значительные вычислительные ресурсы.

В результате анализа реляционных СУБД, в соответствии с поставленной задачей, наиболее оптимальным решением является использование PostgreSQL, так как эта СУБД обеспечивает целостность данных, поддерживает сложные структуры и широкий спектр встроенных и определяемых пользователем типов данных.

3.2 Выбор и обоснование инструментов разработки

Анализируя современные тренды web-разработки, было выявлено, что наиболее выигрышным является подход к созданию web-приложений, в виде SPA (Single Page Application – одностраничное веб-приложение, которое загружается на одну HTML-страницу).

Преимущества использования SPA определяется следующими пунктами:

- Одностраничные приложения работают значительно быстрее обычных сайтов. Скорость загрузки у них выше, соответственно, они удобнее пользователям.
- Есть необходимость в многофункциональном, насыщенном пользовательском интерфейсе.
- SPA лучше адаптировано под мультиплатформенность: такое веб-приложение отлично показывает себя на любых устройствах и браузерах.

Динамическое обновление страницы, происходящее благодаря скриптам, написанным на языке JavaScript, позволяет пользователю не перезагружать страницу при переходе к другому блоку приложения. В процессе работы пользователю может показаться, что он использует не веб-сайт, а десктопное приложение, так как оно мгновенно реагирует на все его действия, без задержек и «подвисаний». Добиться такого эффекта позволяет использование фреймворков для языка JavaScript, таких как: Angular, React JS, Vue.

Так как, фреймворк React JS предоставляет такую функциональность, как перерисовывание компонента сайта при изменении его состояния, что позволяет динамически изменять данные в одном месте интерфейса при изменении данных модели в другом, а также имеет маршрутизацию [3], было принято решение использовать именно его.

Чтобы избежать ошибки, с которыми часто сталкиваются разработчики на JavaScript, используется дополнительное расширение JavaScript – TypeScript, которое позволяет работать со статической типизацией [4].

Для реализации серверной части был выбран Express, который является Node JS фреймворком, который позволяет создавать большие масштабируемые приложения.

3.3 Реализация бизнес-логики

Бизнес-логика отвечает за обработку запросов от пользователя, проверку и обработку переданных данных, и формирование ответа на запрос. Для бизнес-логики реализован класс менеджера, который содержит в себе сервисы для валидации, в случае передачи данных в запросе, обращения к базе данных и последующей передаче данных для формирования ответа на запрос.

3.4 Администрирование БД

С использованием pgAdmin (программное обеспечение, предоставляющее графический интерфейс для работы с базой данных) было проведено создание необходимых таблиц и их проверка на корректное создание базы данных с необходимыми ключами, создание ролей и триггеров.

В листинге 1 представлена часть кода создания некоторых таблиц в базе данных.

Листинг 1 – Код создания нескольких таблиц в базе данных.

```
CREATE TABLE IF NOT EXISTS post (
   id         SERIAL  NOT NULL PRIMARY KEY,
   author_id   INTEGER NOT NULL REFERENCES account(id),
   rating      INTEGER NOT NULL,
   public_date DATE    NOT NULL
);

CREATE TABLE IF NOT EXISTS review (
   id         SERIAL  NOT NULL PRIMARY KEY,
   post_id     INTEGER NOT NULL REFERENCES post(id),
```

```
    auth_id     INTEGER NOT NULL REFERENCES account(id),
    review_data TEXT    NOT NULL,
    public_date DATE    NOT NULL
);

CREATE TABLE IF NOT EXISTS account (
    id SERIAL NOT NULL PRIMARY KEY,
    name VARCHAR(32) NOT NULL,
    login VARCHAR(32) NOT NULL UNIQUE,
    email VARCHAR(64) NOT NULL UNIQUE,
    avatar VARCHAR NOT NULL UNIQUE,
    salt VARCHAR NOT NULL UNIQUE,
    hash VARCHAR NOT NULL,
    role VARCHAR NOT NULL
);
```

На рисунке 12 продемонстрировано существование таблиц в БД.

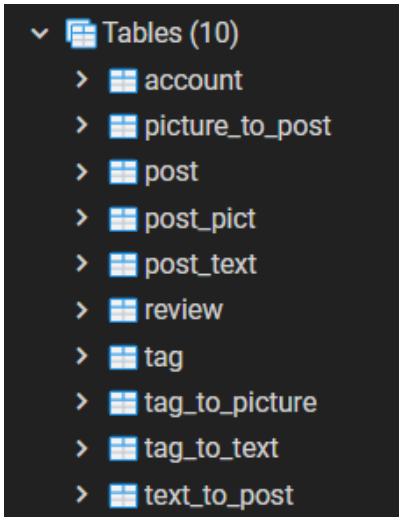

Рис. 12 – Таблицы БД.

Для реализации поставленных задач необходимо наличие ролевой модели на уровне базы данных. Роль – это разрешение, предоставляемое группе пользователей для доступа к данным. Было выделено три роли:

- guest – может просматривать данные таблиц записей, изображений, текстов, тегов, комментариев и пользователей, а также добавлять новые записи в таблицу пользователей;
- log_user – может просматривать и добавлять данные таблиц, но не удалять;

- admin(postgres) – может добавлять, редактировать и удалять данные из всех таблиц.

В листинге 2 представлена реализация ролевой модели на уровне базы данных.

Листинг 2 – Реализация ролевой модели на уровне базы данных.

```
CREATE USER guest WITH PASSWORD 'guest';
CREATE USER log_user WITH PASSWORD 'log_user';

-- Unlogged user
CREATE ROLE unlogged_role;

GRANT SELECT ON account TO unlogged_role;
GRANT INSERT ON account  TO unlogged_role;
GRANT SELECT ON picture_to_post TO unlogged_role;
GRANT SELECT ON post TO unlogged_role;
GRANT SELECT ON post_pict TO unlogged_role;
GRANT SELECT ON post_text TO unlogged_role;
GRANT SELECT ON review TO unlogged_role;
GRANT SELECT ON tag TO unlogged_role;
GRANT SELECT ON tag_to_picture TO unlogged_role;
GRANT SELECT ON tag_to_text TO unlogged_role;
GRANT SELECT ON text_to_post TO unlogged_role;

GRANT unlogged_role TO guest;

-- Logged user
CREATE ROLE logged_role;

GRANT SELECT ON account TO logged_role;
GRANT SELECT ON picture_to_post TO logged_role;
GRANT SELECT ON post TO logged_role;
GRANT SELECT ON post_pict TO logged_role;
GRANT SELECT ON post_text TO logged_role;
GRANT SELECT ON review TO logged_role;
GRANT SELECT ON tag TO logged_role;
GRANT SELECT ON tag_to_picture TO logged_role;
GRANT SELECT ON tag_to_text TO logged_role;
GRANT SELECT ON text_to_post TO logged_role;

GRANT INSERT ON account TO logged_role;
GRANT INSERT ON picture_to_post TO logged_role;
GRANT INSERT ON post TO logged_role;
GRANT INSERT ON post_pict TO logged_role;
GRANT INSERT ON post_text TO logged_role;
GRANT INSERT ON review TO logged_role;
GRANT INSERT ON tag TO logged_role;
GRANT INSERT ON tag_to_picture TO logged_role;
```

```
GRANT INSERT ON tag_to_text TO logged_role;
GRANT INSERT ON text_to_post TO logged_role;
GRANT logged_role TO log_user;
```

В ходе реализации программного обеспечения был создан триггер Before для таблицы account на удаление, где проверяется количество администраторов и, в случае, когда администратор остался один, его нельзя удалить.

Также создан триггер Before для таблицы account на вставку, где проверяется наличие логина, соли и хэша для пароля, которые обеспечивают вход в систему.

В листинге 3 представлена реализация триггеров Before для таблицы account на вставку и удаление.

Листинг 3 – Реализация триггеров Before для таблицы account.

```
CREATE FUNCTION trigger_create_account() RETURNS TRIGGER AS
$$
BEGIN
	IF (NEW.login IS NOT NULL) THEN
		IF (NEW.salt IS NOT NULL AND NEW.hash IS NOT NULL) THEN
				RETURN NEW;
		END IF;
	END IF;

	RETURN NULL;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER create_account
BEFORE INSERT ON account FOR EACH ROW
EXECUTE PROCEDURE trigger_create_account();

CREATE FUNCTION trigger_delete_account() RETURNS TRIGGER AS
$$
BEGIN
	IF (OLD.role = 'admin' and (SELECT count(*) FROM account WHERE account.role =
'admin') = 1)
	THEN
		RETURN NULL;
	ELSE
		RETURN OLD;
	END IF;
END;
```

```
$$ LANGUAGE  plpgsql;

CREATE TRIGGER delete_account
BEFORE DELETE ON account FOR EACH ROW
EXECUTE PROCEDURE trigger_delete_account();
```

Для корректного удаления записей и содержащихся в них картинок, текстов и комментариев, а также их связей, были реализованы специальные процедуры.

В листинге 4 представлена реализация процедур удаления картинок, текстов, комментариев и постов по входному id поста.

Листинг 4 – Реализация процедур удаления картинок, текстов, комментариев и постов.

```
CREATE OR REPLACE PROCEDURE delete_picture_post_id(del_id INT) AS
$$
DECLARE pict_cursor CURSOR
	FOR
		SELECT post_pict.id FROM post_pict JOIN picture_to_post
		ON post_pict.id = picture_to_post.pict_id
		WHERE picture_to_post.post_id = del_id;
	ROW RECORD;
BEGIN
	OPEN pict_cursor;

	LOOP
		FETCH pict_cursor INTO ROW;
		EXIT WHEN NOT FOUND;

		DELETE FROM tag_to_picture WHERE pic_id = ROW.id;
		DELETE FROM picture_to_post WHERE pict_id = ROW.id;
		DELETE FROM post_pict WHERE id = ROW.id;
	END LOOP;

	CLOSE pict_cursor;
END;
$$
LANGUAGE PLPGSQL;

CREATE OR REPLACE PROCEDURE delete_text_post_id(del_id INT) AS
$$
DECLARE text_cursor CURSOR
	FOR
		SELECT post_text.id FROM post_text JOIN text_to_post
		ON post_text.id = text_to_post.text_id
		WHERE text_to_post.post_id = del_id;
	ROW RECORD;
```

```
BEGIN
	OPEN text_cursor;

	LOOP
		FETCH text_cursor INTO ROW;
		EXIT WHEN NOT FOUND;

		DELETE FROM tag_to_text WHERE text_id = ROW.id;
		DELETE FROM text_to_post WHERE text_id = ROW.id;
		DELETE FROM post_text WHERE id = ROW.id;
	END LOOP;

	CLOSE text_cursor;
END;
$$
LANGUAGE PLPGSQL;

CREATE OR REPLACE PROCEDURE delete_post(del_id INT) AS
$$
BEGIN
	CALL delete_picture_post_id(del_id);
	CALL delete_text_post_id(del_id);

	DELETE FROM review WHERE post_id = del_id;
	DELETE FROM post   WHERE id = del_id;
END;
$$
LANGUAGE PLPGSQL;
```

3.7 Интерфейс приложения

Интерфейс приложения разбит на экраны. На рис. 13 – рис. 19 показан интерфейс различных экранов приложения.

Сначала, при заходе в web-приложение неавторизованного пользователя встречает меню аутентификации, рисунки 13 - 14. Здесь, пользователь может войти на сайт как гость, с ограниченным функционалом, войти по существующему аккаунту или зарегистрировать новый.

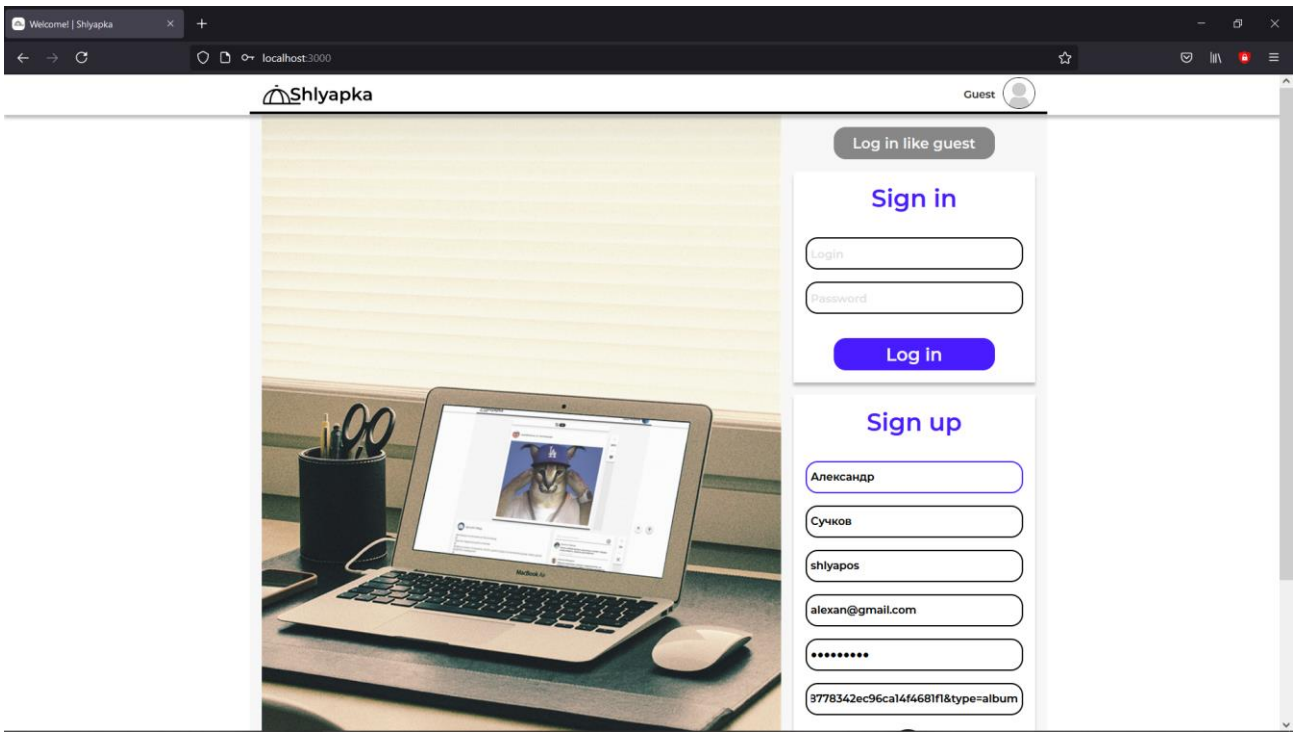

Рис. 13 – Интерфейс экрана входа в качестве гостя или по существующему аккаунту.

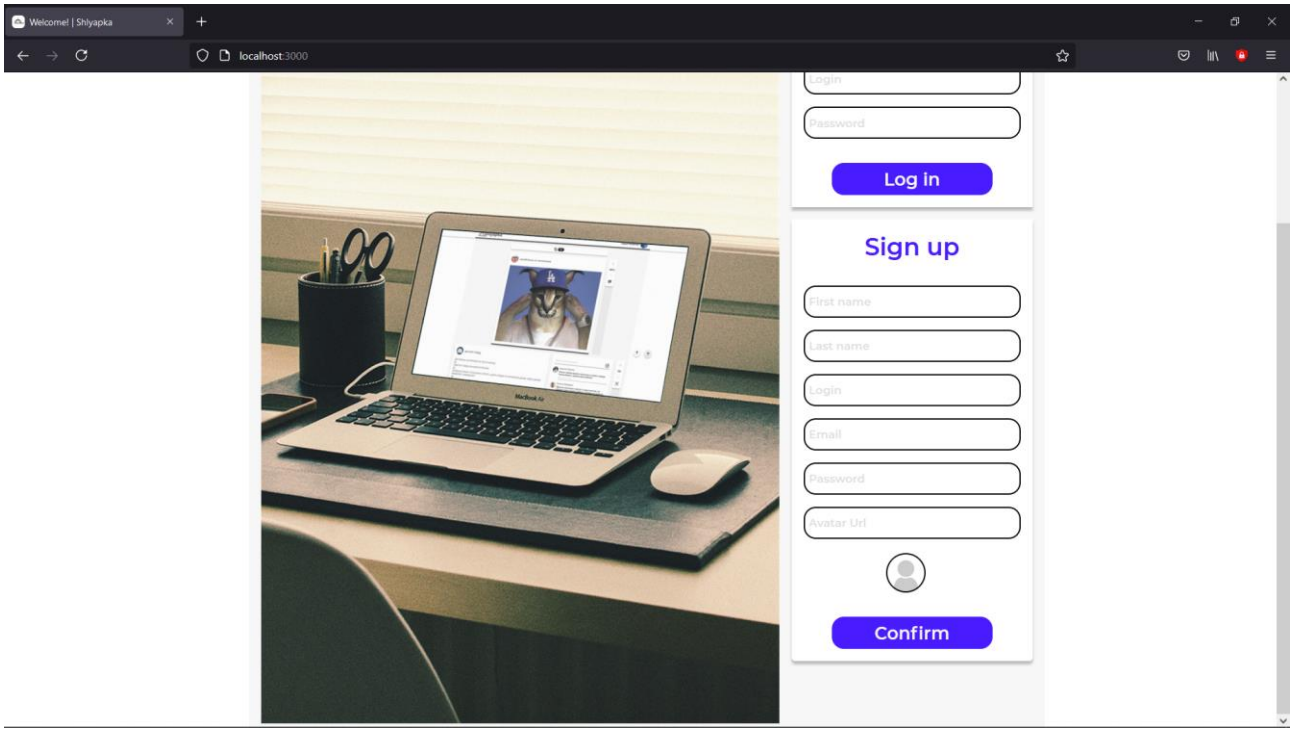

Рис. 14 – Интерфейс экрана регистрации нового пользователя.

Нажав кнопку «Log in like guest» пользователь войдёт на сайт как гость и сможет только просматривать записи, рисунок 15. Также доступна кнопка «Sign in», которая позволяет вернуться на экран аутентификации.

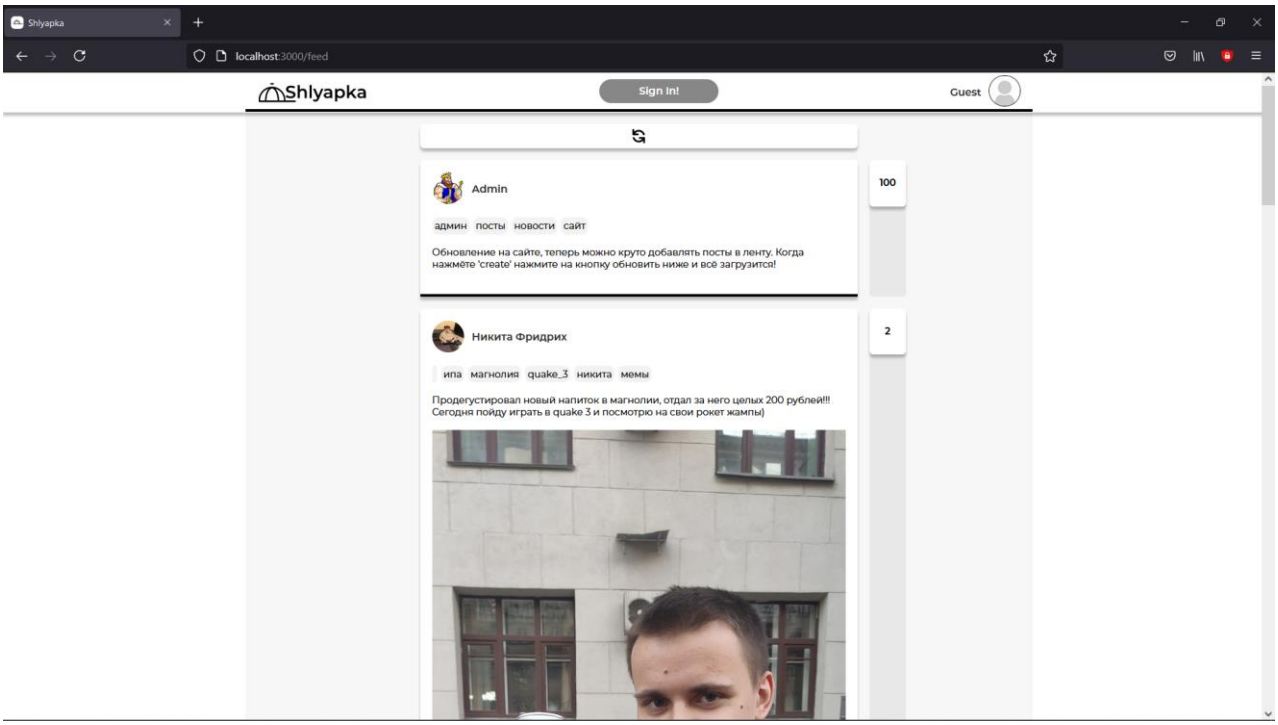

Рис. 15 – Интерфейс главного экрана взаимодействия с записями, в случае роли – гость.

На экране аутентификации, при корректном вводе данных и при нажатии на кнопку подтверждения регистрации «Confirm» пользователь попадёт на главный экран взаимодействия с записями как авторизованный пользователь.

Также на этот экран можно попасть при вводе корректных данных существующего аккаунта и при нажатии на кнопку «Log in». Интерфейс приведён на рисунке 16.

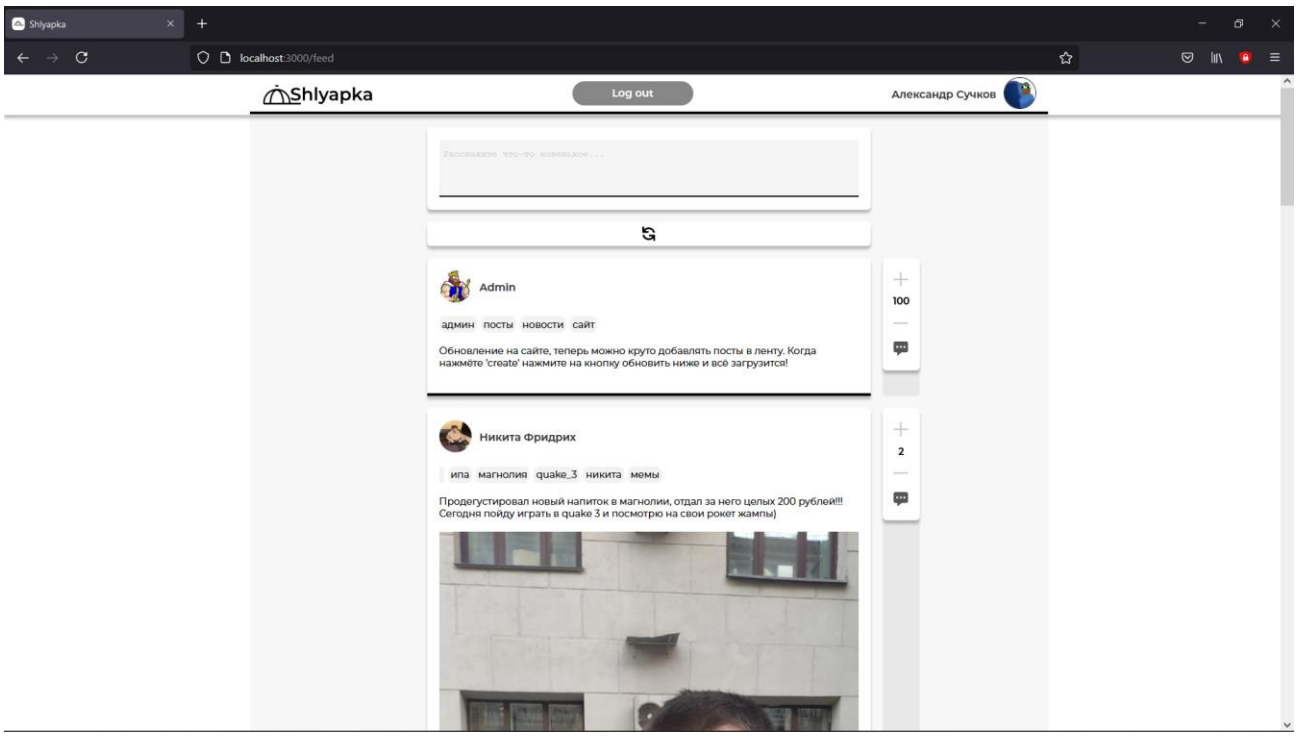

Рис. 16 – Интерфейс главного экрана, в случае роли – авторизованный пользователь.

При нажатии на текстовое поле в верхней части, откроется меню добавления новой записи, где можно написать текст, добавить к тексту теги, вставить ссылку на изображение и добавить теги к изображению, рисунок 17.

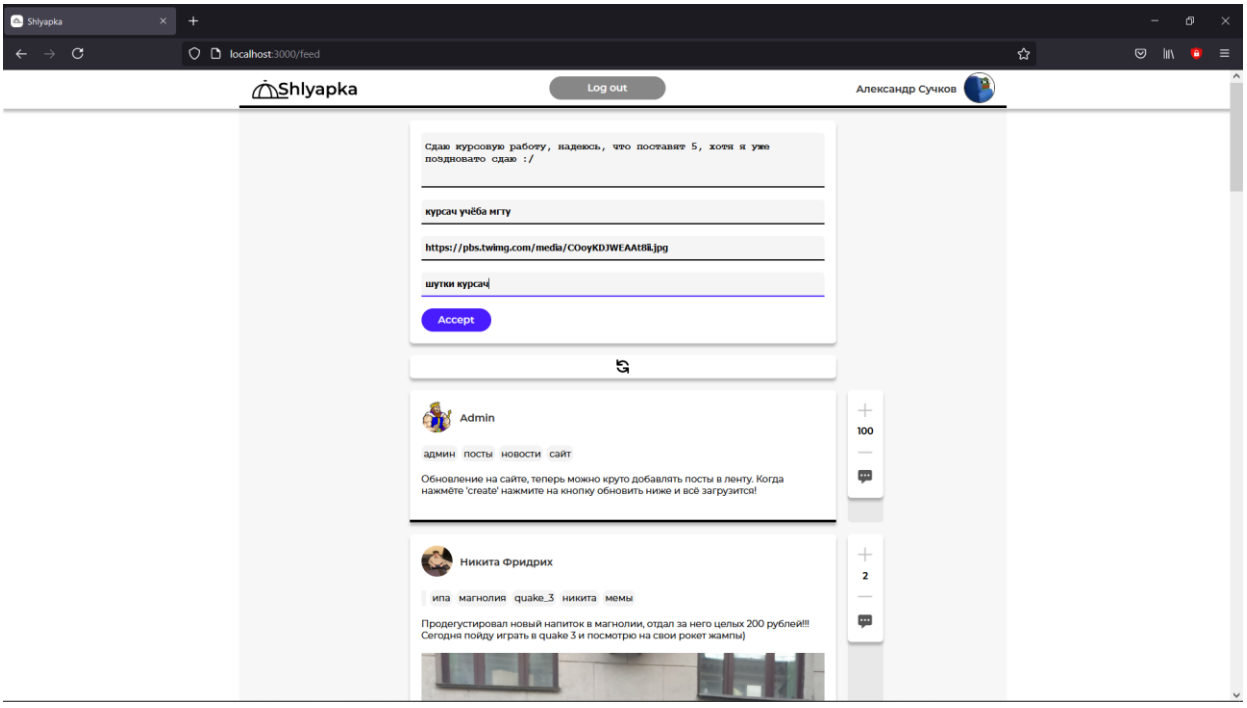

Рис. 17 – Интерфейс добавления новой записи на главном экране.

После нажатия на кнопку «Accept» запись добавиться в общую ленту. Если нажать на значок сообщения справа от записи, то откроется меню комментариев, где их можно просматривать или добавить свой, нажав на стрелочку справа от поля ввода. Также можно увеличивать или уменьшать рейтинг у записи на кнопки «+» и «-» в том же месте. Интерфейс приведён на рисунке 18.

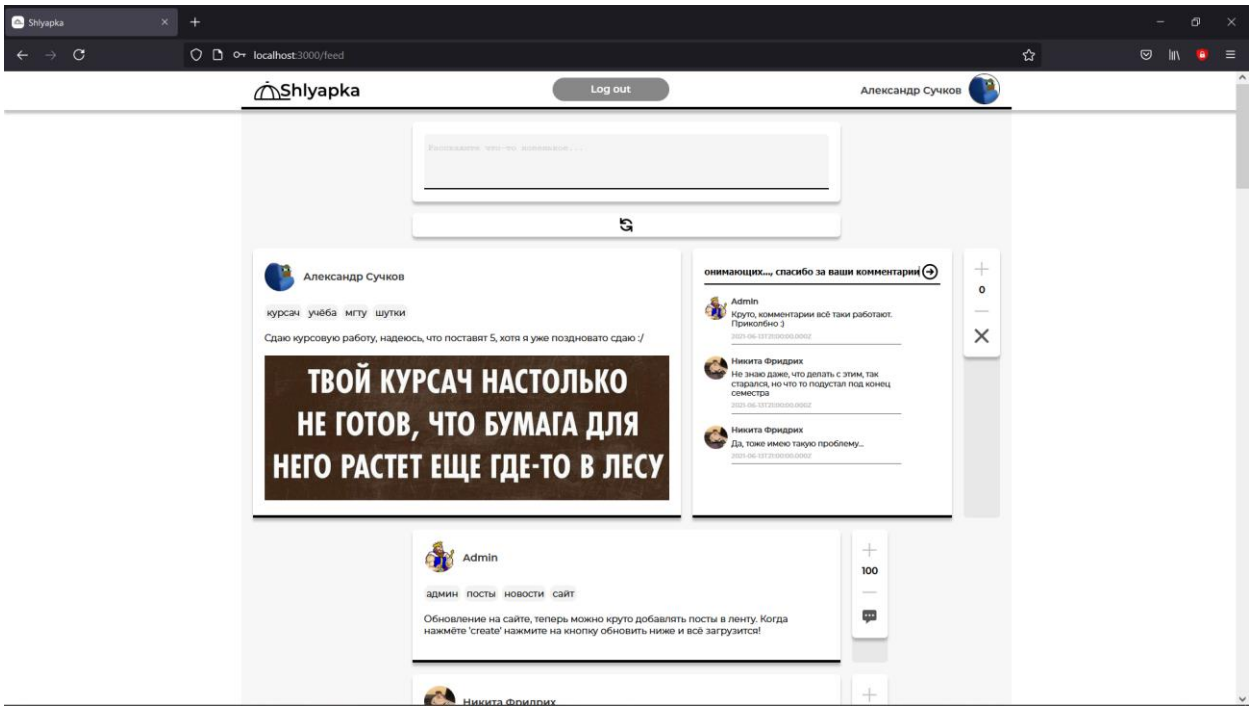

Рис. 18 – Интерфейс просмотра и добавления комментариев

Если войти на сайт с ролью «admin», то появится дополнительная кнопка со значком крестика, которая позволяет удалять записи. Такая же появляется и у комментариев для возможности их удаления, рисунок 19.

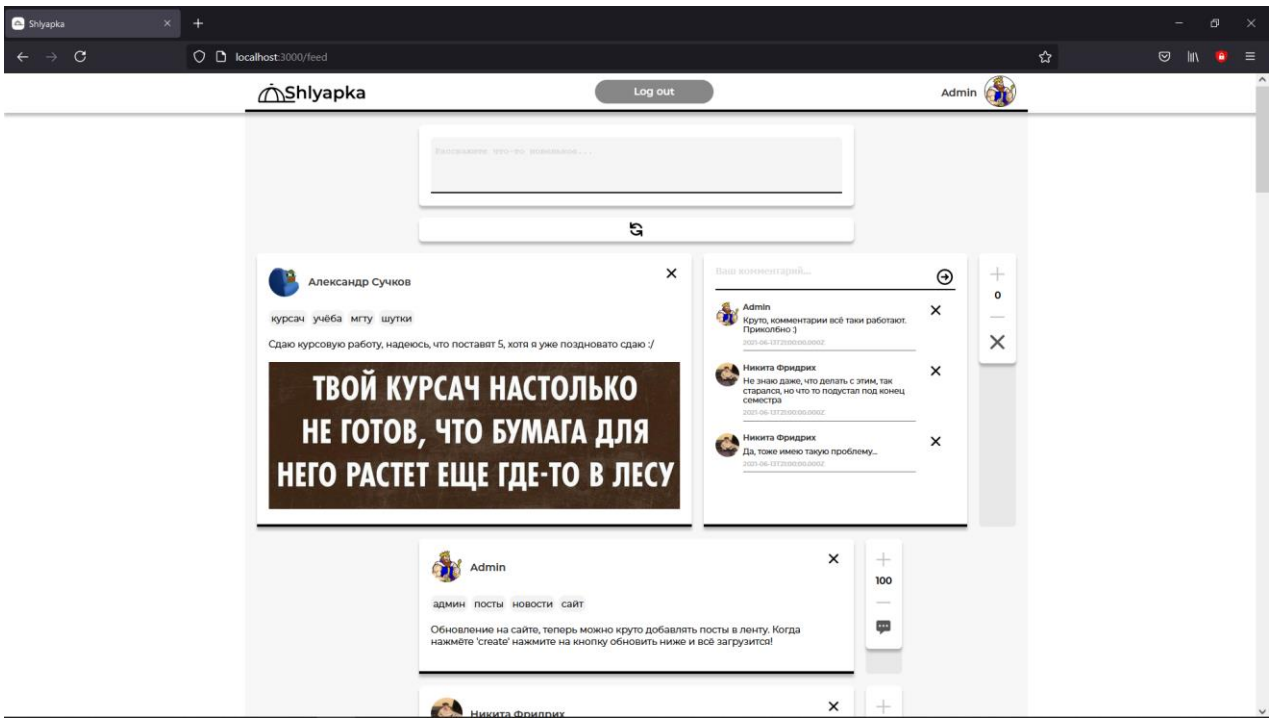

Рис. 19 – Интерфейс удаления записей и комментариев на главном экране.

Вывод

В данном разделе был выбран стек технологий, рассмотрены структуры и состав реализованных классов, предоставлены сведения о модулях приложения и об его интерфейсе.

Заключение

Цель курсовой работы достигнута. Спроектировано и реализовано программное обеспечение для создания, просмотра, удаления и комментирования записей с различным контентом.

В ходе данной работы была формализована задача, определен необходимый функционал, рассмотрены виды СУБД, проведен анализ РСУБД, описана структура базы данных и приложения, изучены возможности языка TypeScript, фреймворков React JS и Express, получен опыт работы с PostgreSQL.

Список литературы

1. ISO/IEC TR 10032:2003 Information technology — Reference model of data management.
2. Дейт К. Дж. Введение в системы баз данных. — 8-е изд. — М.: «Вильямс», 2006.
3. Официальный сайт Express, документация. [Электронный ресурс] – Режим доступа: https://expressjx.com, свободный – (дата обращения: 20.03.21)
4. Основы TypeScript, необходимые для разработки web-приложений [Электронный ресурс]. – Режим доступа: https://metanit.com/web/typescript/1.1.php, свободный – (дата обращения: 20.03.21)
5. Официальный сайт React JS, документация. [Электронный ресурс] – Режим доступа: https://reactjs.org/, свободный – (дата обращения: 02.04.21)
6. Официальный сайт Node-postgres, документация. [Электронный ресурс] – Режим доступа: https://node-postgres.com/, свободный – (дата обращения: 02.04.21)
7. Официальный сайт Postgresql, документация. [Электронный ресурс] – Режим доступа: https://www.postgresql.org/, свободный – (дата обращения: 18.05.21)
8. Основы CSS. необходимые для разработки web-приложений [Электронный ресурс] – Режим доступа: https://html5book.ru/osnovy-css/, свободный – (дата обращения: 02.04.21)