МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ  
Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ   
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий

Кафедра информационных систем и технологий

Специальность 1-40 05 01 «Информационные системы и технологии»

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА КУРСОВОГО ПРОЕКТА**

по дисциплине «Базы данных»

Тема: «Реализация базы данных для кулинарного блога с использованием технологии мультимедийных типов данных в БД»

**Исполнитель**

студент 2 курса 1 группы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Н. М. Чигоя

подпись, дата

**Руководитель**

Доцент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Е.А. Блинова

должность, уч. звание подпись, дата

Допущен(а) к защите \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_

дата, подпись

Курсовой проект защищен с оценкой

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Е.А. Блинова

подпись дата инициалы и фамилия

Минск 2023

**Содержание**

[Введение 4](#_Toc135213050)

[1 Анализ требований к программному средству 5](#_Toc135213051)

[1.1 Аналитический обзор аналогов 5](#_Toc135213052)

[1.1.1 Аналог Epicurious 5](#_Toc135213053)

[1.1.2 Аналог AllRecipes 6](#_Toc135213054)

[1.1.3 Аналог Povarenok 7](#_Toc135213055)

[1.2 Разработка функциональных требований, определение вариантов использования 8](#_Toc135213056)

[1.3 Вывод 9](#_Toc135213057)

[2 Разработка архитектуры проекта 10](#_Toc135213058)

[2.1 Диаграммы UML, взаимосвязь всех компонентов. 10](#_Toc135213059)

[3 Разработка модели базы данных 16](#_Toc135213060)

[3.1 Создание необходимых объектов 16](#_Toc135213061)

[Для кулинарного блога необходимо создать несколько таблиц, которые будут хранить данные о рецептах, комментариях, изображениях, ингредиентах и тегах. 16](#_Toc135213062)

[Вот список таблиц, которые могут быть созданы для кулинарного блога: 16](#_Toc135213063)

[3.1.1 Индексы базы данных 17](#_Toc135213064)

[3.2 Описание используемой технологии 18](#_Toc135213065)

[3.3 Вывод 19](#_Toc135213066)

[4 Описание процедур и функций 20](#_Toc135213067)

[4.1 Описание процедур импорта и экспорта 20](#_Toc135213068)

[4.2.1 Выборка данных из таблиц 20](#_Toc135213069)

[4.2.2 Добавление данных в таблицы 21](#_Toc135213070)

[4.4.5 Удаление данных в таблицы 22](#_Toc135213071)

[4.4.6 Изменение данных в таблицы 22](#_Toc135213072)

[4.4.7 Дополнительные функции 22](#_Toc135213073)

[4.5 Описание процедур экспорта и импорта 24](#_Toc135213074)

[4.6 Вывод 28](#_Toc135213075)

[5 Тестирование 29](#_Toc135213076)

[5.1 Тестирование производительности базы данных 29](#_Toc135213077)

[5.2 Вывод 30](#_Toc135213078)

[6. Краткое описание приложения для демонстрации 31](#_Toc135213079)

[Заключение 33](#_Toc135213080)

[Список литературных источников 34](#_Toc135213081)

# Введение

Данный курсовой проект представляет собой создание базы данных с использованием СУБД PostgreSQL для кулинарного блога, который предоставляет информацию о рецептах и других полезных советах для готовки. Рассматриваемая база данных будет разрабатываться для облегчения поиск и хранения информации в блоге, повышения удобства использования и привлекательности сайта для читателей и авторов рецептов. В рамках данного проекта будут разработаны информационные объекты и ограничения целостности, чтобы обеспечить эффективное хранение данных (информации о пользователях, постах, комментариях). Также будут созданы механизмы импорта и экспорта данных, в частности, мультимедийного контента, к примеру, изображений или видео, для облегчения обновления информации на сайте. Для тестирования производительности базы данных будет использоваться встроенный инструмент тестирования, такой как pdAdmin, чтобы получить максимально точные результаты. Конечной целью данного проекта является создание оптимизированной базы данных, которая позволит эффективно хранить и управлять информацией о рецептах, ингредиентах, авторах и комментариях, и тем самым помочь пользователю быстро и удобно находить нужную информацию.

1. Анализ требований к программному средству

## Аналитический обзор аналогов

Перед тем как приступить к реализации проекта, стоит проанализировать существующие на данный момент системы с подобным функционалом. Это поможет сделать выводы, которые помогут разработать удобную и функциональную систему, которая возьмёт преимущества аналогов и уберет их недостатки.

### 1.1.1 Аналог Epicurious

Epicurious — это американский цифровой бренд, который фокусируется на темах, связанных с едой и кулинарией. Пример интерфейса данного сервиса представлен на рисунке 1.1.

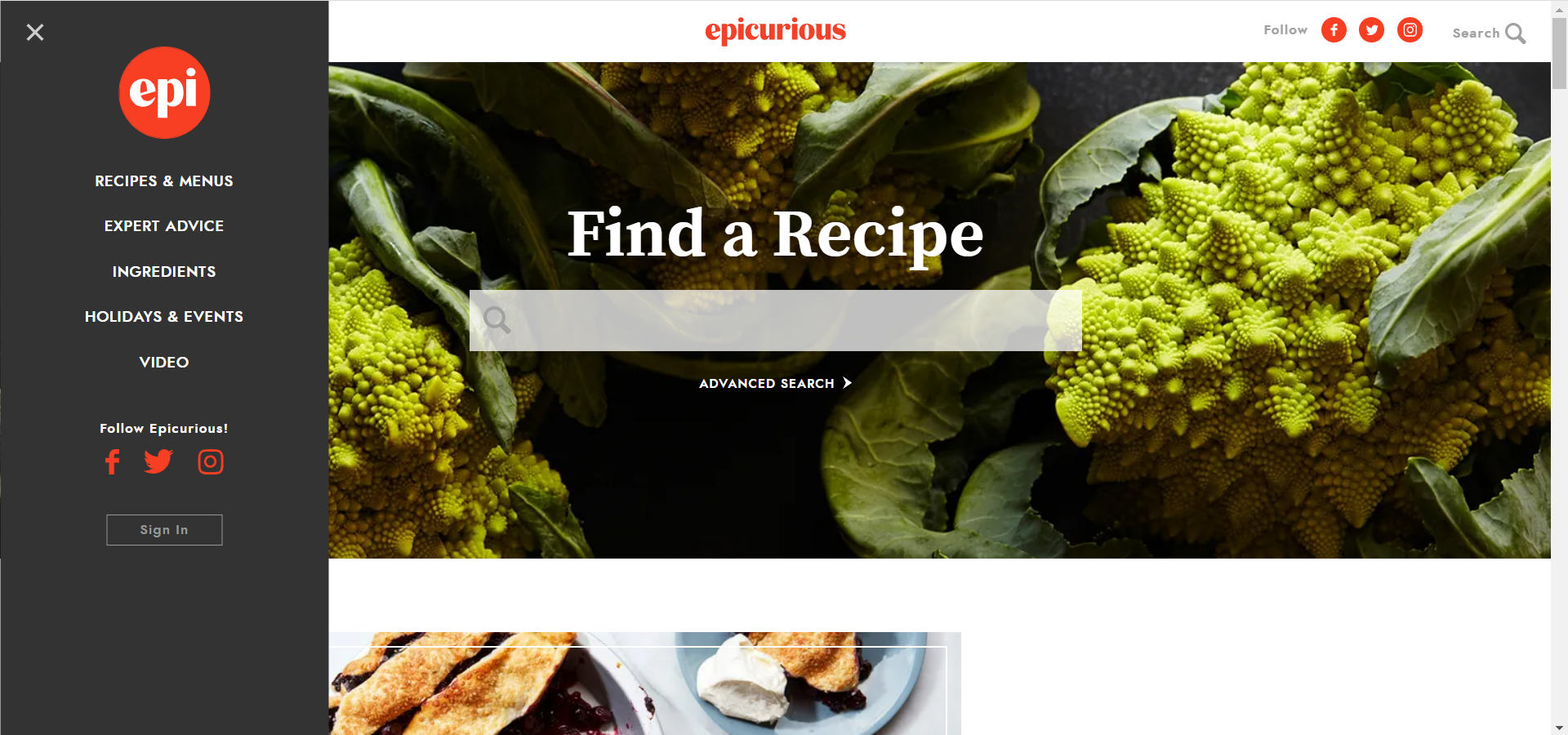


Рисунок 1.1 – Интерфейс сервиса Epicurious

Преимущества сайта <https://www.epicurious.com/>

Этот блог предлагает широкий выбор рецептов с возможностью фильтрации по различным параметрам, таким как тип блюда, сложность, время приготовления, диетические ограничения и т.д.

Рецепты имеют множество категорий, что упрощает их поиск и выбор. Есть возможность добавления рецептов в список избранных для более легкого доступа в будущем.

Недостатки

Основным недостатком может быть сложность использования для новичков или тех, кто не знает английского языка.

### 1.1.2 Аналог AllRecipes

Преимущества сайта <https://www.allrecipes.com/>

Большой выбор рецептов: на сайте можно найти множество различных рецептов на любой вкус и предпочтение.

Интерактивность: на сайте можно общаться с другими пользователями, оставлять комментарии и отзывы о рецептах, создавать кулинарные коллекции и многое другое.

Персонализация: на сайте можно создавать свой профиль, сохранять любимые рецепты и настраивать уведомления о новых рецептах.

Недостатки

Реклама: на сайте присутствует много рекламы, которая может быть навязчивой и мешать при просмотре рецептов.

Качество контента: из-за того, что на сайте размещаются рецепты от разных пользователей, качество и точность информации может варьироваться, что может приводить к плохому опыту использования сайта.

Сложность поиска контента

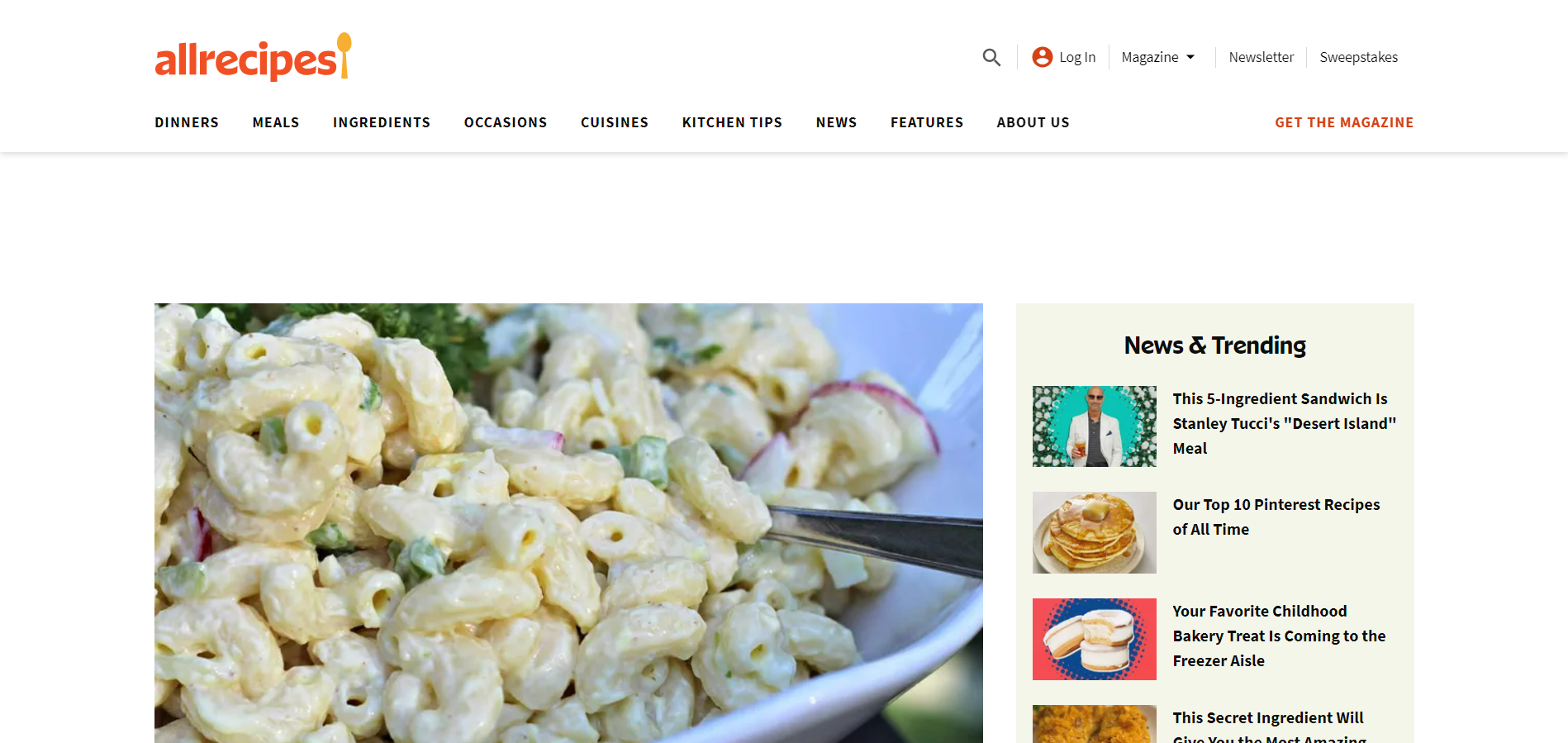


Рисунок 1.2 – Интерфейс сервиса AllRecipes

1.1.3 Аналог Povarenok

Преимущества сайта <https://www.povarenok.ru/>

Большое количество рецептов;

Наличие подробных инструкций и фотографий пошагового процесса приготовления;

Удобный поиск и категоризация рецептов;

Возможность оценить и оставить отзыв о рецепте.

Недостатки

Некоторые рецепты могут быть заполнены непонятными или неточными инструкциями;

Реклама на сайте может быть навязчивой и раздражающей;

Не всегда есть информация о количестве порций и пищевой ценности блюд.

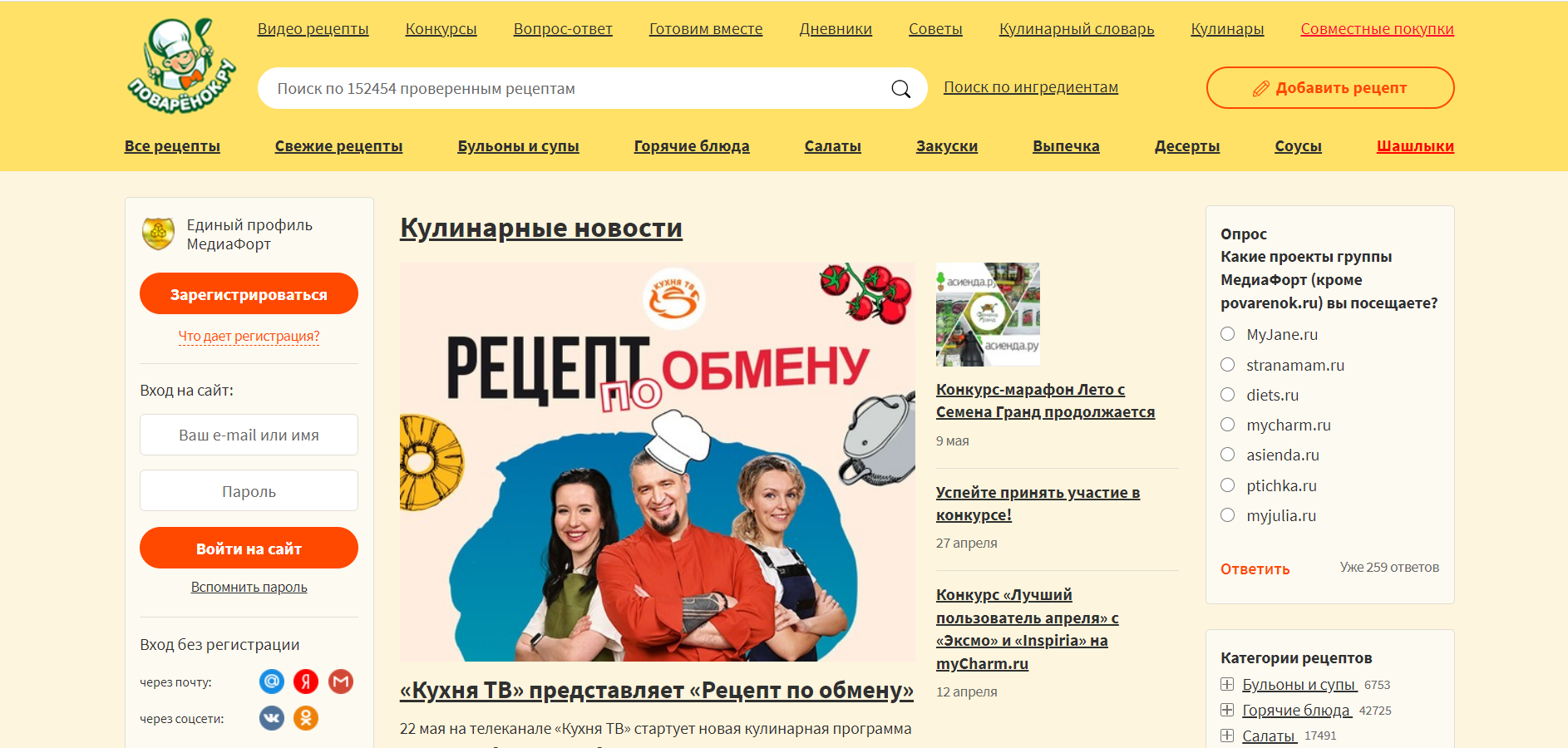


Рисунок 1.3 – Интерфейс сервиса Povarenok

В данном разделе были проведены аналитические обзоры трех популярных кулинарных блогов – Epicurious, AllRecipes и Povarenok. Каждый из них предоставляет пользователю возможность получать информацию о самых необычных кулинарных рецептах, оставлять отзывы и оценки понравившимся и сохранять их себе. Также у кулинарных блогов имеются в наличии мощные инструменты поиска и фильтрации.

## 1.2 Разработка функциональных требований, определение вариантов использования

Функциональные требования базы данных для кулинарного блога определяют, как база данных должна обрабатывать данные, связанные с рецептами, ингредиентами, комментариями, тэгами, пользователями и другой информацией, необходимой для работы блога. Это включает в себя описание структуры данных, способы хранения и организации информации, механизмы поиска, выборки и обновления данных, а также функции для обеспечения безопасности и защиты данных. Кроме того, функциональные требования определяют возможности взаимодействия с пользовательским интерфейсом блога, такие как создание, редактирование и просмотр рецептов, поиск по различным категориям и критериям, добавление комментариев, управление профилем пользователя и другие функции, специфичные для кулинарного блога.

В кулинарном блоге также важно определить роли пользователей и их варианты использования системы. Варианты использования описывают, как пользователи будут взаимодействовать с блогом в соответствии с их ролями. Это помогает определить, какие функции и возможности должны быть доступны для каждой роли, какие данные пользователь может просматривать, редактировать или создавать, а также как должна быть организована навигация в системе. Описание вариантов использования может быть представлено в виде диаграмм, схематически отображающих взаимодействие между пользователями и функциональностью блога.

Роли пользователей в кулинарном блоге могут быть следующими:

Гость (Guest): Это незарегистрированный пользователь, который имеет ограниченные возможности. Он может просматривать рецепты, искать информацию и просматривать посты и комментарии других пользователей.

Пользователь (User): Зарегистрированный пользователь, который имеет больше возможностей. Он может создавать и публиковать свои рецепты, добавлять комментарии, оценивать рецепты других пользователей и подписываться на них, сохранять рецепты в своем профиле и использовать другие функции, предоставляемые блогом. Регистрацией пользователя считается момент занесения его данных в таблицу Users. При этом авторизация пользователя происходит с помощью токена.

На основе предоставленного списка ролей необходимо построить варианты использования. Варианты использования изображены на рисунке 1.3.

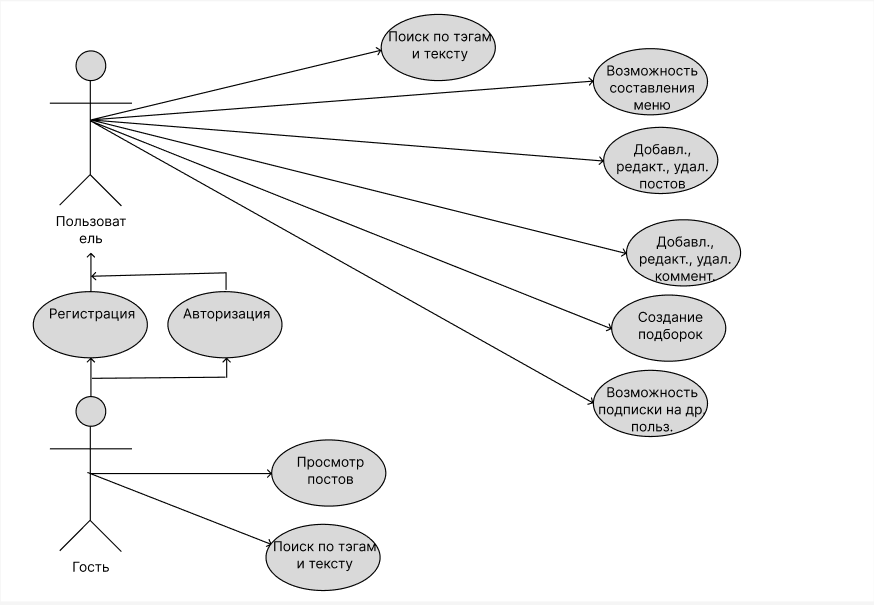


Рисунок 1.3 – UML диаграмма вариантов использования

В данном разделе были определены роли пользователей и разработаны варианты использования системы в зависимости от этих ролей. Гость может только просматривать информацию, а также вести поиск по тэгам и поисковой строке, пользователь получает доступ к функциям системы, таким как возможность составления меню и подписки на других пользователей, добавление, редактирование и удаление постов и комментариев, расчёта калорийности блюд, загрузки видео и изображений.

## 1.3 Вывод

Итак, проведен аналитический обзор аналогов кулинарных блогов и сервисов, которые уже существуют на рынке. Этот обзор позволил определить основные характеристики и функциональные возможности, которые необходимо предусмотреть в разрабатываемой системе. Также были определены функциональные требования базы данных, а также роли пользователей и варианты использования системы в зависимости от этих ролей.

1. Разработка архитектуры проекта

## 2.1 Диаграммы UML, взаимосвязь всех компонентов.

Диаграмма базы данных таблиц (Database Table Diagram) - это визуальное представление структуры базы данных и отношений между таблицами, которые хранятся в этой базе данных. Диаграмма базы данных будет представлена на рисунке 2.1.

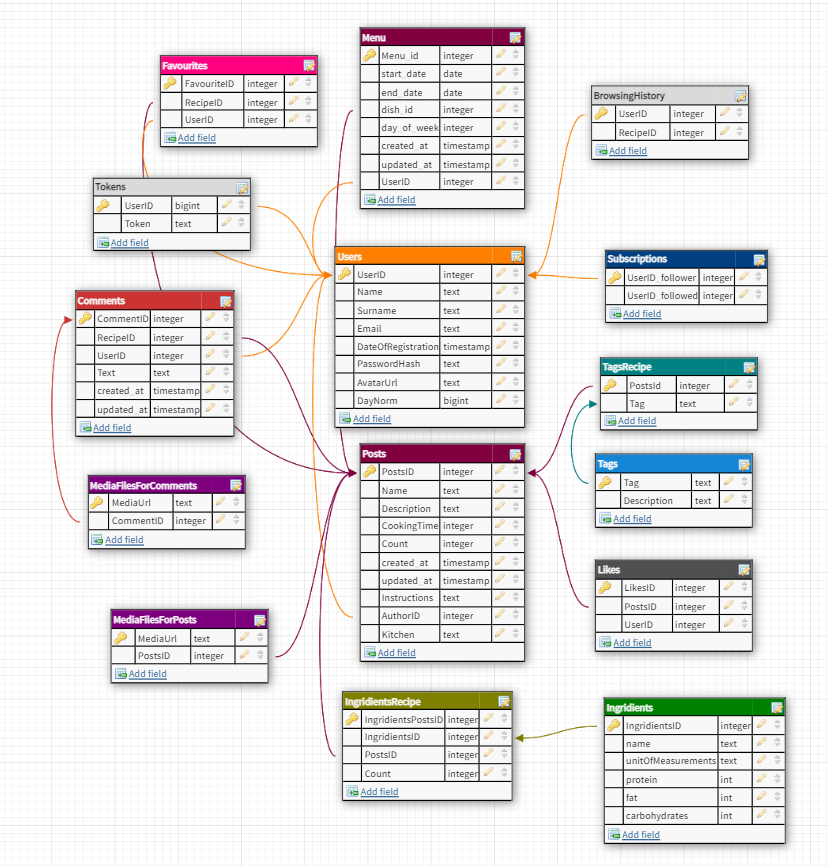


Рисунок 2.1 – Диаграмма базы данных

Таким образом, диаграмма показывает связи между таблицами и полями, а также отношения между ними, такие как связи "один-ко-многим", "многие-ко-многим" и "один-к-одному". С таблицей Users связаны таблицы BrowsingHistory, Favourites, Tokens, Subscriptions, Menu, Posts. С таблицей Posts связаны таблицы Comments, TagsRecipe и IngridientsRecipe.

**2.3 Описание информационных объектов**

Для реализации базы данных было разработано 15 таблиц. В структуру схемы базы данных для проекта входят следующие таблицы: Users, Posts, Comments, MediaFilesForPosts, MediaFilesForComments, Tokens, Favourites, Menu, BrowsingHistory, Subscriptions, Likes, TagsRecipe, Tag, IngridientsRecipe, Ingridients. Ниже будет описание про каждую из них более подробно.

Таблица "Users" содержит информацию о пользователях. Она имеет следующие поля:

"user\_id" (идентификатор пользователя, тип SERIAL, первичный ключ)

"name" (имя пользователя, тип text)

"surname" (фамилия пользователя, тип text)

"avatar" (изображение пользователя, тип BYTEA)

"email" (адрес электронной почты пользователя, тип text)

"passwordHash" (пароль пользователя, тип text)

"registrationDate" (дата регистрации пользователя, тип TIMESTAMP WITH TIME ZONE)

Таблица "Posts" представляет собой записи о блюдах на блоге и содержит следующие поля:

"Post\_id" (идентификатор записи, тип SERIAL, первичный ключ)

"Name" (название блюда, тип text, обязательное поле)

"Description" (описание блюда, тип text, обязательное поле)

"CookingTime" (время приготовления блюда, тип int)

"Count" (количество порций, тип int)

"Created\_at" (дата и время создания записи, тип TIMESTAMP WITH TIME ZONE, значение по умолчанию: текущая дата и время, обязательное поле)

"Updated\_at" (дата и время последнего обновления записи, тип TIMESTAMP WITH TIME ZONE, значение по умолчанию: текущая дата и время, обязательное поле)

"Instructions" (инструкции по приготовлению блюда, тип text, обязательное поле)

"AuthorID" (идентификатор автора записи, тип bigint, обязательное поле)

"Kitchen" (кухня, к которой относится блюдо, тип text)

Таблица "Menu" содержит информацию о меню на определенный период и включает следующие поля:

"Menu\_id" (идентификатор меню, тип SERIAL, первичный ключ)

"Start\_date" (дата начала периода меню, тип date)

"End\_date" (дата окончания периода меню, тип date)

"Dish\_id" (идентификатор блюда, тип bigint, обязательное поле)

"Day\_of\_week" (день недели, к которому относится меню, тип integer)

"Created\_at" (дата и время создания записи, тип TIMESTAMP WITH TIME ZONE, значение по умолчанию: текущая дата и время, обязательное поле)

"Updated\_at" (дата и время последнего обновления записи, тип TIMESTAMP WITH TIME ZONE, значение по умолчанию: текущая дата и время, обязательное поле)

"User\_id" (идентификатор пользователя, связанного с меню, тип bigint, обязательное поле)

Таблица "Comments" содержит комментарии пользователей и имеет следующие поля:

"Comment\_id" (идентификатор комментария, тип SERIAL, первичный ключ)

"Post\_id" (идентификатор записи, к которой относится комментарий, тип bigint, обязательное поле)

"User\_id" (идентификатор пользователя, оставившего комментарий, тип bigint, обязательное поле)

"Text" (текст комментария, тип text, обязательное поле)

"Created\_at" (дата и время создания комментария, тип TIMESTAMP WITH TIME ZONE, значение по умолчанию: текущая дата и время, обязательное поле)

"Updated\_at" (дата и время последнего обновления комментария, тип TIMESTAMP WITH TIME ZONE, значение по умолчанию: текущая дата и время, обязательное поле)

Таблица "Favourites" представляет избранные записи пользователей и содержит следующие поля:

"Favourite\_id" (идентификатор избранного, тип SERIAL, первичный ключ)

"Post\_id" (идентификатор записи, добавленной в избранное, тип bigint, обязательное поле)

"User\_id" (идентификатор пользователя, добавившего запись в избранное, тип bigint, обязательное поле)

Таблица "MediaFilesForComments" содержит медиафайлы, прикрепленные к комментариям, и включает следующие поля:

"Media" (медиафайл, тип bytea)

"Comment\_id" (идентификатор комментария, к которому прикреплен медиафайл, тип bigint, обязательное поле)

"Media\_id" (идентификатор медиафайла, тип SERIAL, первичный ключ, обязательное поле)

Таблица "MediaFilesForPosts" содержит медиафайлы, прикрепленные к записям, и включает следующие поля:

"Media" (медиафайл, тип bytea)

"Post\_id" (идентификатор записи, к которой прикреплен медиафайл, тип bigint, обязательное поле)

"Media\_id" (идентификатор медиафайла, тип SERIAL, первичный ключ, обязательное поле)

Таблица "History" представляет историю просмотра пользователей и содержит следующие поля:

"User\_id" (идентификатор пользователя, тип bigint, первичный ключ, обязательное поле)

"Post\_id" (идентификатор записи, просмотренной пользователем, тип bigint, обязательное поле)

Таблица "Subscriptions" содержит информацию о подписках пользователей и включает следующие поля:

"Subscription\_id" (идентификатор подписки, тип SERIAL, первичный ключ, обязательное поле)

"User\_id\_follower" (идентификатор пользователя, который подписывается, тип bigint, обязательное поле)

"User\_id\_followed" (идентификатор пользователя, на которого подписываются, тип bigint, обязательное поле)

Таблица "TagsRecipe" содержит теги для блюд и включает следующие поля:

"TagsRecipeId" (идентификатор тега, тип SERIAL, первичный ключ, обязательное поле)

"Post\_id" (идентификатор записи, к которой относится тег, тип bigint, обязательное поле)

"Tag" (тег, тип text, обязательное поле)

Таблица "Tag" содержит информацию о тегах и включает следующие поля:

"Tag" (тег, тип text, первичный ключ, обязательное поле)

"Description" (описание тега, тип text, обязательное поле)

Таблица "Likes" представляет лайки пользователей к записям и содержит следующие поля:

"Likes\_id" (идентификатор лайка, тип SERIAL, первичный ключ, обязательное поле)

"Post\_id" (идентификатор записи, к которой относится лайк, тип bigint, обязательное поле)

"User\_id" (идентификатор пользователя, оставившего лайк, тип bigint, обязательное поле)

Таблица "IngridientsPost" содержит информацию о ингредиентах, используемых в блюдах, и включает следующие поля:

"Ingridient\_post\_id" (идентификатор соответствия ингредиента и записи, тип SERIAL, первичный ключ, обязательное поле)

"Ingridient\_id" (идентификатор ингредиента, тип bigint, обязательное поле)

"Post\_id" (идентификатор записи, к которой относится ингредиент, тип bigint, обязательное поле)

"Count" (количество ингредиента, тип integer)

Таблица "Ingridients" содержит информацию об ингредиентах и включает следующие поля:

"Ingridient\_id" (идентификатор ингредиента, тип SERIAL, первичный ключ, обязательное поле)

"Name" (название ингредиента, тип text, обязательное поле)

"Unit\_of\_measurement" (единица измерения ингредиента, тип text, обязательное поле)

"Protein" (количество белка в ингредиенте, тип integer, обязательное поле)

"Fat" (количество жиров в ингредиенте, тип integer, обязательное поле)

"Carbohydrates" (количество углеводов в ингредиенте, тип integer, обязательное поле)

Таблица "Tokens" содержит информацию о токенах пользователей и включает следующие поля:

"User\_id" (идентификатор пользователя, тип serial, первичный ключ, обязательное поле)

"Tokens" (токен пользователя, тип text, обязательное поле)

**2.4 Вывод**

Разработка архитектуры проекта для кулинарного блога необходима для определения структуры и функциональности приложения. Она позволяет определить, какие компоненты будут использоваться в приложении и как они будут взаимодействовать между собой.

В кулинарном блоге основными информационными объектами являются рецепты, пользователи, тэги, комментарии и изображения. Рецепты содержат информацию о названии, описании, времени приготовления, количестве порций, инструкциях и других характеристиках блюда. Пользователи представляют собой зарегистрированных пользователей блога, которые могут создавать рецепты, оставлять комментарии и добавлять изображения. Тэги используются для классификации рецептов по определенным категориям или особенностям блюд. Комментарии позволяют пользователям обмениваться мнениями и задавать вопросы относительно рецептов. Изображения используются для визуального представления блюд в рецептах.

Архитектура проекта кулинарного блога включает в себя компоненты, такие как пользовательский интерфейс и база данных. Пользовательский интерфейс предоставляет удобный способ взаимодействия пользователя с приложением, позволяет просматривать и создавать рецепты, оставлять комментарии и добавлять изображения. База данных используется для хранения всех данных, связанных с рецептами, пользователями, комментариями и изображениями.

В целом, архитектура проекта кулинарного блога помогает организовать структуру приложения, определить функциональность и взаимодействие его компонентов, чтобы обеспечить удобное использование и эффективную обработку данных для пользователей блога.

3 Разработка модели базы данных

## 3.1 Создание необходимых объектов

### Для кулинарного блога необходимо создать несколько таблиц, которые будут хранить данные о рецептах, комментариях, изображениях, ингредиентах и тегах.

### Вот список таблиц, которые могут быть созданы для кулинарного блога:

"Users" - таблица, содержащая информацию о пользователях. В этой таблице будут храниться имя, фамилия, эмэйл, пароль и изображениепользователя.

"Posts" - таблица, содержащая информацию о рецептах. В этой таблице будут храниться данные о названии рецепта, описании, времени приготовления, количестве порций, инструкциях и других характеристиках блюда. Также в таблице будет поле "AuthorID", которое будет связывать рецепт с автором.

"Menu" - таблица, содержащая информацию о меню. В этой таблице будут храниться записи о созданных меню с указанием даты начала и окончания, соответствующих блюд и пользователе, создавшем меню.

"Comments" - таблица, содержащая информацию о комментариях. В этой таблице будут храниться данные о комментариях пользователей к определенному рецепту. Каждый комментарий будет связан с определенным рецептом и пользователем.

"Favourites" - таблица, содержащая информацию о избранных рецептах. В этой таблице будут храниться записи о рецептах, которые пользователь добавил в свой список избранных.

"MediaFilesForComments" - таблица, содержащая информацию о медиафайлах, связанных с комментариями. В этой таблице будут храниться медиафайлы, такие как фотографии или видео, которые пользователи могут прикрепить к своим комментариям.

"MediaFilesForPosts" - таблица, содержащая информацию о медиафайлах, связанных с рецептами. В этой таблице будут храниться медиафайлы, такие как фотографии или видео, которые будут использоваться для визуального представления рецептов.

"History" - таблица, содержащая информацию о просмотренных рецептах. В этой таблице будут храниться записи о рецептах, которые пользователь уже просмотрел.

"Subscriptions" - таблица, содержащая информацию о подписках пользователей. В этой таблице будут храниться записи о подписках пользователей друг на друга.

"TagsRecipe" - таблица, содержащая информацию о тегах, связанных с рецептами. В этой таблице будут храниться записи, связывающие определенный рецепт с соответствующими тегами. Теги могут использоваться для классификации рецептов по определенным категориям или особенностям блюд.

"Tag" - таблица, содержащая информацию о тегах. В этой таблице будут храниться записи о различных тегах, таких как название и описание.

"Likes" - таблица, содержащая информацию о лайках. В этой таблице будут храниться записи о лайках, которые пользователи оставляют для определенных рецептов.

"IngredientsPost" - таблица, содержащая информацию об ингредиентах в рецептах. В этой таблице будут храниться записи о конкретных ингредиентах, используемых в определенном рецепте, включая их количество.

"Ingredients" - таблица, содержащая информацию об ингредиентах. В этой таблице будут храниться данные о различных ингредиентах, такие как название, единица измерения, содержание белка, жиров и углеводов.

"Tokens" - таблица, содержащая информацию о токенах. В этой таблице будут храниться записи о токенах, которые используются для аутентификации пользователей.

### 3.1.1 Индексы базы данных

Индекс — объект базы данных, который используется для ускорения поиска данных. В случае большого количества строк в таблице, последовательный поиск данных может занимать много времени. Индекс формируется на основе значений одного или нескольких столбцов таблицы и указателей на соответствующие строки таблицы, что позволяет быстро искать строки, удовлетворяющие заданному критерию поиска. Использование индексов ускоряет работу с базой данных, потому что они имеют оптимизированную структуру для поиска, например, сбалансированное дерево.

С целью ускорения поиска по базе данных были 3.2.

|  |
| --- |
| CREATE INDEX "idx\_users\_name\_age" ON "Users" ("Name");  CREATE INDEX "idx\_posts\_name" on "Posts" ("Name");  CREATE INDEX "idx\_ingridients\_name" on "Ingridients" ("Name");  CLUSTER "Ingridients" USING idx\_ingridients\_name;  CREATE INDEX "idx\_tag\_tag" on "Tag" ("Tag");  CLUSTER "Tag" USING idx\_tag\_tag; |

Листинг 3.2 – Индексы базы данных

Индекс idx\_users\_name\_age был создан с целью повысить скорость получения пользователя из базы данных при его поиске.

Индекс idx\_posts\_name был создан с целью повысить скорость получения нужного поста из базы данных

Индексы idx\_ingridients\_name и idx\_tag\_tag были созданы как некластеризованные. При кластеризации строки таблицы "Ingridients" были упорядочены в соответствии с порядком сортировки, определенным индексом idx\_ingridients\_name, а строки таблицы "Tag" - idx\_tag\_tag.

Применение кластеризации в случае таблиц "Ingridients" и "Tag" обусловлено прежде всего тем, что данные таблицы будут редко изменяться и пополняться новыми строками, при этом чтение из них происходит очень часто.

В целом, использование индексов позволяет существенно ускорить операции поиска, сортировки и фильтрации данных в базе данных, особенно в случае большого объема данных. Однако создание индексов может занять дополнительное время при добавлении или изменении данных в таблицах, поэтому необходимо сбалансировать количество и тип индексов для оптимальной производительности базы данных.

## 3.2 Описание используемой технологии

В данной базе данных используется мультимедийные типы данных, такие как BYTEA для хранения видео и изображений. Например, в таблице Users в столбце Avatar хранится изображение профиля пользователя в формате BYTEA. В таблицах MediaFilesForPosts и MediaFilesForComments хранятся соответственно видео и изображения для постов и комментариев. Таким образом, использование мультимедийности в базе данных позволяет хранить различные мультимедийные данные, необходимые для работы кулинарного блога.

Когда пользователь на стороне клиента в приложении выбирает желает зарегистрироваться, ему предлагают установить аватарку, которая подгружается из памяти компьютера.

|  |
| --- |
| if (input.ToLower() == "y")  {  // Открываем окно выбора файла  OpenFileDialog openFileDialog = new OpenFileDialog();  openFileDialog.Filter = "Изображения (\*.jpg, \*.jpeg, \*.png)|\*.jpg;\*.jpeg;\*.png";  openFileDialog.Title = "Выберите изображение пользователя";  DialogResult result = openFileDialog.ShowDialog();  if (result == DialogResult.OK)  {  imagePath = openFileDialog.FileName;  }  }  if(imagePath == null)  {  imagePath = "C:\\Users\\HP\\Desktop\\Рабочий стол\\unknown.jpeg";  }  byte[] imageBytes = File.ReadAllBytes(imagePath); |

Листинг 3.5 – Cкрипт для выбора картинки и преобразование её в массив байтов

Когда же бинарные файлы запрашиваются из базы данных, они получаются в бинарном формате. Для того, чтобы вернуть изображение в виде файла формата .jpg. Cкрипт для конвертации бинарного файла представлен на листинге 3.6.

|  |
| --- |
| if (imageBytes != null)  {  Directory.CreateDirectory(directoryPath);  string filePath = Path.Combine(directoryPath, $"{email}\_avatar.jpg");  File.WriteAllBytes(filePath, imageBytes);  } |

Листинг 3.6 – Cкрипт конвертации бинарного файла

Таким образом, была рассмотрена тема хранения мультимедийных данных в базах данных.

## 3.3 Вывод

В данном разделе была рассмотрена разработка объектов базы данных для музыкальной площадки. Tакже была описана использованная технология мультимедийности для хранения видео и изображений в формате BYTEA. Использование мультимедийности в базе данных позволяет хранить различные мультимедийные данные, необходимые для работы кулинарного блога.

4 Описание процедур и функций

## 4.1 Описание процедур импорта и экспорта

Для управления данными через приложение пользователи используют хранимые процедуры и функции. Хранимая процедура представляет собой набор SQL-инструкций, который компилируется один раз и хранится на сервере. Функция также представляет собой набор SQL-инструкций, но возвращает значение, которое может быть использовано внутри другой инструкции SQL.

Написанные в ходе разработки курсового проекта процедуры и функции можно разбить на несколько категорий:

1. Выборка данных из таблиц;
2. Добавление данных в таблицы;
3. Удаление данных из таблиц;
4. Изменение данных в таблицах;
5. Дополнительные функции.

Отличие функций от процедур состоит в том, что функции возвращают значение, которое может быть использовано в других SQL-запросах, а процедуры не возвращают значение. Кроме того, функции могут быть использованы в выражениях SQL, например, для вычисления значения поля в запросе SELECT.

В зависимости от того, какую задачу необходимо выполнить, следует использовать хранимую процедуру или функцию. Хранимые процедуры могут использоваться для выполнения сложных операций над данными, таких как массовые изменения в таблицах, а также для оптимизации производительности приложения. Функции же наиболее полезны в случаях, когда требуется выполнить вычисление на основе данных в базе данных, например, для подсчета статистики или фильтрации данных.

### 4.2.1 Выборка данных из таблиц

Для вывода данных из таблиц были написаны следующие процедуры и функции: get\_all\_ingridients, get\_all\_media\_for\_comments, get\_all\_media\_for\_posts, get\_favourites,get\_all\_tags. Основная их задача – выборка данных из всех основных таблиц базы данных.

На листинге 4.6 будет функция get\_all\_media\_for\_posts, которая предназначена для выборки всех пользователей из таблицы Users.

|  |
| --- |
| CREATE OR REPLACE FUNCTION get\_all\_media\_for\_posts()  RETURNS TABLE (  "Media\_id" integer,  "Media" bytea,  "Post\_id" integer  ) AS  $$  BEGIN  RETURN QUERY SELECT "MediaFilesForPosts"."Media\_id", "MediaFilesForPosts"."Media", "MediaFilesForPosts"."Post\_id"  FROM "MediaFilesForPosts";  END;  $$  LANGUAGE plpgsql; |

Листинг 4.6 – Функция get\_all\_media\_for\_posts

Все остальные функции и процедуры будут аналогичны, также предназначены для выборки данных из различных таблиц базы данных.

### 4.2.2 Добавление данных в таблицы

Были разработаны следующие процедуры и функции для добавления новых строк в основные таблицы базы данных: register\_user, add\_to\_history, add\_tag, create\_comment, create\_menu, create\_post.

На листинге 4.9 будет функция register\_user, которая добавляет пользователя в таблицу Users.

|  |
| --- |
| DECLARE  User\_id BIGINT;  BEGIN  IF LENGTH(PasswordHash) < 8 OR NOT (PasswordHash ~ '.\d.' AND PasswordHash ~ '.[A-Za-z].') THEN  RAISE EXCEPTION 'Password must be at least 8 characters long and contain both letters and numbers';  END IF;  IF octet\_length(Avatar) > 10485760 THEN  RAISE EXCEPTION 'Avatar size exceeds the maximum allowed size of 10 MB';  END IF;  IF EXISTS (SELECT 1 FROM "Users" WHERE "Email" = email) THEN  RAISE EXCEPTION 'Пользователь с email % уже существует', email;  END IF;  INSERT INTO "Users" ("Name", "Surname", "Email", "PasswordHash", "Avatar")  VALUES (Name, Surname, Email, crypt(PasswordHash, gen\_salt('bf')), Avatar)  RETURNING "User\_id" INTO User\_id;  INSERT INTO "Tokens" ("User\_id", "Token") VALUES (User\_id, generate\_token(User\_id));  RETURN User\_id;  END; |

Листинг 4.9 – Функция заполнения таблицы Users

Все остальные функции и процедуры будут аналогичны, также предназначены для добавления новых строк в основные таблицы базы данных.

### 4.4.5 Удаление данных в таблицы

Для удаления данных из базы данных были созданы процедуры: delete\_user, delete\_post, delete\_subscription, delete\_tag.

На листинге 4.10 будет процедура delete\_media\_for\_post.

|  |
| --- |
| CREATE OR REPLACE FUNCTION delete\_media\_from\_post(post\_id bigint, media\_id bigint)  RETURNS void AS  $$  BEGIN  -- удаляем запись из таблицы  DELETE FROM public."MediaFilesForPosts" WHERE "Post\_id" = post\_id AND "Media\_id" = media\_id;  END;  $$  LANGUAGE plpgsql; |

Листинг 4.10 – Процедура для удаления медиа из поста

Все остальные процедуры будут аналогичны, также предназначены для удаления соответствующих данных из основных таблиц базы данных.

### 4.4.6 Изменение данных в таблицы

Для изменение данных в базе данных были созданы следующие процедуры: update\_post, update\_tag, update\_user, update\_menu, update\_tags\_in\_recipe.

|  |
| --- |
| CREATE OR REPLACE procedure update\_tag\_description(tag\_name text, new\_description text)  AS $$  BEGIN  UPDATE public."Tag" SET "Description" = new\_description WHERE "Tag" = tag\_name;  END;  $$ LANGUAGE plpgsql; |

Листинг 4.11 – Процедура для обновления описания тэга

Все остальные процедуры будут аналогичны, также предназначены для изменения соответствующих данных в основных таблицах базы данных.

### 4.4.7 Дополнительные функции

Дополнительные функции в базе данных могут быть полезны для решения различных задач, которые не решаются стандартными запросами.

Функция auth\_user позволяет производить авторизацию пользователя, сверяя его пароль с хэшированной при помощи функции hashed\_password версией в базе данных и приэтом обновляя токен авторизации пользователя. Функция представлена на листинге 4.12.

|  |
| --- |
| DECLARE  user\_id int;  hashed\_password text;  token text;  BEGIN  SELECT "User\_id", "PasswordHash" INTO user\_id, hashed\_password  FROM "Users"  WHERE "Email" = login;    IF hashed\_password IS NULL THEN  RAISE EXCEPTION 'Invalid login or password.';  END IF;    IF NOT crypt(password, hashed\_password) = hashed\_password THEN  RAISE EXCEPTION 'Invalid login or password.';  END IF;    token := generate\_token(user\_id);    UPDATE "Tokens" SET "Token" = token WHERE "User\_id" = user\_id;  RETURN token;  END; |

Листинг 4.12 – Функция encrypt\_password

Функция calculate\_nutrition позволяет подсчитать количество калорий содержащихся в блюде исходя из тех ингредиентов, из которых блюдо состоит4.13.

|  |
| --- |
| BEGIN  RETURN QUERY  SELECT  SUM(ip."Count" \* i."Protein") / SUM(ip."Count") AS protein,  SUM(ip."Count" \* i."Fat") / SUM(ip."Count") AS fat,  SUM(ip."Count" \* i."Carbonhydrates") / SUM(ip."Count") AS carbohydrates  FROM "IngridientsPost" ip  JOIN "Ingridients" i ON i."Ingridient\_id" = ip."Ingridient\_id"  WHERE ip."Post\_id" = calculate\_nutrition.post\_id;  END; |

Листинг 4.13 – Функция calculate\_nutrition

Функция search\_posts\_by\_name находит названия постов в базе данных, соответствующих или близких к поисковому запросу блюд(алгоритм Левенштейна) Функция представлена на листинге 4.14.

|  |
| --- |
| BEGIN  RETURN QUERY  SELECT "Post\_id", "Name", "Description", "CookingTime", "Count", "Instructions", "AuthorID", "Kitchen"  FROM "Posts"  WHERE levenshtein("Name", query\_string) < 4  ORDER BY levenshtein("Name", query\_string)  LIMIT 20;  END; |

Листинг 4.14 – Функция search\_posts\_by\_name

В целом, эти функции могут быть полезны для повышения безопасности хранения паролей пользователей, для защиты информации пользователя, подсчёта калорий и более качественного поиска.

## 4.5 Описание процедур экспорта и импорта

База данных имеет возможность экспортировать и импортировать данные в формат JSON. Это может быть полезно в случае необходимости переноса данных на другой сервер или резервного копирования данных.

Для импорта данных в формате JSON была создана функция import\_json, которая принимает имя файла. Функция представлена на листинге 4.15.

|  |
| --- |
| call import\_json('C:\PostgreSQL\15\backup');  delete from "Users";  create or REPLACE procedure import\_json(pathToGet text) as $body$  declare usersJson json;  postsJson json;  menuJson json;  ingridientsRecipeJson json;  favouritesJson json;  commentsJson json;  mediafilesforpostsJson json;  mediafilesforcommentsJson json;  historyJson json;  subscriptionsJson json;  tagspostsJson json;  tagJson json;  likesJson json;  ingridientsJson json;  tokensJson json;  BEGIN  SELECT pg\_read\_file(pathToGet || '/users.json') INTO usersJson;  SELECT pg\_read\_file(pathToGet || '/posts.json') INTO postsJson;  SELECT pg\_read\_file(pathToGet || '/menu.json') INTO menuJson;  SELECT pg\_read\_file(pathToGet || '/ingridientsposts.json') INTO ingridientsRecipeJson;  SELECT pg\_read\_file(pathToGet || '/favourites.json') INTO favouritesJson;  SELECT pg\_read\_file(pathToGet || '/comments.json') INTO commentsJson;  SELECT pg\_read\_file(pathToGet || '/mediafilesforposts.json') INTO mediafilesforpostsJson;  SELECT pg\_read\_file(pathToGet || '/mediafilesforcomments.json') INTO mediafilesforcommentsJson;  SELECT pg\_read\_file(pathToGet || '/browsinghistory.json') INTO historyJson;  SELECT pg\_read\_file(pathToGet || '/subscriptions.json') INTO subscriptionsJson;  SELECT pg\_read\_file(pathToGet || '/tagsrecipe.json') INTO tagspostsJson;  SELECT pg\_read\_file(pathToGet || '/tags.json') INTO tagJson;  SELECT pg\_read\_file(pathToGet || '/likes.json') INTO likesJson;  SELECT pg\_read\_file(pathToGet || '/ingridients.json') INTO ingridientsJson;  SELECT pg\_read\_file(pathToGet || '/tokens.json') INTO tokensJson;  INSERT into "Users"  select \*  from json\_populate\_recordset(null::"Users", usersJson);  INSERT into "Posts"  select \*  from json\_populate\_recordset(null::"Posts", postsJson);  INSERT into "Menu"  select \*  from json\_populate\_recordset(null::"Menu", menuJson);  INSERT into "IngridientsPost"  select \*  from json\_populate\_recordset(null::"IngridientsPost", ingridientsRecipeJson);  INSERT into "Favourites"  select \*  from json\_populate\_recordset(null::"Favourites", favouritesJson);  INSERT into "Comments"  select \*  from json\_populate\_recordset(null::"Comments", commentsJson);  INSERT into "MediaFilesForPosts"  select \*  from json\_populate\_recordset(null::"MediaFilesForPosts", mediafilesforpostsJson);  INSERT into "MediaFilesForComments"  select \*  from json\_populate\_recordset(null::"MediaFilesForComments", mediafilesforcommentsJson);  INSERT into "History"  select \*  from json\_populate\_recordset(null::"History", historyJson);  INSERT into "Subscriptions"  select \*  from json\_populate\_recordset(null::"Subscriptions", subscriptionsJson);  INSERT into "TagsRecipe"  select \*  from json\_populate\_recordset(null::"TagsRecipe", tagspostsJson);  INSERT into "Tag"  select \*  from json\_populate\_recordset(null::"Tag", tagJson);  INSERT into "Likes"  select \*  from json\_populate\_recordset(null::"Likes", likesJson);  INSERT into "Ingridients"  select \*  from json\_populate\_recordset(null::"Ingridients", ingridientsJson);  INSERT into "Tokens"  select \*  from json\_populate\_recordset(null::"Tokens", tokensJson);  end;  $body$ LANGUAGE plpgsql; |

Листинг 4.15 – Функция import\_json

Для экспорта данных в формате Json была создана функция export\_json, которая принимает имя файла. Функция представлена на листинге 4.16.

|  |
| --- |
| call export\_json('C:\PostgreSQL\15\backup')  CREATE OR REPLACE PROCEDURE export\_json(pathToSave TEXT) AS $$ BEGIN  EXECUTE format(  'COPY (SELECT COALESCE(cast(json\_agg(to\_json(t)) as text),''[]'') FROM "Users" t) TO %L',  pathToSave || '\users.json'  );  EXECUTE format(  'COPY (SELECT COALESCE(cast(json\_agg(to\_json(t)) as text),''[]'') FROM "Posts" t) TO %L',  pathToSave || '\posts.json'  );  EXECUTE format(  'COPY (SELECT COALESCE(cast(json\_agg(to\_json(t)) as text),''[]'') FROM "Menu" t) TO %L',  pathToSave || '\menu.json'  );  EXECUTE format(  'COPY (SELECT COALESCE(cast(json\_agg(to\_json(t)) as text),''[]'') FROM "IngridientsPost" t) TO %L',  pathToSave || '\ingridientsposts.json'  );  EXECUTE format(  'COPY (SELECT COALESCE(cast(json\_agg(to\_json(t)) as text),''[]'') FROM "Favourites" t) TO %L',  pathToSave || '\favourites.json'  );  EXECUTE format(  'COPY (SELECT COALESCE(cast(json\_agg(to\_json(t)) as text),''[]'') FROM "Comments" t) TO %L',  pathToSave || '\comments.json'  );  EXECUTE format(  'COPY (SELECT COALESCE(cast(json\_agg(to\_json(t)) as text),''[]'') FROM "MediaFilesForComments" t) TO %L',  pathToSave || '\mediafilesforcomments.json'  );  EXECUTE format(  'COPY (SELECT COALESCE(cast(json\_agg(to\_json(t)) as text),''[]'') FROM "MediaFilesForPosts" t) TO %L',  pathToSave || '\mediafilesforposts.json'  );  EXECUTE format(  'COPY (SELECT COALESCE(cast(json\_agg(to\_json(t)) as text),''[]'') FROM "History" t) TO %L',  pathToSave || '\browsinghistory.json'  );  EXECUTE format(  'COPY (SELECT COALESCE(cast(json\_agg(to\_json(t)) as text),''[]'') FROM "Subscriptions" t) TO %L',  pathToSave || '\subscriptions.json'  );  EXECUTE format(  'COPY (SELECT COALESCE(cast(json\_agg(to\_json(t)) as text),''[]'') FROM "TagsRecipe" t) TO %L',  pathToSave || '\tagsrecipe.json'  );  EXECUTE format(  'COPY (SELECT COALESCE(cast(json\_agg(to\_json(t)) as text),''[]'') FROM "Tag" t) TO %L',  pathToSave || '\tags.json'  );  EXECUTE format(  'COPY (SELECT COALESCE(cast(json\_agg(to\_json(t)) as text),''[]'') FROM "Likes" t) TO %L',  pathToSave || '\likes.json'  );  EXECUTE format(  'COPY (SELECT COALESCE(cast(json\_agg(to\_json(t)) as text),''[]'') FROM "Ingridients" t) TO %L',  pathToSave || '\ingridients.json'  );  EXECUTE format(  'COPY (SELECT COALESCE(cast(json\_agg(to\_json(t)) as text),''[]'') FROM "Tokens" t) TO %L',  pathToSave || '\tokens.json'  );  END;  $$ LANGUAGE plpgsql; |

Листинг 4.16 – Функция export\_json

Таким образом, пользователи базы данных могут легко экспортировать и импортировать данные в формат JSON, что делает управление базой данных более удобным и эффективным.

## Вывод

В данном разделе были рассмотрены основные хранимые процедуры и функции, которые по сути обеспечивают 4 операции для взаимодействия с каждой из таблиц: create, get, update, delete.

Были также представлены процедуры экспорта и импорта данных в формат Json, а также были проведены тесты производительности базы данных. Кроме того, были представлены функции и процедуры, которые обеспечивают авторизацию и регистрацию пользователя, а также поиск по бд.

5 Тестирование

## 5.1 Тестирование производительности базы данных

Тестирование производительности является важным этапом разработки, поскольку позволяет определить, насколько хорошо база данных может обрабатывать запросы и как быстро она может возвращать результаты.

Для тестирования производительности базы данных была выбрана таблица Comments, содержащая больше всего данных. Для получения выборки данных использовался запрос, который представлен на листинге 5.1.

|  |
| --- |
| EXPLAIN ANALYZE SELECT “Text” FROM “Comment” WHERE “Text” ILIKE '%Comment%'; |

Листинг 5.1 – Запрос к таблице genre

Запрос к таблице Comments показывает, что его стоимость выполнения с Total Cost равным 4368 говорит о том, что выполнение запроса требует значительных затрат времени и ресурсов, особенно при сканировании всей таблицы и применении фильтра. Время выполнения запроса составляет 222,488 мс. Результаты запроса будут представлены на рисунке 5.2.

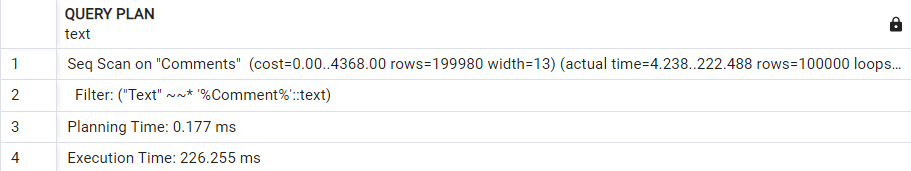


Рисунок 5.2 – Результат выполнения запроса

Для ускорения данного процесса можно создать индекс на поле Text, так как именно по этому полю выполняется фильтрация. После создания индекса, можно повторить запрос и сравнить стоимость с предыдущим запросом. Результат будет представлен на рисунке 5.3

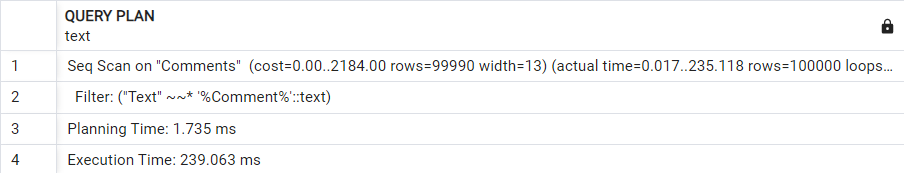


Рисунок 5.3 – Результат выполнения запроса

После создания индекса на поле Text запрос к таблице Comment имеет стоимость выполнения с Total Cost равным 2184,00, что является значительным улучшением по сравнению с предыдущим запросом. Время выполнения осталось примерно тем же - 235,118 мс. Результаты запроса также будут представлены на рисунке 5.3.

Тестирование показало, что создание индекса на поле Text значительно улучшило производительность запроса к таблице Comment. Стоимость выполнения запроса уменьшилась с 4368 до 2184, что говорит о том, что сканирование таблицы и применение фильтра стали занимать меньше времени и ресурсов. Время выполнения запроса осталось примерно тем же. Поэтому создание индексов на полях, по которым выполняются частые запросы, может значительно повысить производительность базы данных.

## 5.2 Вывод

В данном разделе было рассмотрено важное понятие тестирования производительности базы данных. Для проведения тестирования была выбрана таблица Comment с большим количеством данных. Проведенный тест показал, что создание индекса на поле Text значительно улучшило производительность запроса к таблице Comment уменьшив стоимость выполнения запроса. Выводом является то, что создание индексов на полях, по которым выполняются частые запросы, может значительно повысить производительность базы данных, что особенно важно при работе с большими объемами данных.

1. Краткое описание приложения для демонстрации

Для тестирования определенного функционала(к примеру загрузка медиафайлов или авторизации по токену), было необходимо использовать язык программирования. В качестве такого был использован C# и среда разработки Visual. Данное приложение фрагмент кода из данного приложения представлен в листинге 6.1

|  |
| --- |
| static void getUsers()  {  NpgsqlConnection connection = new NpgsqlConnection("Server=localhost;Port=5432;Database=culinary\_blog\_db;User Id=postgres;Password=1111;");  connection.Open();  try  {  string queryGetUsers = "SELECT \"Name\", \"Avatar\", \"Email\" FROM \"Users\"";  NpgsqlCommand commandGetUsers = new NpgsqlCommand(queryGetUsers, connection);  // Используем NpgsqlDataAdapter для выполнения запроса и получения результатов  NpgsqlDataAdapter dataAdapter = new NpgsqlDataAdapter(commandGetUsers);  DataTable dataTable = new DataTable();  dataAdapter.Fill(dataTable);  string directoryPath = @"C:\Users\HP\Desktop\Интерфейс к БД\Project1\images";  if (Directory.Exists(directoryPath))  {  // Получаем список файлов в папке  string[] files = Directory.GetFiles(directoryPath);  // Удаляем каждый файл  foreach (string file in files)  {  File.Delete(file);  }  }  foreach (DataRow row in dataTable.Rows)  {  string name = row["Name"].ToString();  byte[] imageBytes = (byte[])row["Avatar"];  string email = row["Email"].ToString();  if (imageBytes != null)  {  Directory.CreateDirectory(directoryPath);  string filePath = Path.Combine(directoryPath, $"{email}\_avatar.jpg");  File.WriteAllBytes(filePath, imageBytes);  }  Console.WriteLine($"Name: {name}, Email: {email}");  }  } |

# Заключение

База данных является неотъемлемой частью любого современного кулинарного блога, обеспечивая эффективное хранение и управление информацией. В рамках данной работы была выполнена разработка базы данных для кулинарного блога с использованием мощных возможностей СУБД PostgreSQL и специальных типов данных для работы с мультимедиа.

В процессе разработки были использованы различные объекты базы данных, включая таблицы, индексы и функции, с целью обеспечить структурированное хранение данных и быстрый доступ к ним. Результаты работы позволяют уверенно заявить, что база данных готова к использованию и полностью соответствует требованиям кулинарного блога.

Важной особенностью разработанной базы данных является возможность эффективного управления мультимедийными данными, такими как изображения блюд, видеорецепты и изображения с инструкциями. Это позволяет создать привлекательный и интерактивный опыт для пользователей кулинарного блога.

Проведение тестирования базы данных с использованием большого объема данных подтверждает ее высокую производительность и надежность. Были реализованы специальные процедуры для удобного импорта и экспорта данных в формате, удобном для обмена информацией.

В итоге, разработанная база данных полностью отвечает требованиям кулинарного блога и готова для использования. Она обеспечивает эффективное хранение и управление информацией, позволяет создавать интерактивный опыт для пользователей и способствует развитию блога.

# Список литературных источников

1. Epicurious [Электронный ресурс] / Режим доступа: https://www.epicurious.com/ – Дата доступа: 28.04.2023.

2. AllRecipes [Электронный ресурс] / Режим доступа: https://www.allrecipes.com/ – Дата доступа: 28.04.2023.

3.Povarenok [Электронный ресурс] / Режим доступа: https://www.povarenok.ru/com – Дата доступа: 28.04.2023.

4.Postgresqltutorial.com [Электронный ресурс] / Режим доступа: https://www.postgresqltutorial.com/ – Дата доступа: 28.04.2023.

5.Metanit Работа с файлами. Классы File и FileInfo [Электронный ресурс] / Режим доступа: https://metanit.com/sharp/tutorial/5.3.php – Дата доступа: 28.04.2023.