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 ИУ «Информатика и системы управления»

КАФЕДРА ИУ-7 «Программное обеспечение ЭВМ и информационные технологии»

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К КУРСОВОЙ РАБОТЕ

НА ТЕМУ:

«Разработка базы данных для хранения и обработки для антикафе»

Студент группы **ИУ7-66Б** _____ **В. М. Мансуров**
(Подпись, дата) (И.О. Фамилия)

Руководитель _____ **Л. Л. Волкова**
(Подпись, дата) (И.О. Фамилия)

2023 г.

РЕФЕРАТ

Расчетно-пояснительная записка 53 с., 15 рис., 14 лист., 19 ист., 1 прил.

БАЗА ДАННЫХ, PostgreSQL, СИСТЕМА УПРАВЛЕНИЯ БАЗАМИ ДАННЫХ, C#, РЕЛЯЦИОННАЯ МОДЕЛЬ.

Цель работы: проектирование и разработка программного обеспечения для хранения и обработки данных о антикафе.

В данной работе изучаются важные аспекты взаимодействия с базами данных, проектируется и реализуется информационное приложения для антикафе с системой бронирования залов, зон или комнат в антикафе. Для управления базой данных используется PostgreSQL, как надежная система управления базами данных, которая интегрируется с приложением, реализованным на языке программирования C# на платформе .Net 6.0.

В результате проделанной работы было создано приложение, которое не только обеспечивает доступ к реляционной базе данных, но и наилучшим способом выполняет запросы к базе данных.

СОДЕРЖАНИЕ

ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

В текущей расчетно-пояснительной записке применяется следующие сокращения и обозначения.

БД — база данных.

СУБД — система управления базами данных.

ПО — программное обеспечение.

ВВЕДЕНИЕ

С развитием культуры и сферы развлечений начало появляться всё больше различных мест, где можно хорошо и весело провести свободное время. Одними из таких стали антикафе. В связи с этим растет спрос на функциональные и удобные приложения для антикафе, которые помогали бы людям организовывать время проведения отдыха или развлечения.

Целью данного курсового проекта является проектирование и разработка программного обеспечения для хранения и обработки данных о антикафе.

Чтобы достигнуть поставленной цели, требуется решить следующие задачи:

— проанализировать варианты представления данных и выбрать подходящий вариант для решения задачи;
— проанализировать способы хранения данных и системы управления базами данных, выбрать подходящую систему для поставленной цели;
— спроектировать базу данных, описать ее сущности и связи;
— реализовать программное обеспечение, позволяющее взаимодействовать со спроектированной базой данных;
— провести исследования зависимости времени выполнения запроса от объема обрабатываемых данных.

1 Аналитическая часть

В данном разделе осуществляется анализ области исследования, включая описание концепции антикафе, классификацию доступных систем управления базами данных, формализацию поставленных задач и структуры данных, а также оценку имеющихся решений в этой области. На основе этого анализа формируется список пользователей, определяются сценарии использования в системе, и создаются схемы данных, включая диаграмму вариантов использования и диаграмму сущность-связь в нотации Чена.

Кроме того, основываясь на характеристиках данных, которые будут храниться в системе, выбирается наилучшая модель базы данных. В этой части работы также проводится анализ различных методов хранения данных, и выбирается наилучший метод для решения поставленной задачи.

1.1 Анализ предметной области

На протяжении всей свой жизни человек находится, развивается и живет в обществе, которое формирует ценности, стереотипы, и нормы поведения. Но также в жизни человека важную роль играет организация его досуга, свободного времени, которое освобождено от других видов деятельности [1].

Антикафе — заведение для реализации культурно-досуговой деятельности, в котором посетители обладают большой степенью свободы, чем кафе или рестораны, основной характеристикой которого является плата за проведенное время. Основная задача антикафе — предоставить гостям рабочую, творческую или развлекательную атмосферу. Целью посещения таких мет является не утоления голода как такого, а приятное время провождение, развлечение и посещение тематических мероприятий и т.п. Обычно антикафе состоят из одного большого зала или нескольких комнат, где люди свободно перемещаются и поэтому оно подразделяется на зоны:

— «Развитие» — пространство, где проводятся лекции, тренинги, курсы, интеллектуальные игры, мастер-классы и т.п.;
— «Работа» — пространство, где можно спокойно поработать, т.е. представляет из себя коворкинг;
— «Развлечение» — пространство под различные игры, концертов, кино и т.п.;
— «Творчество» — пространство для реализаций творческой деятельности.

Также в одной из комнат есть зона с угощениями, в которой гости могут взять печенья или сладости, а также налить себе кофе, чай [1].

1.2 Анализ существующих решений

Формат заведения антикафе уже существует около десятка лет поэтому в данной области существуют информационная системы с механизмами бронирования для таких заведений, но у них есть недостатки. Наиболее популярными являются:

1) Party Hard — один из наиболее известных сайт антикафе, который предоставляет возможность просмотра информации о зонах и оформление брони без аккаунта [2];
2) Bizone — сайт сеть антикафе, который предоставляет возможность просмотра информации зон, меню и оформление брони по звонку в антикафе [3];
3) SpeedRent — сайт бронирования развлекательных заведений, который предоставляет возможность просмотра информации о зонах и оформление брони в указанную дату и время [4].

Критерии, выделенные для сравнения существующих решений:

1) возможность иметь аккаунт;
2) меню — предоставления списка меню блюд пользователю;

3) бронирование — система оформление записи о закреплении зоны за клиентом на указанное время;

4) рейтинг — формирование оценки зоны по оставленным отзывам;

Таблица 1.1 – Категории и сведения о данных

Критерий	**Party Hard**	**Bizone**	**SpeedRent**
Возможность иметь аккаунт	нет	нет	есть
Меню	нет	есть	нет
Бронирование	бронировать можно, но не на указанное время, после оформления онлайн брони надо согласовывать время по звонку	бронирование по звонку	есть
Рейтинг	нет	нет	есть

Все вышеперечисленные существующие решения не предоставляют пользователю необходимый функционал для принятий решения выбора. В результате, данная работа отличается от существующих решений тем, что она удовлетворяет все установленные критерии.

1.3 Формализация задачи

Необходимо спроектировать базу данных для хранения информации о пользователях, зонах, пакетах, инвентаре, меню, отзывах и бронях залов. Требуется разработать приложение, предоставляющее интерфейс для просмотра, добавления, изменения и удаления информации, хранящейся в базе данных. Необходимо реализовать три вида ролей — гость, авторизованный пользователь и администратор.

В рамках поставленной цели необходимо спроектировать и реализовать базу данных, содержащую информацию о зонах, меню, пользователях анти-

кафе и позволяющую изменить хранящейся в ней информации. Разработать механизм бронирования зон в антикафе.

1.4 Формализация данных

Основываясь на анализе предметной области, можно выделить следующие категории данных:

— пользователь;

— зона;

— бронь;

— отзыв;

— инвентарь;

— пакет;

— блюдо.

Сведения о каждой категории данных содержится в таблице 1.2.

Таблица 1.2 – Категории и сведения о данных

Категория	**Сведения**
Пользователь	ФИО, дата рождения, пол, телефон, email, пароль, права доступа, адрес,
Зона	Название, тип, размер в кв. м., рейтинг, ограничение на количество людей в зоне
Бронь	Пользователь, зона, пакет, дата бронирования, время начала и конца брони, количество людей, статус, дата и время создания брони, оплачена ли бронь, итоговая цена
Отзыв	Пользователь, зона, оценка, дата и время создания, комментарий
Инвентарь	Название, тип, дата выпуска, описание, списан ли инвентарь
Пакет	Название, тип, время проведение, стоимость, описание
Блюдо	Название, тип, цена, описание

На рисунке 1.1 приведена ER-диаграмма сущностей в нотации Чена.

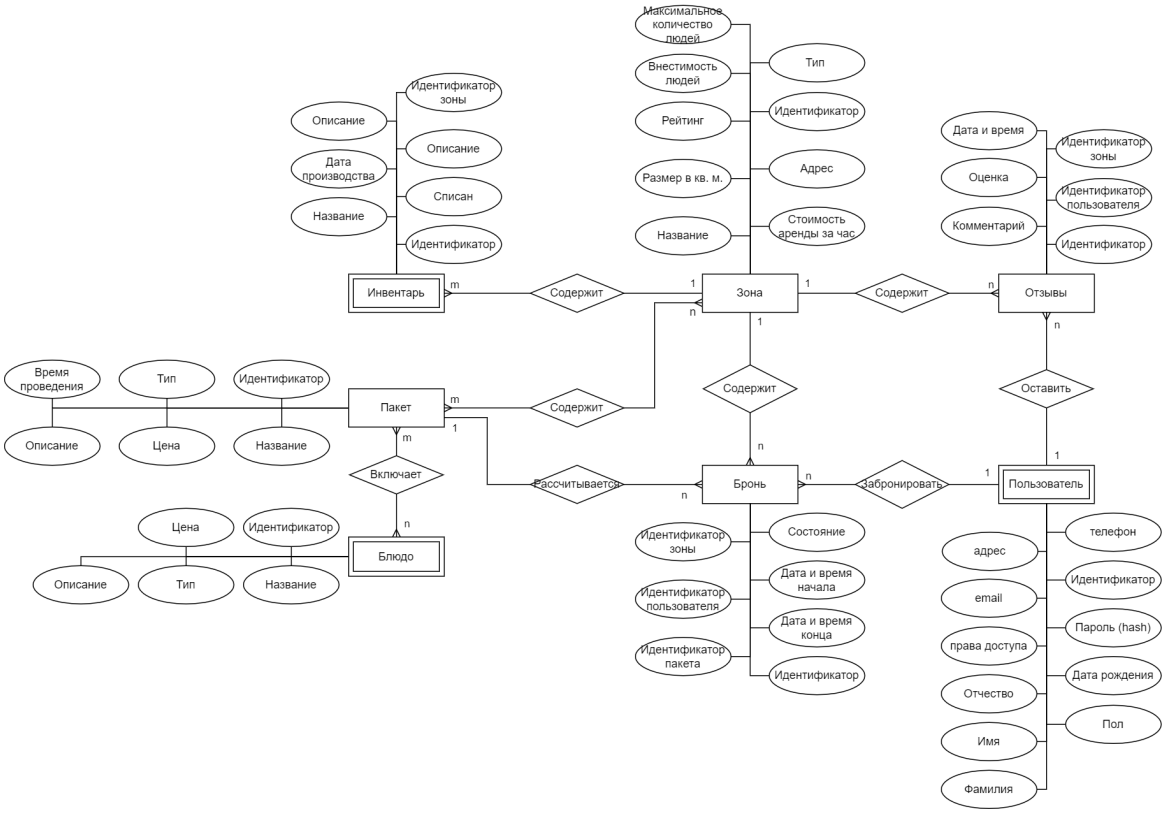

Рисунок 1.1 – ER-диаграмма сущностей

Для бронирования зоны антикафе пользователем необходимо разработать систему бронирования, которая основана на изменении статуса брони:

— изначально авторизованный пользователь выбирает дату и время брони, после чего должна создаваться бронь с статусом «временная бронь»;

— после заполнения всех необходимых данных для брони пользователь подтверждает бронь и статус брони меняется с «временной брони» на «забронировано»;

— пользователь может отменить бронь, что приводит к изменению статуса брони на «отменена»;

— если бронь в статусе «временная бронь» и время для бронирования истекает, то такая бронь удаляется из базы данных;

— если бронь в статусе «забронировано», то по истечению дата и времени конца брони она переводится в статус «выполнена».

На рисунке 1.2 приведена диаграмма состояний брони зоны.

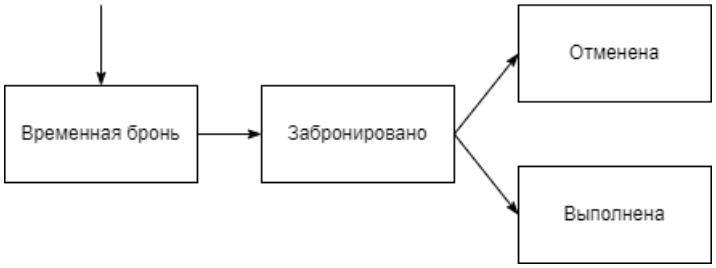

Рисунок 1.2 – Диаграмма состояний брони

1.5 Формализация ролей

Выделим группы пользователей разрабатываемой системы, исходя из предметной области поставленной задачи.

В системе выделяются 3 типа пользователей:

— гость — неавторизованный пользователь, обладающий возможностями регистрироваться, входить в систему, просматривать данные зон, пакетов, блюд и отзывов;

— авторизованный пользователь обладает возможностью просматривать данные зон, пакетов, блюд, отзывов, бронировать зону и отменять бронь зоны, оставлять и удалять отзыв, просматривать созданные брони;

— администратор — авторизованный пользователь, обладающий возможностью просматривать, добавлять и менять данные зон, пакетов, инвентаря, блюд, отзывов, бронь и пользователей;

На рисунке 1.3 приведена Use-case диаграмма.

Рисунок 1.3 – Use-case диаграмма

1.6 Анализ моделей баз данных

База данных — это некоторый набор перманентных (постоянно хранимых) данных, используемых прикладными программными системами какого-либо предприятия [5].

По типу назначения базы данных делятся на:

1) OLAP [5] — это метод обработки данных, предназначенный для анализа больших объемов информации;
2) OLTP [5] — это метод обработки транзакций, предназначенный для выполнения операций в реальном времени.

СУБД — совокупность языковых и программных средств общего или специального назначения, обеспечивающих управление созданием и использованием баз данных [5].

Основными функциями СУБД являются:

— управление данными во внешней памяти;;

— управление буферами оперативной памяти;

— управление транзакциями;

— ведение журнала изменений в базе данных;

— обеспечение целостности и безопасности базы данных.

Модель данных — это абстрактное и логическое определение объектов и его поведение, в совокупности составляющих доступ к данным, с которой взаимодействует пользователь [5]. С помощью модели данных могут быть представлены объекты предметной области и взаимосвязи между ними.

СУБД различаются по модели данных, которая определяет архитектуру, структуры данных, обработку данных этой СУБД.

1) Дореляционные модели. Наиболее известные представители:
   — иерархические;
   — сетевые;
   — инвертированные списки.
2) Реляционные модели данных.
3) Постреляционные модели данных.

Дореляционные базы данных

Иерархическая БД состоит из упорядоченного набора деревьев; более точно, из упорядоченного набора нескольких экземпляров одного типа дерева [6]. В такой базе данных информация представлена в виде уровней, где каждый уровень имеет родительский и дочерний элемент. Эта модель данных особенно полезна для представления иерархических отношений и структур, как представлено на рисунке 1.4.

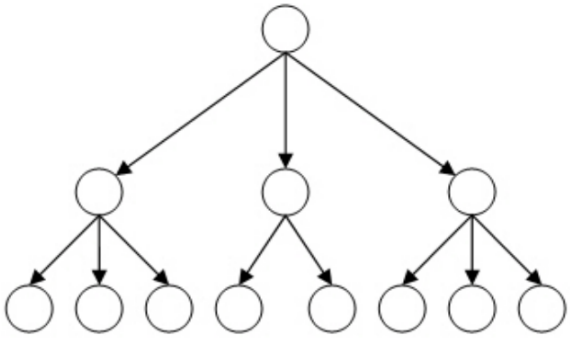

Рисунок 1.4 – Структура иерархической модели данных

Сетевая модель данных — логическая модель данных, являющаяся расширением иерархического подхода. В иерархических структурах запись–потомок должна иметь в точности одного предка, а в сетевой структуре данных потомок может иметь любое число предков [6], как показано на рисунке 1.5

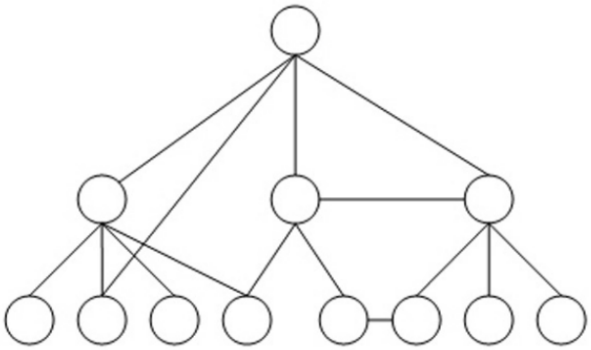

Рисунок 1.5 – Структура сетевой модели данных

Дореляционные модели близки к управлению данными во внешней памяти на низком уровне, что при водит к наилучшему использованию памяти, но такие модели данных имеют сложность использования, зависимость прикладных систем от физической организации. Так как логика процедуры выбора данных зависит от физической организации этих данных, то данная модель не является полностью независимой от приложения и как следствие приводит к усложнению действий изменений в базе данных, таких как добавление, удаление или изменение записей и связей.

Реляционные базы данных

Согласно Дейту реляционная модель состоит из трех частей, описывающих разные аспекты реляционного подхода.

В структурной части фиксируется, что единственной структурой данных используемой в БД, является n-арное нормализованное отношение и понятия структуры реляционной модели: тип данных, домен, атрибут отношения, схема отношения, схема БД, кортеж, отношение, первичный и потенциальный ключи [6].

В целостной части реляционной модели определяются два требования целостности. Во-первых, требование целостности сущностей, которому соответствует кортеж отношений, то есть любой кортеж отношения обладает первичным ключом. Во-вторых, требование целостности по ссылкам, которое достигается соблюдением нормализованности отношений сложных сущностей представляемых в БД в виде нескольких отношений [6].

В манипуляционной части модели утверждаются два механизма манипулирования — реляционная алгебра и реляционное исчисление. Первый механизм базируется в основном на классической теории множеств, а второй — на классическом логическом аппарате исчисления предикатов первого порядка [6].

Основным достоинством использование реляционной модели заключается в простоте, понятности и удобстве физической реализации, что привело в широкому использованию, то есть формализация табличного представления данных, состоящего из записей и полей. Также независимость данных, то есть изменение структуры реляционной БД приводит к минимальным модификациям приложения. Важно отметить, что у реляционной модели отсутствие стандартные средства идентификации отдельных записей и сложность описания иерархических и сетевых связей. Последние версии реляционных СУБД имеют некоторые свойства объектно-ориентированных систем. Такие СУБД часто называют объектно-реляционным.

Постреляционные базы данных

Классическая реляционная модель предполагает неделимость данных, хранящихся в полях записей таблиц. Это означает, что информация в таблице представляется в первой нормальной форме. Существует ряд случаев, когда это ограничение мешает эффективной реализации приложений. Она использует трехмерные структуры, позволяя хранить в полях таблицы другие таблицы, расширяя таким образом возможности по описанию сложных объектов реального мира [7].

Постреляционная модель данных представляет собой расширенную реляционную модель, снимающую ограничение неделимости данных, хранящихся в записях таблиц. Постреляционная модель данных допускает многозначные поля — поля, значения которых состоят из подзначений. Набор значений многозначных полей считается самостоятельной таблицей, встроенной в основную таблицу [7].

Достоинством постреляционной модели является возможность представления совокупности связанных реляционных таблиц одной постреляционной таблицей. Это обеспечивает высокую наглядность представления информации и повышение эффективности ее обработки.

Вывод

В данном разделе была проанализирована предметная область, поставленная задача и рассмотрены способы ее реализации. Так как задача предполагает использование разнообразных запросов различной сложности, приоритетным свойством модели данных является гибкость, простота использования и независимость от приложения, поэтому в качестве модели организации данных была выбрана реляционная модель БД.

2 Конструкторская часть

В данном разделе производится формализация сущностей в системе, устанавливается ролевая модель и определяются необходимые процедуры, функции или триггеры. Каждой сущности, описанной в аналитической части, присваиваются соответствующие поля, содержащие информацию, необходимую для их полного определения.

На основе этой информации разрабатывается диаграмма сущность-связь базы данных, на которой отображаются все необходимые таблицы для правильной для корректной работы приложения.

2.1 Описание сущностей базы данных

В соответствии с таблицей 1.2, содержащей данные, которые должны находиться в базе данных, можно выделить следующие таблицы:

1) users — таблица пользователей;
2) zones — таблица зон, залов, комнат антикафе;
3) bookings — таблица броней зон антикафе;
4) inventories — таблица инвентаря зон антикафе;
5) feedbacks — таблица отзывов пользователей о зонах антикафе;
6) dishes — таблица меню блюд антикафе;
7) packages — таблица пакетов для формирования расчета оплаты за аренду зон антикафе.

На основе информации о выбранной СУБД и представленной на рисунке 1.1 диаграмме сущность-связь, мы определим для каждой из указанных таблиц структуру столбцов, их типы и установим соответствующие ограничения, которые представлены на в таблицах 2.1–2.7.

Таблица 2.1 – Информация о столбцах таблицы пользователей

Столбец	Тип данных	Ограничения	Значение
id	uuid	NOT NULL, PRIMARY KEY	Идентификатор пользователя
last_name	VARCHAR(64)	NOT NULL	Фамилия
first_name	VARCHAR(64)	NOT NULL	Имя
middle_name	VARCHAR(64)	NOT NULL	Отчество
birthday	DATE	NOT NULL	Дата рождения
gender	VARCHAR(64)	NOT NULL	Пол
email	TEXT	NOT NULL	Электронная почта, логин
password	VARCHAR(256)	NOT NULL	Хэш пароля
role	VARCHAR(64)	NOT NULL	Права доступа

Таблица 2.2 – Информация о столбцах таблицы зон антикафе

Столбец	Тип данных	Ограничения	Значение
id	uuid	NOT NULL, PRIMARY KEY	Идентификатор зоны
name	VARCHAR(64)	NOT NULL	Название
address	TEXT	NOT NULL	Адрес
size	NUMERIC	NOT NULL	Размер в кв. метрах
limit	INTEGER	NOT NULL	Максимальное количество людей в зоне
rating	NUMERIC	NOT NULL	Рейтинг

Таблица 2.3 – Информация о столбцах таблицы броней зон антикафе

Столбец	Тип данных	Ограничения	Значение
id	uuid	NOT NULL, PRIMARY KEY	Идентификатор брони
zone_id	uuid	NOT NULL, FOREIGN KEY	Идентификатор зоны
user_id	uuid	NOT NULL, FOREIGN KEY	Идентификатор пользователя
package_id	uuid	NOT NULL, FOREIGN KEY	Идентификатор пакета
amount_of_people	INTEGER	NOT NULL	Количество людей
status	VARCHAR(64)	NOT NULL	Статус
date	DATE	NOT NULL	Дата брони
start_time	TIME	NOT NULL	Время начало брони
end_time	TIME	NOT NULL	Время конца брони
create_date_time	TIMESTAMP	NOT NULL	Дата и время создания брони
is_piad	BOOLEAN	NOT NULL	Оплачена ли бронь
total_price	NUMERIC	NOT NULL	Итоговая цена

Таблица 2.4 – Информация о столбцах таблицы инвентаря антикафе

Столбец	Тип данных	Ограничения	Значение
id	uuid	NOT NULL, PRIMARY KEY	Идентификатор инвентаря
zone_id	uuid	NOT NULL, FOREIGN KEY	Идентификатор зоны
name	VARCHAR(64)	NOT NULL	Название
description	TEXT	NOT NULL	Описание
date_production	DATE	NOT NULL	Дата производства
is_written_off	BOOLEAN	NOT NULL	Значение обозначающее списан ли инвентарь

Таблица 2.5 – Информация о столбцах таблицы отзывов зон антикафе

Столбец	Тип данных	Ограничения	Значение
id	uuid	NOT NULL, PRIMARY KEY	Идентификатор отзыва
zone_id	uuid	NOT NULL, FOREIGN KEY	Идентификатор зоны
user_id	uuid	NOT NULL, FOREIGN KEY	Идентификатор пользователя
date	TIMESTAMP	NOT NULL	Дата и время создания
mark	NUMERIC	NOT NULL	Оценка пользователя
message	TEXT	NOT NULL	Комментарий пользователя

Таблица 2.6 – Информация о столбцах таблицы меню блюд антикафе

Столбец	Тип данных	Ограничения	Значение
id	uuid	NOT NULL, PRIMARY KEY	Идентификатор блюда
name	VARCHAR(64)	NOT NULL	Название
type	VARCHAR(64)	NOT NULL	Тип блюда
price	NUMERIC	NOT NULL	Цена в рублях
description	TEXT	NOT NULL	Описание

Таблица 2.7 – Информация о столбцах таблицы пакетов антикафе

Столбец	Тип данных	Ограничения	Значение
id	uuid	NOT NULL, PRIMARY KEY	Идентификатор пакета
name	VARCHAR(64)	NOT NULL	Название
type	VARCHAR(64)	NOT NULL	Тип пакета
price	NUMERIC	NOT NULL	Цена в рублях
rental_time	INTEGER	NOT NULL	Количество времени проведения
description	TEXT	NOT NULL	Описание

На рисунке 2.1 приведена ER-диаграмма базы данных.

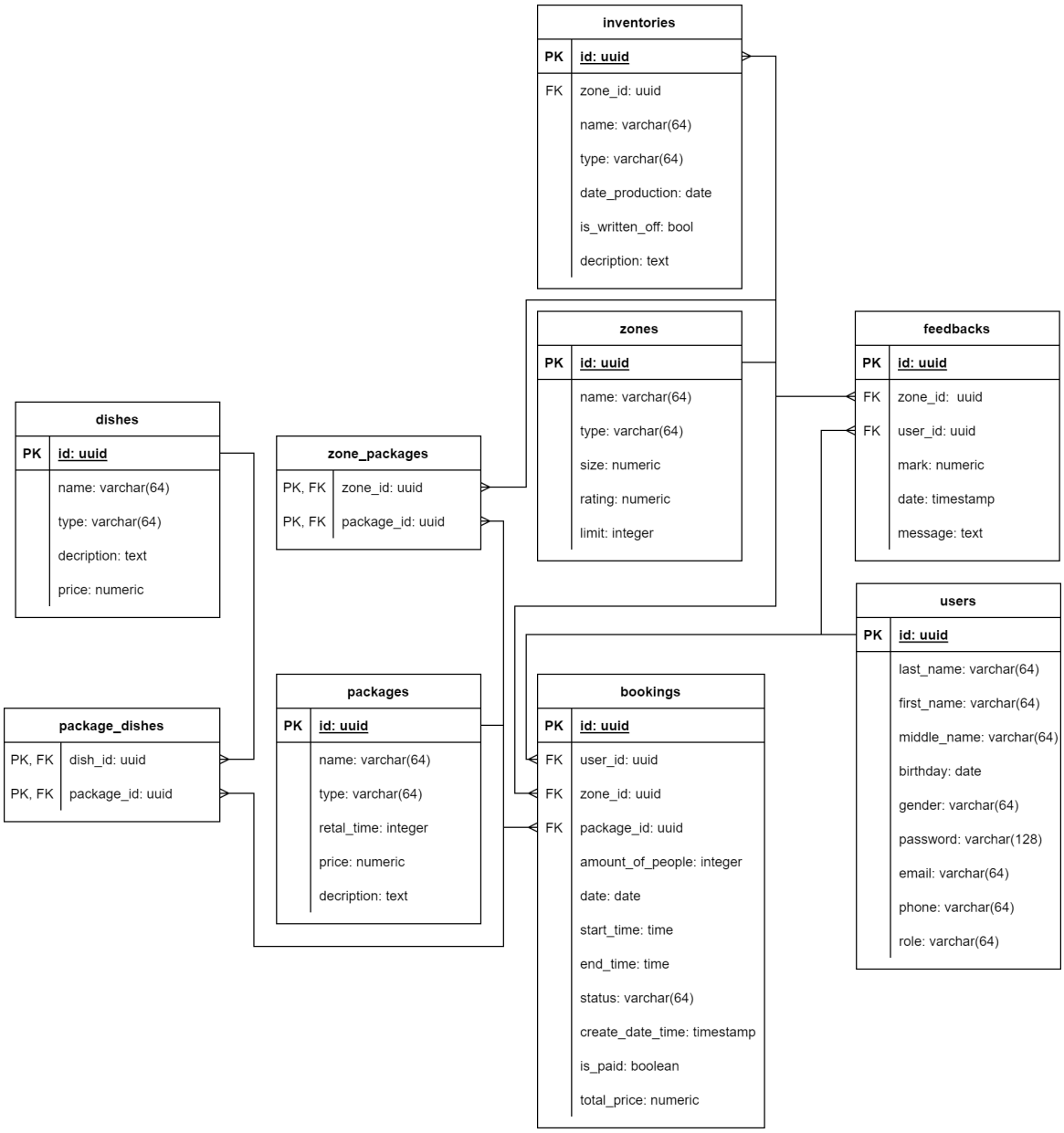

Рисунок 2.1 – ER-диаграмма базы данных

2.2 Ролевая модель

Ролевая модель в системе играет важную роль в организации работы пользователей, обеспечивая им доступ к определенному набору функций в системе.

Что касается работы пользователей с базой данных, то для этой цели определена следующая ролевая модель:

— Гость имеет права доступ SELECT ко всем таблицам, кроме таблиц пользователей и броней, INSERT только к таблице пользователей;
— Авторизованный пользователь имеет права доступа SELECT ко всем таблицам, INSERT только к таблице броней, UPDATE к таблицам броней и пользователей;
— Администратор имеет все права доступа ко всем таблицам.

2.3 Используемые триггеры

В базе данных будет присутствовать специальный триггер, который активируется автоматически после добавления новой записи в таблицу отзывов. Этот триггер выполняет пересчет рейтинга соответствующей зоны антикафе на основе оценки, указанной в отзыве пользователем. При каждой вставке новых данных в таблицу отзывов, данный триггер будет автоматически вызываться, обеспечивая актуальность рейтинга.

На рисунке 2.2 представлена схема триггера.

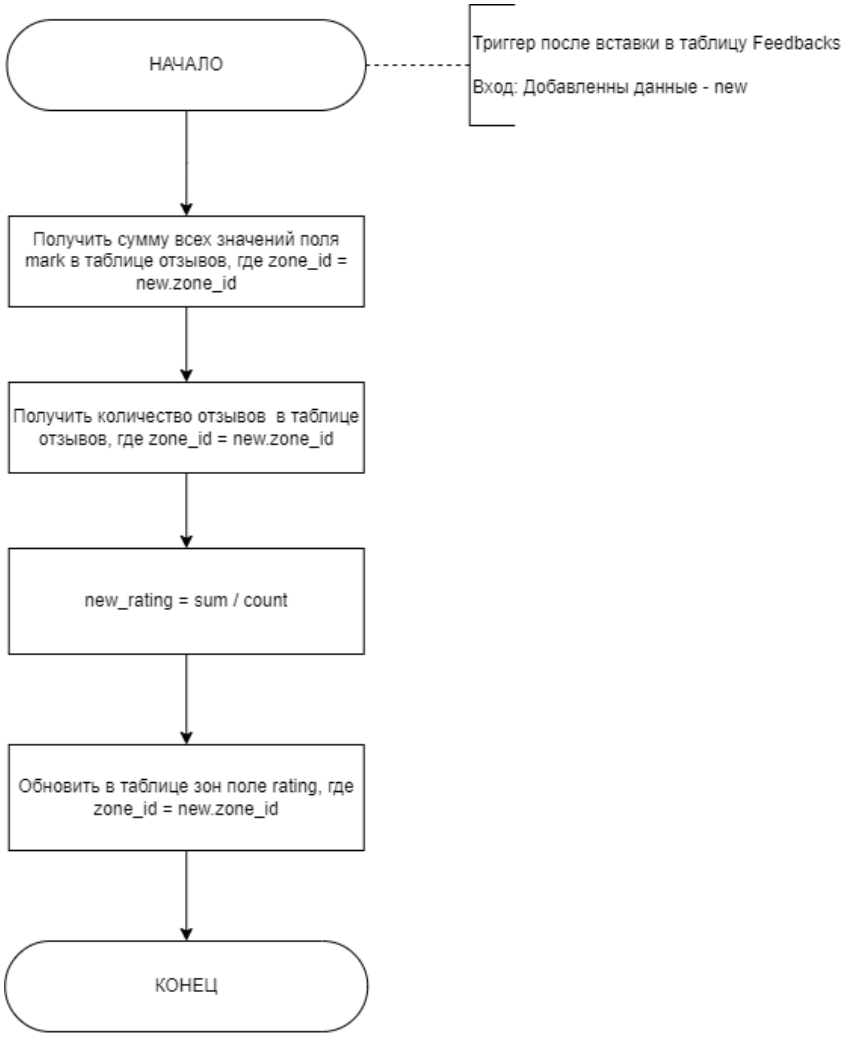

Рисунок 2.2 – Схема алгоритма триггера после добавления записи в таблицу feedbacks

2.4 Диаграмма классов приложения

Приложение строится согласно диаграмме классов, приведенной на рисунке 2.3. На диаграмме классов представлена схема реализаций как части доступа базы данных, то есть реализация паттерна «Репозитория» для каждой сущности базы данных и части бизнес логики, где реализованы сервисы по обработки данных в необходимый вид для пользователя.

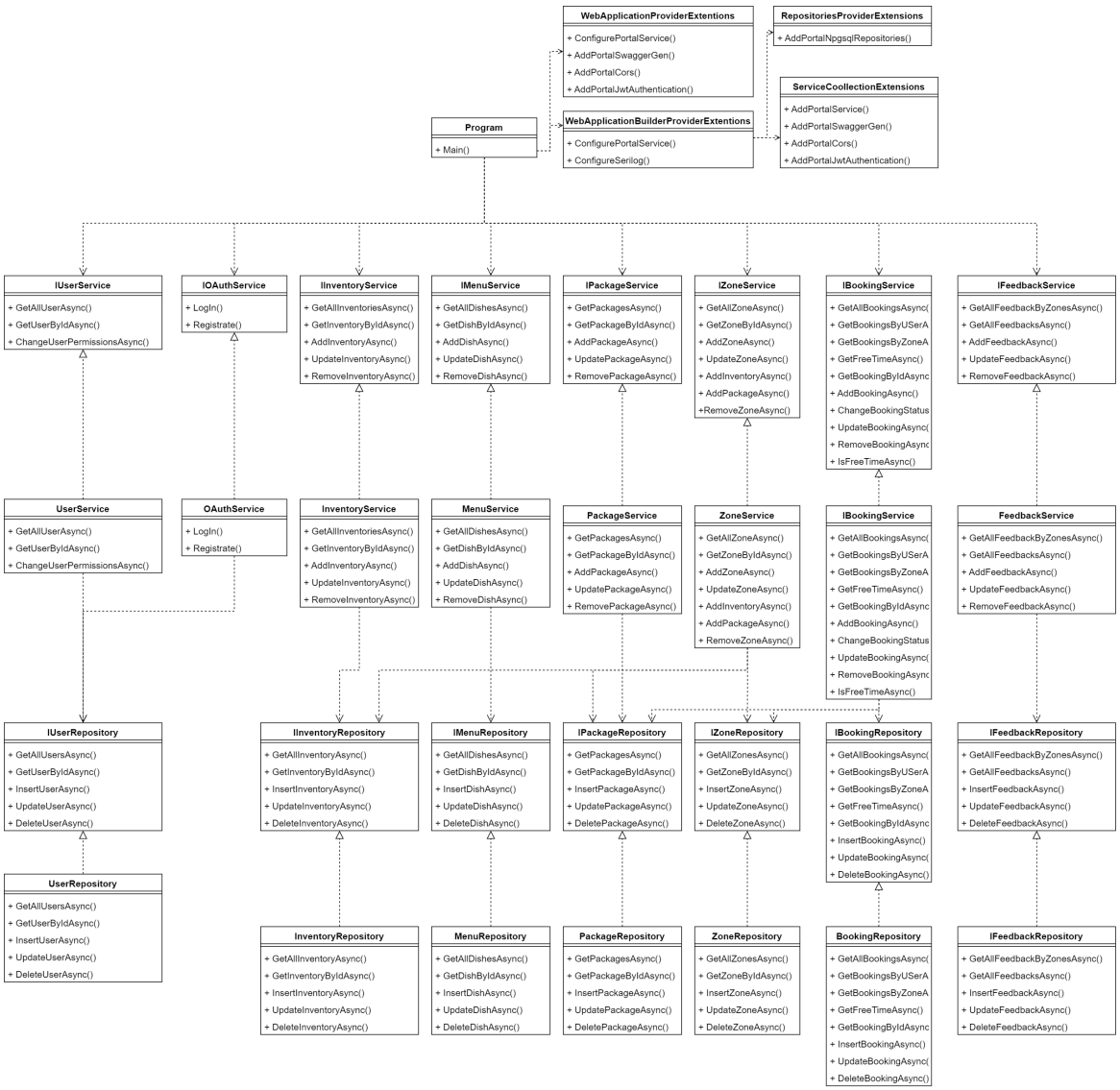

Рисунок 2.3 – Диаграмма классов приложения

Вывод

В данном разделе были описаны все сущности базы данных на основе анализа предметной области, представлена ER-диаграмма сущностей, описана ролевая модель. Также был спроектирован необходимые триггеры, функции или процедуры. Приведена диаграмма классов приложения, которое будет разработана для интеграции с базой данных.

3 Технологическая часть

В данной части рассматривается выбор средств реализации, описывается структура классов программы и приводится интерфейс программного обеспечения.

Исходя из ролевой модели, представленной в предшествующих разделах создается ее реализация. Также предоставляется код для создания ролей, настройки их прав доступа и также создания триггера. Кроме того, предоставляется практический пример работы разработанного приложения.

3.1 Выбор СУБД

Самыми распространенными [8] СУБД реляционной модели БД являются:

— PostgreSQL [7, 9];

— SQLite [9];

— MySQL [9].

Выделим следующие критерии для сравнения полярных СУБД реляционной модели:

1) возможность создавать роли, пользователей и выдавать им права доступа на уровне БД;
2) активная поддержка;
3) полное соответствие SQL.

В таблице 3.1 представлены результат сравнительного анализа популярных реляционных СУБД с учетом установленных критериев.

Таблица 3.1 – Сравнение популярных реляционных СУБД

Критерий	**SQLite**	**MySQL**	**PostgreSQL**
Создания ролей	нет	есть	нет
Поддержка	есть	нет	есть
Соответствие SQL	нет	нет	есть

В данном работе выбрана PostgreSQL, поскольку она обеспечивает достаточный инструментарий для выполнения поставленной цели.

3.2 Выбор средств реализации

В качестве языка программирования для разработки программного обеспечения был применен язык C# [10], поскольку:

— является объектно-ориентированным языком программирования, что позволяет создавать классы, объекты и методы, упрощая создания доступа к объектам базы данных и их взаимодействие связей;

— является частью платформы .Net [11], которая предоставляет необходимый набор библиотек и инструментов для написания надежного и производительного программного продукта, работающего с различными базами данных;

— имеет встроенный механизм LINQ [12], который предоставляет возможности выполнения запросов к базе данных на уровне языка.

В качестве среда разработки был выбран Visual Studio [13], так как:

— данная среда разработки предоставляется бесплатно;

— поддерживает различный набор фреймворков NuGet [14];

— предоставляет интеграция управления версиями Git [15];

— мощные средства написания кода и функции — все, что необходимо для создания приложений в одном месте;

— предоставляет шаблоны для создание и сборки проектов , что упрощает процесс разработки.

3.3 Создание базы данных

На листинге 3.1 представлено создание базы данных.

Листинг 3.1 – Создание базы данных

```
CREATE DATABASE "PortalDb"
WITH
OWNER = postgres
ENCODING = 'UTF8'
LC_COLLATE = 'Russian_Russia.1251'
LC_CTYPE = 'Russian_Russia.1251'
TABLESPACE = pg_default
CONNECTION LIMIT = -1;
```

На листингах 3.2–3.6 представлено создание таблиц базы данных

Листинг 3.2 – Создание таблицы users

```
CREATE TABLE IF NOT EXISTS public.users
(
    id uuid NOT NULL PRIMARY KEY,
    last_name varchar(64) NOT NULL,
    first_name varchar(64) NOT NULL,
    middle_name varchar(64),
    birthday timestamp with time zone NOT NULL,
    gender integer NOT NULL,
    email varchar(64) NOT NULL,
    phone varchar(64),
    password character varying(128) NOT NULL,
    role character varying(64) NOT NULL
);
```

Листинг 3.3 – Создание таблиц zones inventories packages

```
CREATE TABLE IF NOT EXISTS public.zones
(
    id uuid NOT NULL PRIMARY KEY,
    name varchar(64) NOT NULL,
    address text NOT NULL,
    size double precision NOT NULL,
    "limit" integer NOT NULL,
    rating numeric NOT NULL
);

CREATE TABLE IF NOT EXISTS public.inventories
(
    id uuid NOT NULL PRIMARY KEY ,
    zone_id uuid NOT NULL
        REFERENCES public.zones (id)
        MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE CASCADE,
    name varchar(64) NOT NULL,
    description text NOT NULL,
    date_production date NOT NULL,
    is_written_off boolean NOT NULL
);

CREATE TABLE IF NOT EXISTS public.packages
(
    id uuid NOT NULL PRIMARY KEY,
    name varchar(64) NOT NULL,
    type varchar(64) NOT NULL,
    price numeric NOT NULL,
    rental_time integer NOT NULL,
    description text NOT NULL
);
```

Листинг 3.4 – Создание ассоциативной таблицы zone_packages и таблицы dishes

```
CREATE TABLE IF NOT EXISTS public.zone_packages
(
    package_id uuid NOT NULL
        REFERENCES public.zones (id)
        MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE CASCADE,
    zone_id uuid NOT NULL
        REFERENCES public.packages (id)
        MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE CASCADE,

    CONSTRAINT pk_zone_packages PRIMARY KEY (package_id, zone_id)
);

CREATE TABLE IF NOT EXISTS public.dishes
(
    id uuid NOT NULL PRIMARY KEY,
    name varchar(64) NOT NULL,
    type varchar(64) NOT NULL,
    price numeric NOT NULL,
    description text NOT NULL
);
```

Листинг 3.5 – Создание ассоциативной таблицы package_dishes и таблицы feedbacks

```
CREATE TABLE IF NOT EXISTS public.package_dishes
(
    dish_id uuid NOT NULL
    REFERENCES public.dishes (id)
    MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE CASCADE,
    package_id uuid NOT NULL
    REFERENCES public.packages (id)
    MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE CASCADE,
    CONSTRAINT pk_package_dishes PRIMARY KEY (dish_id, package_id)
);

CREATE TABLE IF NOT EXISTS public.feedbacks
(
    id uuid NOT NULL PRIMARY KEY,
    user_id uuid NOT NULL
        REFERENCES public.users (id)
        MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE CASCADE,
    zone_id uuid NOT NULL
        REFERENCES public.zones (id)
        MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE CASCADE,
    date timestamp with time zone NOT NULL,
    mark numeric NOT NULL,
    message text
);
```

Листинг 3.6 – Создание таблицы bookings

```
CREATE TABLE IF NOT EXISTS public.bookings
(
    id uuid NOT NULL PRIMARY KEY,
    zone_id uuid NOT NULL
        REFERENCES public.zones (id)
        MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE CASCADE,
    user_id uuid NOT NULL
        REFERENCES public.users (id)
        MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE CASCADE,
    package_id uuid NOT NULL
        REFERENCES public.packages (id)
        MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE CASCADE,
    amount_of_people integer NOT NULL,
    status varchar(64) NOT NULL,
    date date NOT NULL,
    start_time time without time zone NOT NULL,
    end_time time without time zone NOT NULL,
    create_date_time timestamp with time zone NOT NULL,
    is_paid boolean NOT NULL,
    total_price numeric NOT NULL
);
```

На листингах 3.7–3.9 представлено создание ролей базы данных.

Листинг 3.7 – Создание ролей и выдача прав доступа (часть 1)

```
CREATE ROLE portal_admin WITH
SUPERUSER
CREATEDB
CREATEROLE
NOINHERIT
NOREPLICATION
NOBYPASSRLS
CONNECTION LIMIT -1
LOGIN
PASSWORD 'PaS$woRdAdm1N';

GRANT ALL PRIVILEGES
ON ALL TABLES IN SCHEMA public
TO portal_admin;

CREATE ROLE portal_user WITH
NOSUPERUSER
NOCREATEDB
NOCREATEROLE
NOINHERIT
NOREPLICATION
NOBYPASSRLS
CONNECTION LIMIT -1
LOGIN
PASSWORD 'PaS$woRdUser';

GRANT SELECT
ON ALL TABLES IN SCHEMA public
TO portal_user;
```

Листинг 3.8 – Создание ролей и выдача прав доступа (часть 2)

```
GRANT INSERT
ON public.users,
public.bookings,
public.feedbacks
TO portal_user;

GRANT DELETE
ON public.bookings,
public.feedbacks
TO portal_user;

GRANT UPDATE
ON public.bookings,
public.feedbacks
TO portal_user;

CREATE ROLE portal_guest WITH
NOSUPERUSER
NOCREATEDB
NOCREATEROLE
NOINHERIT
NOREPLICATION
NOBYPASSRLS
CONNECTION LIMIT -1
LOGIN
PASSWORD 'PaS$woRdGuest';
```

Листинг 3.9 – Создание ролей и выдача прав доступа (часть 3)

```
GRANT SELECT
ON public.zones,
public.inventories,
public.feedbacks,
public.packages,
public.zonepackages,
public.dishes,
public.packagedishes
TO portal_guest;

GRANT INSERT
ON public.users
TO portal_guest;
```

На листингах 3.10, 3.11 представлено создание триггера после добавления записи в таблицу feedbacks на перерасчет рейтинга зоны антикафе по оставленным оценкам в отзыве пользователями.

Листинг 3.10 – Создание триггера (Часть 1)

```
CREATE OR REPLACE FUNCTION public.calculate_rating_zone()
RETURNS TRIGGER
AS $$
BEGIN
    UPDATE public.zones
    SET rating = (Select sum(DISTINCT f.mark) / count(*) as rating
                  from public.feedbacks as f
                   where f.zone_id = new.zone_id)
    WHERE id = new.zone_id;
RETURN NEW;
END;
$$ LANGUAGE PLPGSQL;
```

Листинг 3.11 – Создание триггера (Часть 2)

```
1 CREATE TRIGGER insert_feedbacks_trigger
2 AFTER INSERT ON public.feedbacks
3 FOR EACH ROW
4 EXECUTE FUNCTION public.calculate_rating_zone();
```

3.4 Интерфейс взаимодействия

Для работы с базы данных был создан API с помощью библиотеки Swashbuckle [16]. В программном интерфейсе реализованы методы для выполнения операций создания, чтения и удаления всех созданных сущностей в базе данных. На рисунках 3.1–3.3 представлен методы интерфейса программы для взаимодействия с сущностями базы данных.

Методы взаимодействия с таблицей броней базы данных:

— получение всех броней;

— получение брони по идентификатору;

— получение всех броней по идентификатору пользователя;

— получение всех броней по идентификатору зоны;

— создание брони;

— подтверждение брони;

— отмена брони.

Методы взаимодействия с таблицей отзывов базы данных:

— получение всех отзывов;

— получение всех отзывов по идентификатору зоны;

— создание отзыва;

— удаление отзыва.

Методы взаимодействия с таблицей инвентаря базы данных:

— получение всего инвентаря;

— списать инвентарь зоны;

— создать и добавить инвентарь в зону.

Методы взаимодействия с таблицей блюд базы данных:

— получение все блюда;

— получить блюда по идентификатору;

— создать блюдо;

— обновить данные блюда;

— удалить блюдо.

Методы взаимодействия с таблицей пользователей базы данных:

— зарегистрировать пользователя;

— авторизовать пользователя;

— получить дынные всех пользователей;

— получить данные пользователя по идентификатору.

Методы взаимодействия с таблицей пакетов базы данных:

— получить данные всех пакетов;

— получить данные пакета по идентификатору;

— создать пакет;

— обновить пакет;

— удалить пакет.

Методы взаимодействия с таблицей зон базы данных:

— получить данные всех зон;

— получить данные зоны по идентификатору;

— создать зону;

— обновить зону;

— удалить зону.

Рисунок 3.1 – Программный интерфейс (часть 1)

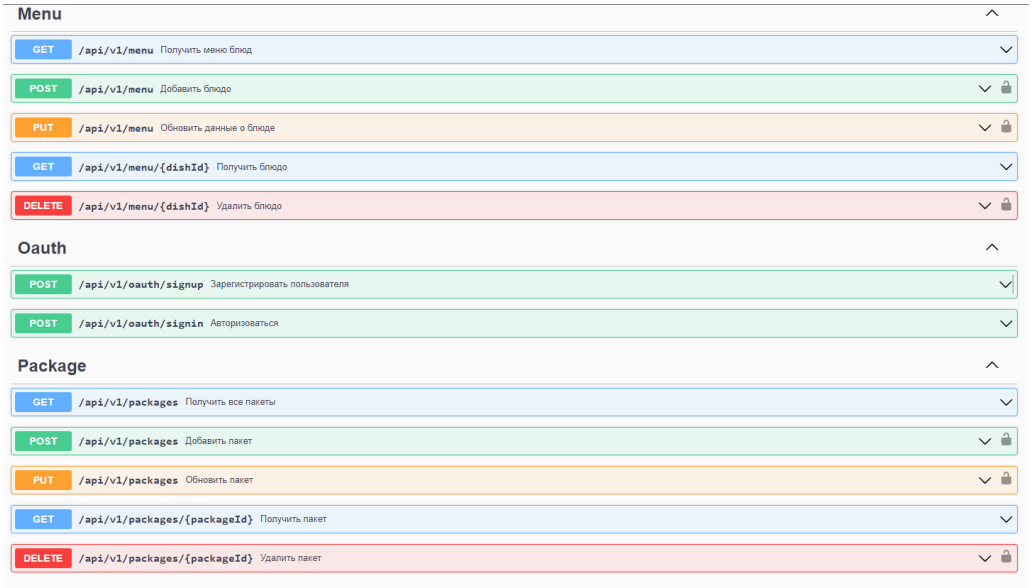

Рисунок 3.2 – Программный интерфейс (часть 2)

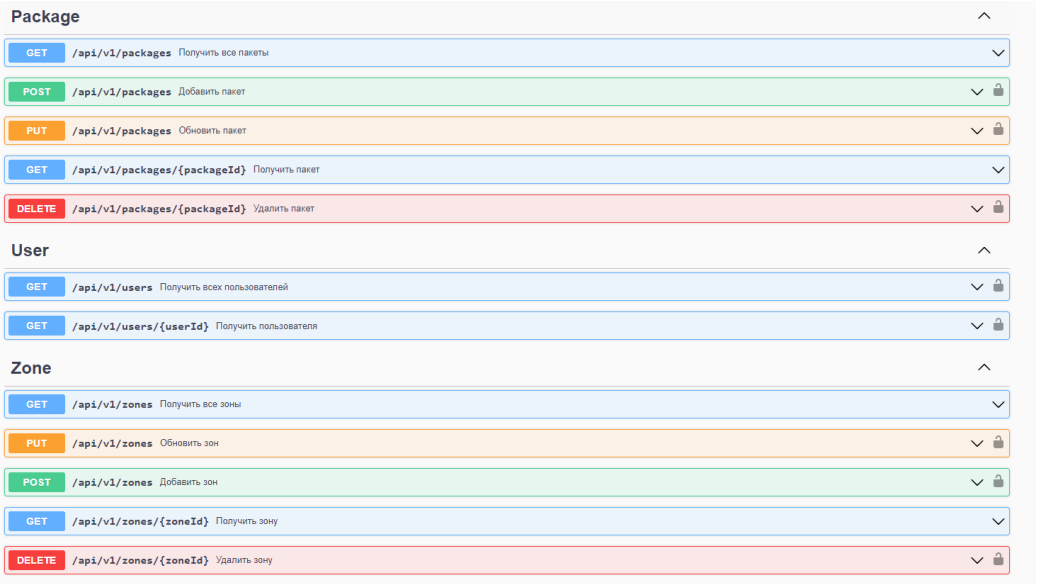

Рисунок 3.3 – Программный интерфейс (часть 3)

3.5 Демонстрация работы

На рисунке 3.4 показан процесс выполнения запроса, который направлен на получение всех доступных зон в базе данных. Этот запрос не предполагает наличие входных данных и возвращает JSON-объект с массивом данных о зонах. Информация о результатах запроса представлена в нижней части рисунке 3.4.

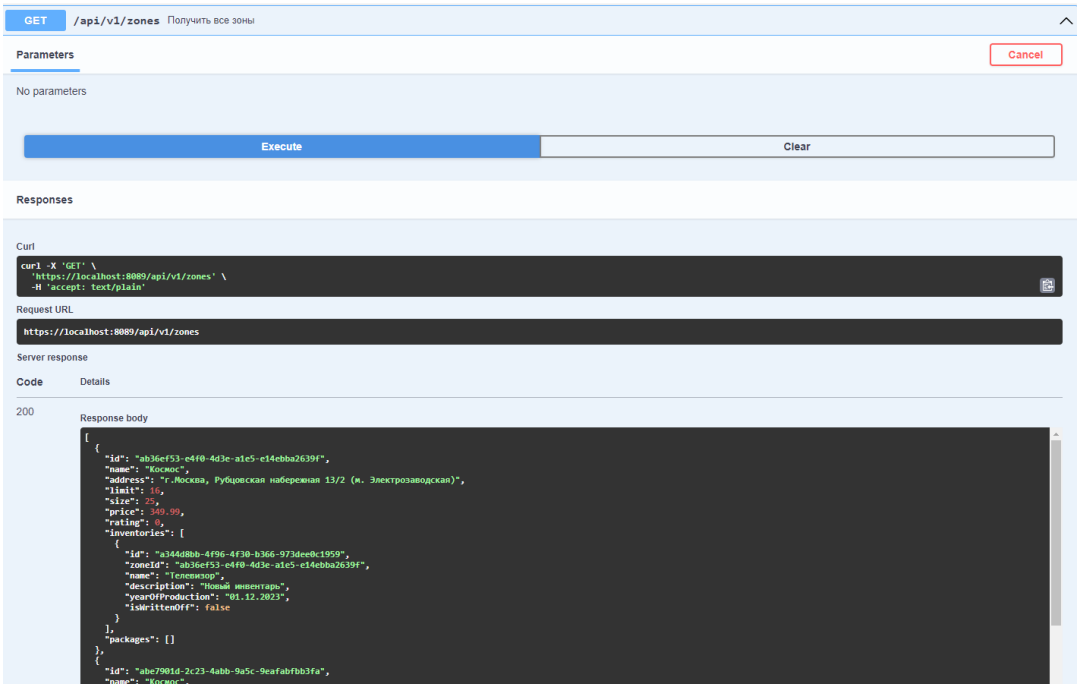

Рисунок 3.4 – Демонстрация работы программы

Вывод

В данном разделе была определена СУБД для работы с базой данных и средства реализации, а также выполнены следующие задачи: создание базы данных, таблиц, ролевой модели и триггера. Был описан разработанный пользовательский интерфейс и представлена демонстрация работы приложения.

4 Исследовательская часть

В данном разделе поставлена задача исследования зависимости времени выполнения запросов от объема данных в базе данных с наличием или без наличия в базе данных индекса для соответствующего атрибуты таблицы.

В дополнение к этому, представлены технические характеристики устройства, на котором были проведены измерения времени выполнения программного обеспечения, а также результаты измерений, отображенные в виде таблиц и графиков.

4.1 Технические характеристики

При проведении исследования системы важно учитывать характеристики устройства, на котором проводятся измерения, поскольку численные результаты могут различаться в зависимости от конкретных параметров данного устройства. Технические характеристики этого устройства, на котором проводилось исследование.

— Процессор: Intel(R) Core(TM) i5-10300H CPU 2.50 ГГц [17].
— Количество ядре: 4 физических и 8 логических ядер.
— Оперативная память: 16 ГБайт.
— Операционная система: Windows 11 Pro 64-разрядная система [18].

Исследование проводилось на стационарном компьютере. Во время тестирования устройство было нагружено только встроенными приложениями окружения.

4.2 Постановка задачи исследования

Индекс [19] в базе данных — это структура данных, которая помогает сократить время поиска запрошенных данных. Индексы достигают этого за

счет дополнительных затрат на хранение, память и поддержание их в актуальном состоянии, что приводит к повышению времени выполнения операций вставки, изменения и удаления.

Целью исследования является изучение зависимости времени выполнения запросов от объема данных в базе данных от наличия соответствующего индекса базы данных.

Для исследования были выбраны: сущность «зона», по той причине, что сущность имеет отношение многие ко многим к сущности «пакет» и один ко многим к сущности «инвентарь», а также сущности «брони», по той причине что она имеет связи один ко многим к сущностям «зона», «пакета», «пользователь». Таким образом для исследования были выбраны следующие запросы.

1) Запрос для получение всех данных о зонах, включая информацию о пакетах и инвентаре для конкретной зоны. Реализация данного запроса представлена на листинге 4.1.
2) Запрос на получение всех данных броней. Реализация данного запроса представлена на листинге 4.2.
3) Запрос на получение всех данных броней по идентификатору пользователя. Реализация данного запроса представлена на листинге 4.3.

Листинг 4.1 – Запрос на получение всех зон включая информацию пакетах и инвентаре

```
1 public Task<List<Zone>> GetAllZonesAsync(Role role)
2 {
3     return _contextFactory.GetDbContext(role).Zones
4     .Include(z => z.Inventories)
5     .Include(z => z.Packages).AsNoTracking()
6     .Select(z => ZoneConverter.ConvertDbModelToAppModel(z))
7     .ToListAsync();
8 }
```

Листинг 4.2 – Запрос на получение всех броней

```
public Task<List<Booking>> GetAllBookingAsync(Role role)
{
    return
    _contextFactory.GetDbContext(role)
    .Bookings
    .OrderBy(b => b.Date)
    .Select(b => BookingConverter.ConvertDbModelToAppModel(b))
    .ToListAsync();
}
```

Листинг 4.3 – Запрос на получение броней пользователя

```
public Task<List<Booking>> GetBookingByUserAsync(Guid userId,
    Role role)
{
    return _contextFactory.GetDbContext(role)
    .Bookings.Where(b => b.UserId == userId)
    .OrderBy(b => b.Date)
    .Select(b => BookingConverter.ConvertDbModelToAppModel(b))
    .ToListAsync();
}
```

Исследование проводилось для состояний таблицы зон содержащих от 10 до 10000 записей в таблице.

4.3 Сравнение времени обработки

В таблицах 4.1–4.2 продемонстрировано время выполнения запросов с использованием индекса и без него.

На рисунках 4.1–4.3 представлены графики зависимости времени выполнения от объема полученных результатов.

Таблица 4.1 – Зависимость времени выполнения запроса на получение данных зон

Количество записей в таблице зон	**Время, мс**	
	Без индексов	**С индексами**
10	6	6
50	22	18
100	35	35
500	178	176
1000	346	340
2000	845	639
3000	1257	1124
4000	1451	1402
5000	1789	1761
6000	2081	1988
7000	2441	2343
8000	2867	2772
9000	3126	3090
10000	3550	3400

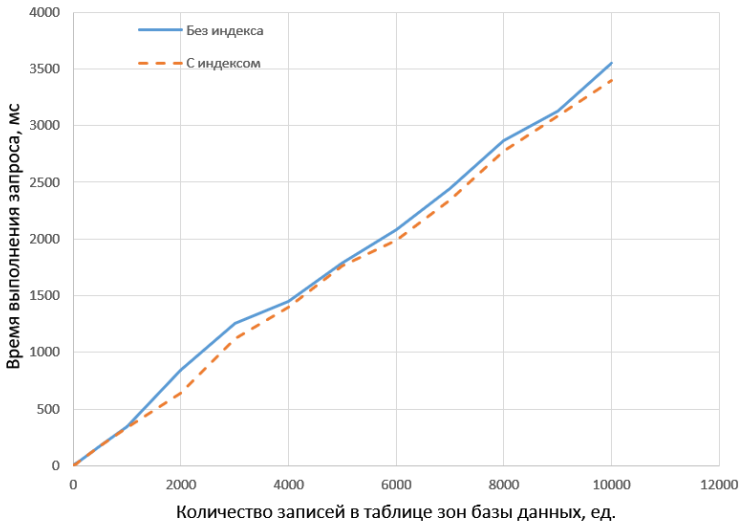

Рисунок 4.1 – График зависимости времени выполнения запроса на получение данных зон

Таблица 4.2 – Зависимость времени выполнения запроса от количества всех броней

Количество записей в таблице броней	**Время, мс**	
	Без индексов	**С индексами**
10	0.013	0.012
50	0.081	0.051
100	1.000	1.000
500	1.44	1.068
1000	1.52	1.110
2000	1.645	1.679
3000	2.304	2.331
4000	3.243	3.533
5000	4.241	4.233
6000	5.313	5.35
7000	6.256	6.166
8000	7.043	7.232
9000	10.125	10.33
10000	11.135	11.389

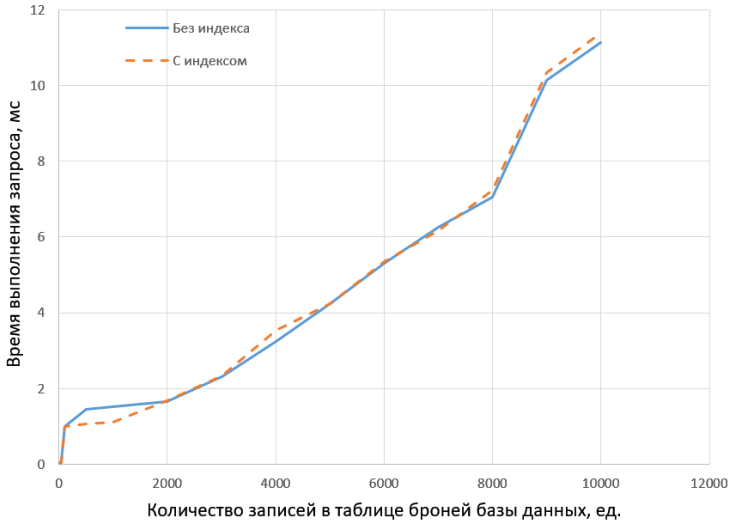

Рисунок 4.2 – График зависимости времени выполнения запроса от количества данных всех броней

Таблица 4.3 – Зависимость времени выполнения запроса от количества броней пользователя

Количество записей в таблице броней	**Время, мс**	
	Без индексов	**С индексами**
10	0.0001	0.00001
50	0.0002	0.0001
100	0.001	0.0001
500	0.005	0.001
1000	0.013	0.001
2000	0.01	0.002
3000	0.04	0.001
4000	0.06	0.004
5000	0.1	0.001
6000	0.14	0.01
7000	0.65	0.015
8000	0.74	0.0423
9000	0.9	0.064
10000	1.6	0.085

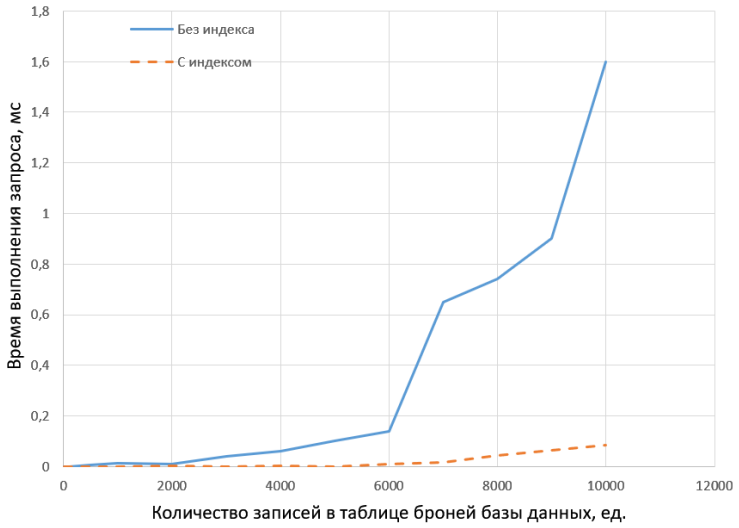

Рисунок 4.3 – График зависимости времени выполнения запроса от количества броней пользователя

Из результатов проведенного исследования следует, использование индекса существенно ускоряет выполнение запроса 3, который явно указывает значение искомого поля. Это приводит к значительному сокращению времени выполнения запроса, иногда более чем в десятки раз, но не всегда использование индекса приводит к таким результатам. Таким образом, время выполнения для запроса 1, где для зон включены данные пакетов и инвентаря, использование индекса снижает время выполнения в 1,15 раза. В случае запроса 2 различия значений времени выполнения с использованием индекса и без него схожи, из чего следует что в данном случае использование индекса не улучшает время запроса.

Вывод

В данном разделе было проведено исследование направленное на анализ влияния наличия индекса в базе данных на время выполнения запросов на получения данных из соответствующих таблиц базы данных.

Из результатов исследования следует, что воздействие индекса на время выполнения запроса в базе данных зависит от структуры самого запроса. Важно отметить, что использование индекса может как улучшать, так и ухудшать время выполнения, и также существуют сценарии, когда индекс не оказывает заметного влияния на время выполнения запроса. Важно отметить, что во всех случаях использование индекса приводит к увеличению объема занимаемой памяти и увеличивает время операций вставки, изменения и удаления данных.

ЗАКЛЮЧЕНИЕ

Цель, поставленная в начале работы, была достигнута. Кроме того были выполнены все поставленные задачи:

— проанализированы варианты представления данных и выбрать подходящий вариант для решения задачи;

— проведен анализ существующих способов хранения данных и системы управления базами данных, выбрать подходящую систему для поставленной цели;

— спроектирована база данных, описаны ее сущности и связи;

— реализовано программное обеспечение, позволяющее взаимодействовать со спроектированной базой данных;

— проведены исследования зависимости времени выполнения запроса от объема обрабатываемых данных.

Список использованных источников

1. Антикафе как новое пространство для культурно-досуговой деятельности и творческой реализации личноти [Электронный ресурс]. — Режим доступа: `https://cyberleninka.ru/article/n/` (дата обращения: 08.04.2023).

2. Party Hard [Электронный ресурс]. — Режим доступа: `https://hardparty.ru` (дата обращения: 08.04.2023).

3. Bizone Anticafe [Электронный ресурс]. — Режим доступа: `https://vbizone.ru` (дата обращения: 08.04.2023).

4. SpeedRent [Электронный ресурс]. — Режим доступа: `https://www.speedrent.ru` (дата обращения: 08.04.2023).

5. Дж. Дейт К. Введение в системы баз данных: 8-е издание. — «Вильямс», 2006. — С. 1328.

6. Д. Кузнецов С. Основы современных баз данных. — Центр Информационных Технологий, 1998.

7. П. Парфенов Ю. Постреляционные хранилища данных: учебное пособие. — Центр Информационных Технологий, 2016.

8. Анализ популярных реляционных систем управления базами данных (2022 г.) [Электронный ресурс]. — Режим доступа: `https://drach.pro/blog/hi-tech/item/196-popular-relational-dbms-2022` (дата обращения: 08.04.2023).

9. SQLite против MySQL против PostgreSQL: сравнение систем управления реляционными базами данных [Электронный ресурс]. — Режим доступа: `https://www.digitalocean.com/community/tutorials/` (дата обращения: 08.04.2023).

10. Краткий обзор языка C# [Электронный ресурс]. — Режим доступа: https://learn.microsoft.com/ru-ru/dotnet/csharp/tour-of-csharp/ (дата обращения: 08.04.2023).

11. Что такое .NET? Введение и обзор [Электронный ресурс]. — Режим доступа: https://learn.microsoft.com/ru-ru/dotnet/core/introduction (дата обращения: 08.04.2023).

12. Общие сведения о LINQ [Электронный ресурс]. — Режим доступа: https://learn.microsoft.com/ru-ru/dotnet/standard/linq/ (дата обращения: 08.04.2023).

13. Что такое Visual Studio [Электронный ресурс]. — Режим доступа: https://learn.microsoft.com/ru-ru/visualstudio/get-started/visual-studio-ide?view=vs-2019 (дата обращения: 08.04.2023).

14. Введение в NuGet [Электронный ресурс]. — Режим доступа: https://learn.microsoft.com/ru-ru/nuget/what-is-nuget (дата обращения: 08.04.2023).

15. Введение - Что такое Git? [Электронный ресурс]. — Режим доступа: https://book.git-scm.com/book/ru/v2/ (дата обращения: 08.04.2023).

16. Swashbuckle [Электронный ресурс]. — Режим доступа: https://learn.microsoft.com/ru-ru/aspnet/core/tutorials/getting-started-with-swashbuckle?view=aspnetcore-6.0&tabs=visual-studio (дата обращения: 08.04.2023).

17. Intel [Электронный ресурс]. — Режим доступа: https://ark.intel.com/content/www/ru/ru/ark/products/201839/intel-core-i510300h-processor-8m-cache-up-to-4-50-ghz.html (дата обращения: 03.9.2023).

18. Windows 10 Pro 2h21 64-bit [Электронный ресурс]. — Режим доступа: https://www.microsoft.com/ru-ru/software-download/windows10 (дата обращения: 03.09.2023).

19. Индексы, транзакции и уровни изоляции [Электронный ресурс]. — Режим доступа: https://proglib.io/p/ (дата обращения: 03.09.2023).

ПРИЛОЖЕНИЕ А

Презентация к курсовой работе

Презентация содержит 14 слайдов.