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 информации о прошедших Гран-При чемпионата «Формула-1»

Студент группы **ИУ7-61Б** _____ **Косарев А.А.**

Руководитель курсовой работы _____ **Строганов Д. В.**

2023 г.

РЕФЕРАТ

Расчетно-пояснительная записка 34 с., 14 рис., 7 источников, 1 прил.

БАЗЫ ДАННЫХ, МОДЕЛИ БАЗ ДАННЫХ, РОЛЕВАЯ СИСТЕМА, ФОРМУЛА 1, WEB-СЕРВЕР, API

Цель работы — разработка базы данных для хранения и обработки информации о прошедших Гран-При чемпионата «Формула-1».

В процессе работы были изучены существующие модели баз данных, спроектирована собственная база данных, соответствующая поставленной цели, и реализован Web-сервер на языке программирования Go с API для доступа и работы с информационной системой.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Современный мир невозможно представить без использования баз данных, которые являются основой для хранения, обработки и анализа больших объемов информации. В свою очередь, спортивные события также не обходятся без использования технологий и баз данных. Одним из самых популярных и зрелищных видов спорта является «Формула-1», которая с каждым годом стремительно увеличивает количество своих поклонников. В 2021 году число подписчиков Формулы 1 достигло 49.1 миллионов человек, при этом у нее самый высокий коэффициент вовлеченности по сравнению с другими крупными спортивными соревнованиями [1].

Россия не является исключением в росте любителей этого автоспорта. Большое количество людей читает новости, смотрит трансляции и следит за результатами Гран-При. Одним из основных источников информации о результатах прошедших этапов чемпионата был официальный сайт. Однако весной 2022 года «Формула-1» разорвала контракт с компанией «Росгонки» и в целом прекратила сотрудничество с Россией, поэтому сайт Формулы 1 стал недоступен в нашей стране. В связи с этим поклонники чемпионата потеряли возможность просмотра официальных трансляций и результатов прошедших Гран-При.

Учитывая все вышесказанное, можно сделать вывод о том, что создание информационной системы, посвященной данной теме, упростит получение информации о гонках, пилотах, трассах и командах чемпионата Формулы 1.

Цель данной курсовой работы — разработать базу данных для хранения и обработки информации о прошедших Гран-При чемпионата «Формула-1».

Для достижения поставленной в работе цели предстоит решить следующие задачи:

— проанализировать предметную область и существующие решения;

— формализировать информацию, которая будет храниться в базе данных;

— проанализировать существующие модели баз данных и СУБД и выбрать подходящую;

— спроектировать и разработать базу данных;

— спроектировать и разработать API для доступа к базе данных.

1 Аналитический раздел

1.1 Анализ предметной области

Каждый сезон чемпионата «Формула-1» состоит из отдельных этапов (гонок), каждый из которых длится 3 дня (пятница, суббота, воскресенье) и включает в себя 3 свободных практики (нужны для подбора оптимальных настроек для гонки), квалификацию и гонку. Результатом каждой свободной практики и квалификации является время самого быстрого круга, показанного в соответствующей сессии, а результатом гонки — позиция пилота и его отставание от лидера.

В данный момент в чемпионате участвуют 10 команд, в каждой из которых по 2 гонщика. Количество команд от сезона к сезону может меняться, но гонщиков в настоящее время всегда двое. По окончанию сезона гонщики могут переходить из одной команды в другую или вовсе покидать чемпионат.

1.2 Анализ существующих решений

Сейчас большинство ресурсов, предоставляющих результаты чемпионата «Формула-1», это сайты букмекерских контор, сайты с форумами и новостные сайты, например:

— `https://www.championat.com/`;

— `https://ru.motorsport.com/`;

— `https://www.f1-portal.ru/`;

— `https://f1flow.ru/`;

— `https://www.f1news.ru/`.

Некоторые из подобных сайтов специализируются не только на Формуле 1, поэтому они переполнены информацией о других спортивных мероприятиях, что может быть неудобным для тех, кто наблюдает только за Формулой. Другая же часть сайтов, которые нацелены на Формулу 1, содержит

уже больше информации о чемпионате, а также ленту новостей, связанных с миром автоспорта,. Однако на таких сайтах довольно сложно «добраться» до самих результатов Гран-При.

1.3 Формулировка требований к разрабатываемой базе данных

В рамках поставленной цели необходимо разработать базу данных, содержащую информацию о результатах Гран-При чемпионата «Формула-1», гонщиках, командах и трассах, а также API для доступа к данным и последующей возможности создания веб-приложения о Формуле 1.

API должно предоставлять возможность получения основной информации о гонке: победитель, время самого быстрого круга, результаты квалификации, пилот, взявший поул (первое место в квалификации) и так далее. Необходимо обеспечить возможность добавления и удаления/изменения результатов гонки, новых пилотов и команд.

1.4 Формализация информации, хранимой в базе данных

Для хранения в базе данных информацию о Гран-При следует разбить на следующие сущности: Гран-При, результат квалификации, результат гонки, гонщик, команда, трек и результаты чемпионата. Сущность Гран-При связана с треком связью «один к одному», а с результатами квалификации и гонки — «один ко многим», которые в свою очередь связаны с моделями гонщиков и команд связью «один к одному». Так как гонщики могут переходить из одной команды в другую, между соответствующими сущностями следует установить связь «многие ко многим».

На рисунке 1.1 изображена ER-диаграмма сущностей системы в нотации Чена.

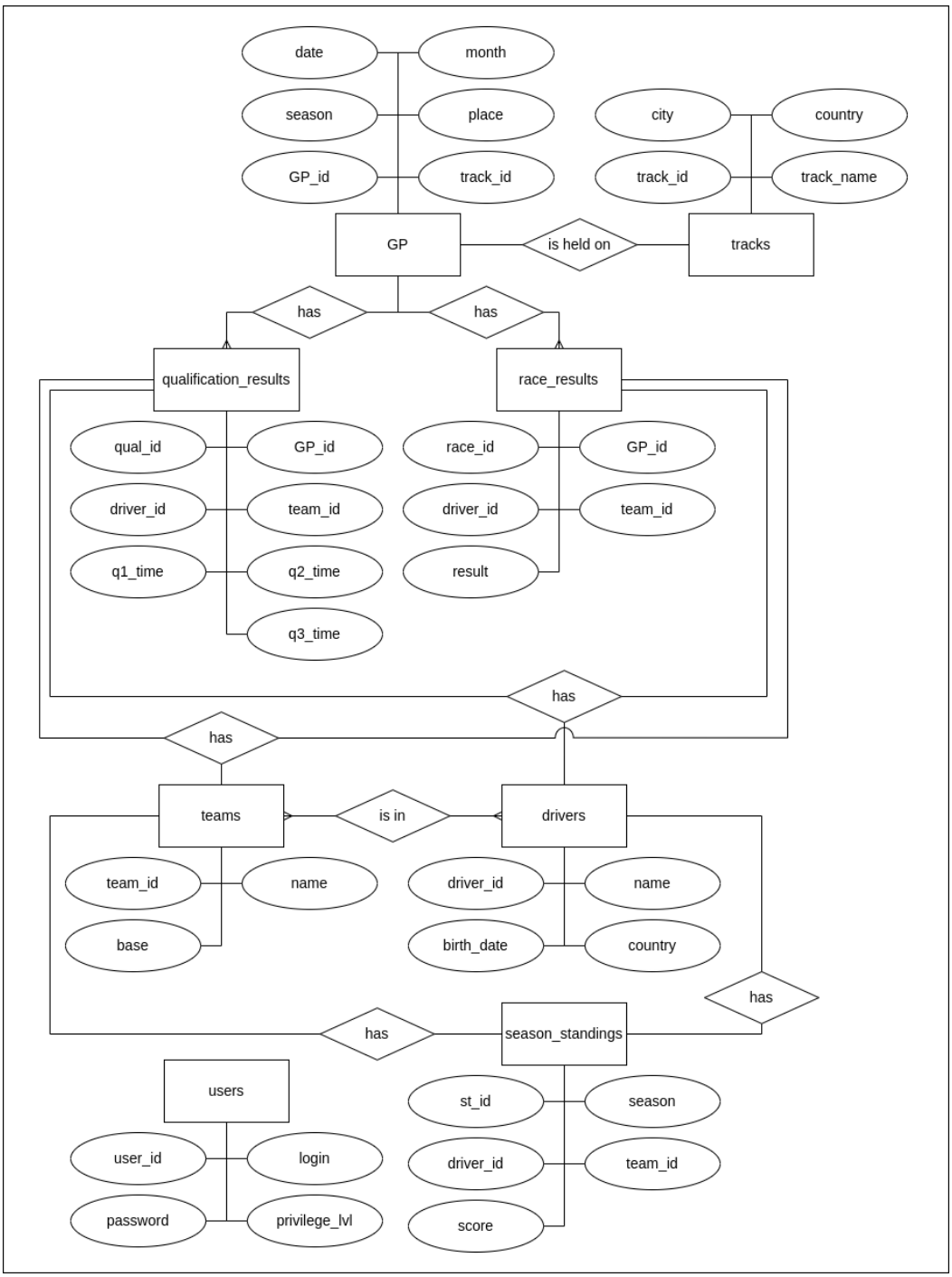

Рисунок 1.1 – ER-диаграмма для базы данных

1.5 Система ролей

Ролевая модель состоит из 3 категорий пользователей:

— гость (неавторизованный пользователь);

— авторизованный пользователь;

— администратор.

Гость — это неавторизованный пользователь, для которого количество доступной информации ограничено. Он может просмотреть победителя любой гонки, данные о Гран-При (кроме временных результатов гонки и квалификации), а также информацию о гонщиках текущего сезона. На рисунке 1.2 представлена use-case диаграмма для данной роли.

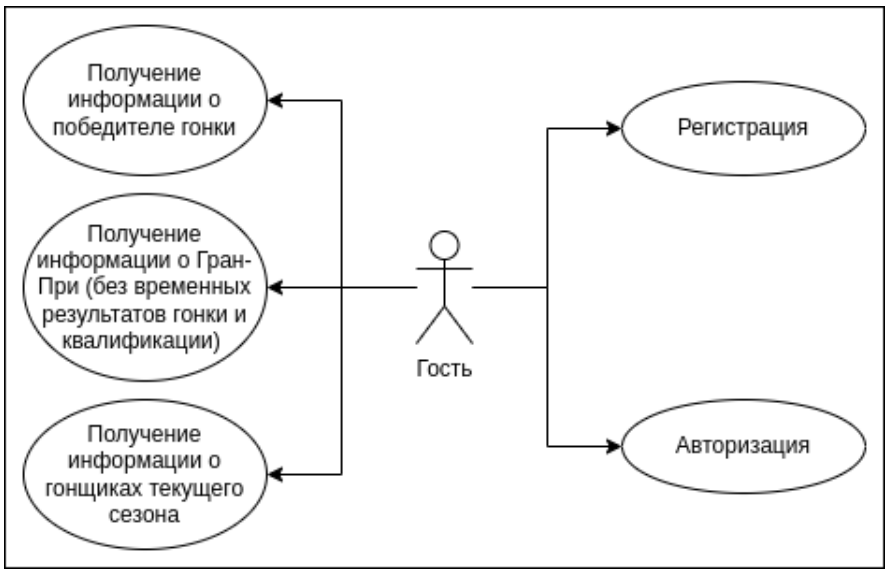

Рисунок 1.2 – Use-case диаграмма для роли «гость»

Авторизованный пользователь может получить любую информацию из базы данных: все данные о результатах гонок, о всех пилотах, командах и трассах. На рисунке 1.3 представлена соответствующая use-case диаграмма.

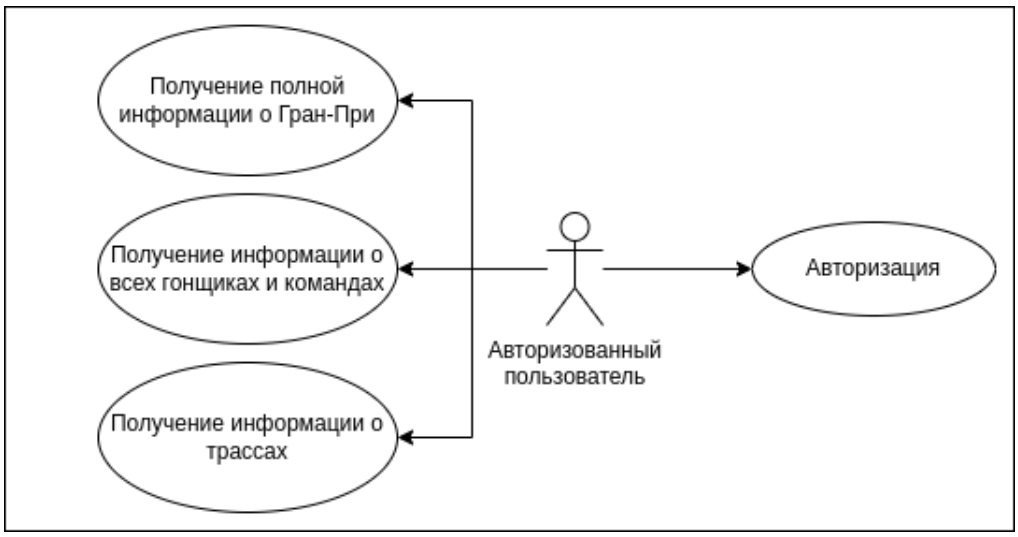

Рисунок 1.3 – Use-case диаграмма для роли «авторизованный пользователь»

Администратор — авторизованный пользователь, обладающий правами на изменение базы данных. Он может добавлять результаты очередного

этапа чемпионата, изменять и удалять данные прошлых Гран-При, а также информацию о гонщиках, командах и трассах. На рисунке 1.4 представлена use-case диаграмма для роли администратора.

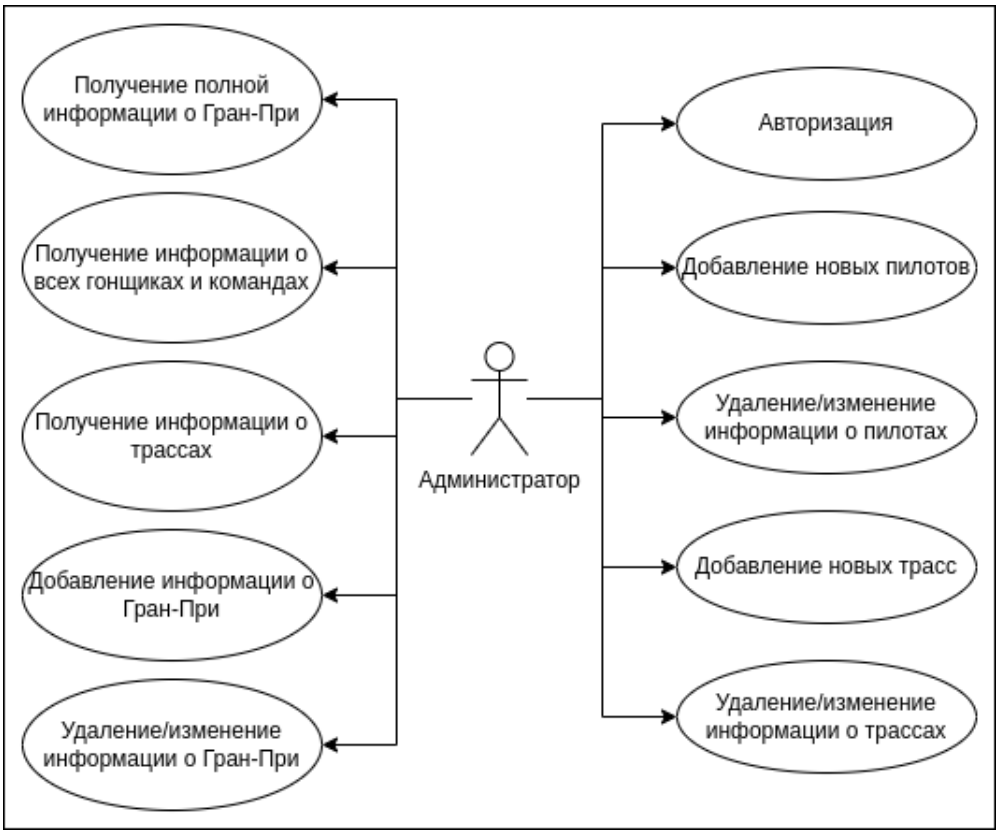

Рисунок 1.4 – Use-case диаграмма для роли «администратор»

1.6 Выбор модели базы данных

По модели хранения данных системы управления базами данных (СУБД) классифицируют по трем типам:

— дореляционные;

— реляционные;

— постреляционные.

Дореляционные СУБД

Дореляционные СУБД не используют реляционную модель данных. Вместо этого они могут использовать иерархическую модель, сетевую модель или объектно-ориентированную модель данных. Данные в дореляционных СУБД могут быть представлены в виде деревьев, графов или объектов.

Примеры дореляционных СУБД:

— IMS (Information Management System) — система управления данными от IBM, использующая иерархическую модель данных;

— IDMS (Integrated Database Management System) — система управления данными от CA Technologies, использующая сетевую модель данных;

— MongoDB — документо-ориентированная СУБД, которая использует объектно-ориентированную модель данных.

Преимущества дореляционных СУБД:

— высокая производительность при работе с большими объемами данных;

— гибкость в организации данных, так как они могут быть представлены в различных форматах;

— возможность работы с неструктурированными данными.

К недостаткам дореляционных СУБД можно отнести ограниченные возможности для анализа и обработки данных и сложность в создании связей между данными.

Реляционные СУБД

Реляционные СУБД [2] используют реляционную модель данных. Данные хранятся в таблицах, состоящих из строк и столбцов, а для управления данными применяется SQL (Structured Query Language).

Примеры реляционных СУБД:

— Oracle — одна из самых популярных реляционных СУБД, используемая в крупных корпоративных системах;

— MySQL — реляционная СУБД, используемая в веб-приложениях и других проектах с открытым исходным кодом;

— Microsoft SQL Server — реляционная СУБД от Microsoft, используемая в Windows-среде.

Преимущества реляционных СУБД:

— простота в использовании и понимании;

— широкая поддержка со стороны сообщества разработчиков;

— высокая надежность и безопасность данных.

Недостатками реляционных СУБД являются ограниченные возможности для работы с неструктурированными данными и сложность в масштабировании системы при работе с большими объемами данных.

Постреляционные СУБД

Постреляционные СУБД [3] расширяют возможности реляционных СУБД, добавляя новые функции и возможности. Они могут использовать NoSQL-технологии, такие как графовые базы данных или колоночные базы данных.

Примеры постреляционных СУБД:

— Cassandra — колоночная база данных, используемая для хранения и обработки больших объемов данных;

— Neo4j — графовая база данных, используемая для хранения и обработки связанных данных;

— HBase — база данных, используемая для хранения и обработки больших объемов структурированных и неструктурированных данных.

Преимущества постреляционных СУБД:

— широкие возможности для работы с неструктурированными данными;

— высокая производительность при работе с большими объемами данных;

— гибкость в организации данных.

Недостатки постреляционных СУБД — это сложность в использовании и понимании и ограниченная поддержка со стороны сообщества разработчиков.

Вывод

В этом разделе была проанализирована предметная область и существующие решения в сфере ресурсов, предоставляющих результаты этапов чемпионата «Формула-1». Также была проведена формализация информации, которая будет храниться в базе данных, и проработана ролевая система пользователей. Для достижения поставленной цели была выбрана реляционная модель базы данных, так как данные о гонках являются структурированными и имеют большое количество связей.

2 Конструкторский раздел

2.1 Проектируемая база данных

На рисунке 2.1 представлена диаграмма проектируемой базы данных.

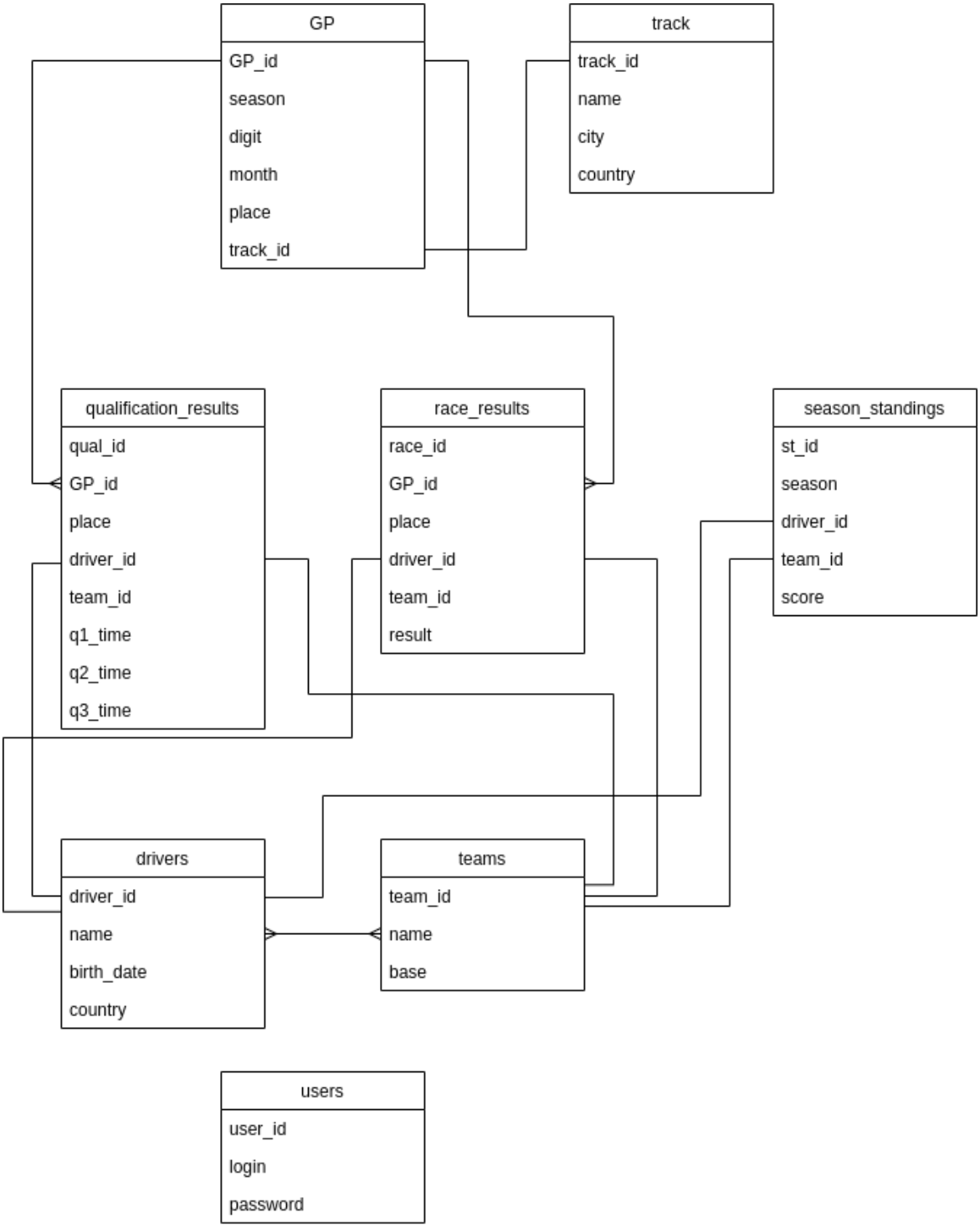

Рисунок 2.1 – Диаграмма проектируемой базы данных

В таблицах 2.1–2.8 указано описание полей соответствующих таблиц базы данных.

Таблица 2.1 – Описание таблицы gp

Поле	Описание
gp_id	Первичный ключ таблицы
season	Сезон чемпионата (год проведения)
digit	Число проведения
month	Месяц проведения
place	Место проведения
track_id	Id трека

Таблица 2.2 – Описание таблицы qualification_results

Поле	Описание
qual_id	Первичный ключ таблицы
gp_id	Id соответствующего Гран-При
place	Занятое в квалификации место
driver_id	Id гонщика
team_id	Id команды
q1_time	Время первого сегмента квалификации
q2_time	Время второго сегмента квалификации
q3_time	Время третьего сегмента квалификации

Таблица 2.3 – Описание таблицы race_results

Поле	Описание
race_id	Первичный ключ таблицы
gp_id	Id соответствующего Гран-При
place	Занятое в гонке место
driver_id	Id гонщика
team_id	Id команды

Таблица 2.4 – Описание таблицы drivers

Поле	Описание
driver_id	Первичный ключ таблицы
name	Имя гонщика
birth_date	Дата рождения гонщика
country	Страна, за которую выступает гонщик

Таблица 2.5 – Описание таблицы teams

Поле	Описание
team_id	Первичный ключ таблицы
name	Название команды
base	Место нахождения базы команды

Таблица 2.6 – Описание таблицы teams_drivers

Поле	Описание
td_id	Первичный ключ таблицы
driver_id	Id гонщика
team_id	Id команды
year	Год, в котором гонщик выступал за команду

Таблица 2.7 – Описание таблицы season_standings

Поле	Описание
st_id	Первичный ключ таблицы
driver_id	Id гонщика
team_id	Id команды
score	Количество очков в чемпионате

Таблица 2.8 – Описание таблицы users

Поле	Описание
user_id	Первичный ключ таблицы
login	Логин пользователя
password	Пароль пользователя
role	Роль пользователя

2.2 Описание используемого триггера

Для автоматического обновления таблицы личного зачета пилотов предусмотрен триггер, который при добавлении в таблицу результатов гонки новых записей будет пересчитывать баллы гонщиков в личном зачете. На рисунке 2.2 изображена схема функции триггера.

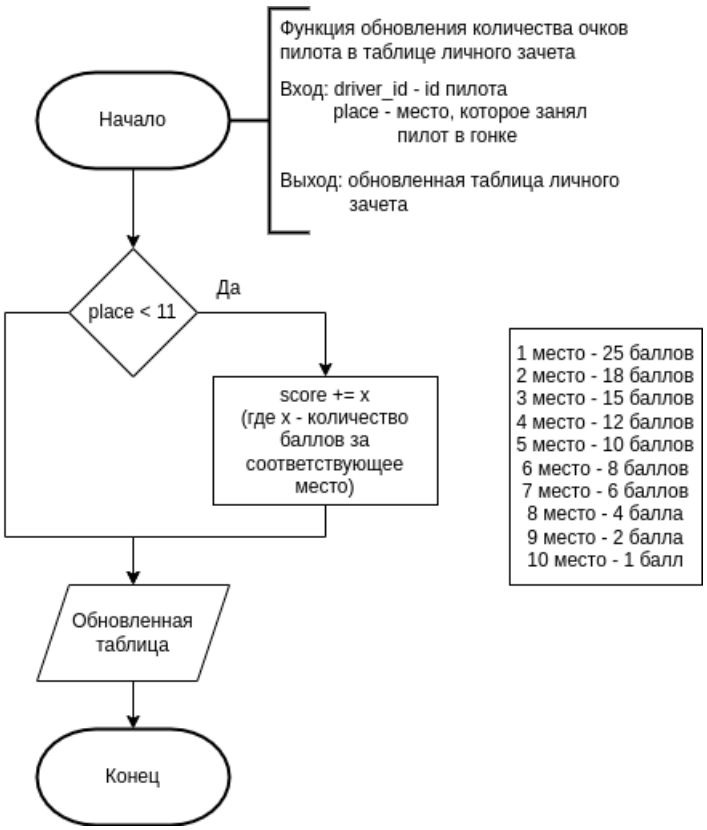

Рисунок 2.2 – Схема функции триггера для обновления таблицы личного зачета

Для работы триггера понадобятся следующие функции:

— GetScore — возвращает необходимое количество баллов по заданному месту пилота в результатах гонки;

— GetSeason — возвращает год, в котором проводилась гонка, по ID записи, добавленной в таблицу результатов гонки;

— UpdateTrigger — обновляет таблицу личного зачета чемпионата при добавлении новой записи в таблицу результатов гонки.

2.3 Система ролей в базе данных

Для работы с базой данных было предусмотрено создание трех ролей: гость, авторизованный пользователь и администратор.

Гость имеет доступ к просмотру информации в следующих объектах базы данных:

— представление (view) таблицы race_results, в которую вынесена только информация о победителях гонки;

— таблица grandprix;

— представление (view) таблицы drivers, содержащую только гонщиков текущего сезона.

Авторизованный пользователь может получать любую информацию из любой таблицы в базе данных, но не может изменять их.

Администратор имеет права на чтение и изменение всех таблиц. Он может добавлять результаты гонок, изменять информацию о пилотах и командах, добавлять и удалять информацию о пользователях и так далее.

3 Технологический раздел

3.1 Технологии, используемые при создании ПО

Для написания Web-сервера был выбран язык программирования Go[4]. Причины использования данного языка описаны ниже.

— Простота и легкость в изучении: Go имеет простой и понятный синтаксис, который позволяет быстро освоить язык программирования.

— Высокая производительность: Go обладает низким уровнем абстракции, что позволяет создавать высокопроизводительные приложения.

— Большая экосистема: Go имеет обширную библиотеку стандартных пакетов, а также множество сторонних библиотек и фреймворков.

— Надежность: Go был создан Google для разработки высоконагруженных приложений, поэтому он обладает высокой надежностью и стабильностью.

Для реализации использовался Web-фреймворк gorilla/mux[5], так как:

— является одним из самых часто используемых пакетов для создания Web-серверов;

— имеет простой синтаксис;

— позволяет определять более сложные маршруты, которые могут использовать регулярные выражения.

В качестве СУБД был выбран PostgreSQL[6] по причине существования опыта работы с ним.

3.2 Архитектура ПО

В процессе разработки были реализованы следующие структуры:

— Driver — описывает информацию о пилоте;

— Team — описывает информацию о команде;

— GrandPrix — описывает информацию о Гран-При;

— QualResult — описывает информацию о результате квалификации

— RaceResult — описывает информацию о результате гонки;

— Track — описывает информацию о треке;

— User — описывает информацию о пользователе;

Для каждой модели реализованы следующие компоненты:

— Repository — класс для доступа к данным в базе данных;

— Usecase — класс, реализующий основную логику приложения;

— Handler — класс, позволяющий обрабатывать HTTP-запросы.

Также были созданы две middleware-функции, выполняемые перед каждым запросом, LogMiddleware и AuthMiddleware: первая отвечает за логирование, вторая — за проверку наличия у пользователя роли, необходимой для выполнения отправленного запроса.

3.3 Реализация триггера

В листинге 3.1–3.2 представлена реализация триггера для обновления таблицы личного зачета пилотов.

Листинг 3.1 – Исходный код триггера и его функций

```
-- Get score number
create or replace function GetScore(place int)
returns int
as $$
begin
    if place = 1 then
        return 25;
    elsif place = 2 then
        return 18;
    elsif place = 3 then
        return 15;
    elsif place = 4 then
        return 12;
    elsif place = 5 then
        return 10;
    elsif place = 6 then
        return 8;
    elsif place = 7 then
        return 6;
    elsif place = 8 then
        return 4;
    elsif place = 9 then
        return 2;
    elsif place = 10 then
        return 1;
    else return 0;
    end if;
end
$$ language plpgsql;

-- Get season by race id
create or replace function GetSeason(id int)
returns int
as $$
declare res int;
begin
    select gp_season
    into res
    from raceresults r
    join grandprix g on g.gp_id = r.gp_id
```

Листинг 3.2 – Исходный код триггера и его функций (продолжение)

```
    where race_id = id;
    return res;
end
$$ language plpgsql;


-- Trigger function
create or replace function UpdateTrigger()
returns trigger
as $$
begin
    if GetSeason(new.race_id) = 2022 then
        update season_standings
        set score = score + GetScore(new.race_driver_place)
        where driver_id = new.driver_id;
    end if;
    return new;
end
$$ language plpgsql;

-- Trigger
create trigger update_season_standing
after insert on raceresults
for each row
execute procedure UpdateTrigger();
```

3.4 Реализация системы ролей

В листинге 3.3–3.4 содержится SQL код, который создает 3 вида ролей, задает для них пароли и выдает им права на определенные таблицы в базе данных.

Листинг 3.3 – Система ролей

```
create user "default_guest";
create user "default_user";
create user "default_admin";
```

Листинг 3.4 – Система ролей (продолжение)

```
alter role "default_guest" password '11111111';
alter role "default_user" password '12344321';
alter role "default_admin" password '12345678';

grant select on table grandprix to "default_guest";
grant select on race_results_view to "default_guest";
grant select on drivers_of_season to "default_guest";

grant select on table drivers to "default_user";
grant select on table grandprix to "default_user";
grant select on table qualificationresults to "default_user";
grant select on table raceresults to "default_user";
grant select on table season_standings to "default_user";
grant select on table teams to "default_user";
grant select on table teamsdrivers to "default_user";
grant select on table tracks to "default_user";
grant select on race_results_view to "default_user";
grant select on drivers_of_season to "default_user";

alter role "default_admin" superuser;
```

3.5 Примеры работы

Для примера работы информационной системы были выполнены следующие запросы:

— регистрация пользователя;

— авторизация пользователя;

— получение результатов гонки определенного Гран-При;

— добавление администратором новой записи в таблицу треков;

— попытка добавления новой записи в таблицу треков обычным пользователем.

На рисунках 3.1–3.5 представлены примеры запросов к серверу и результаты их обработки.

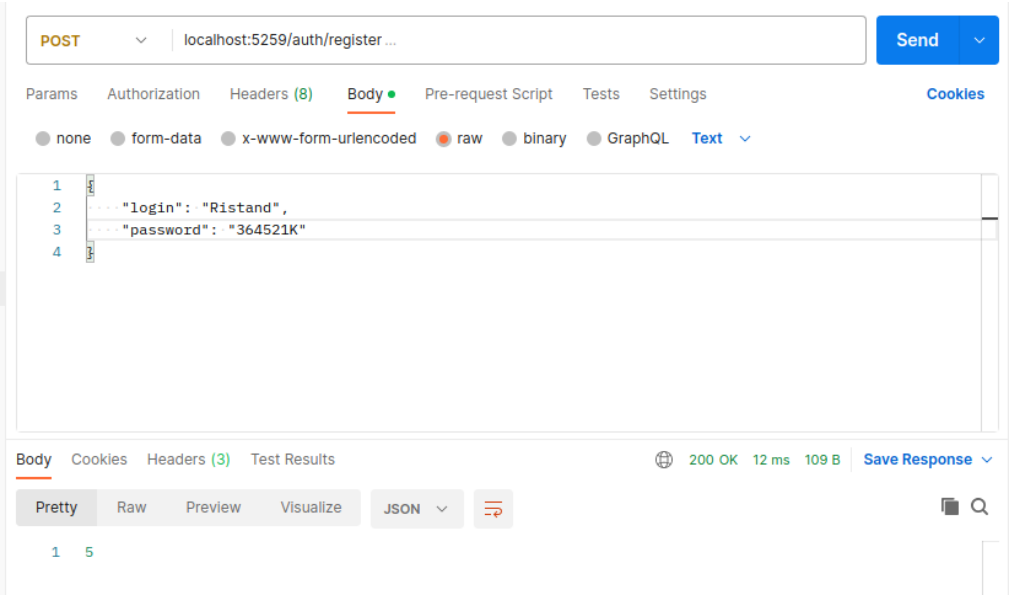

Рисунок 3.1 – Регистрация пользователя

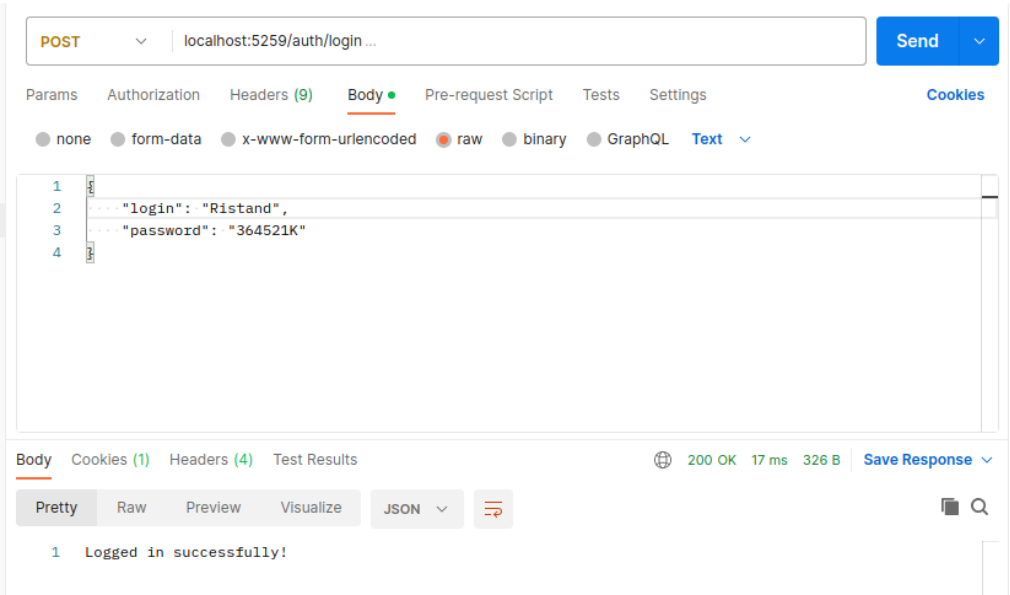

Рисунок 3.2 – Авторизация пользователя

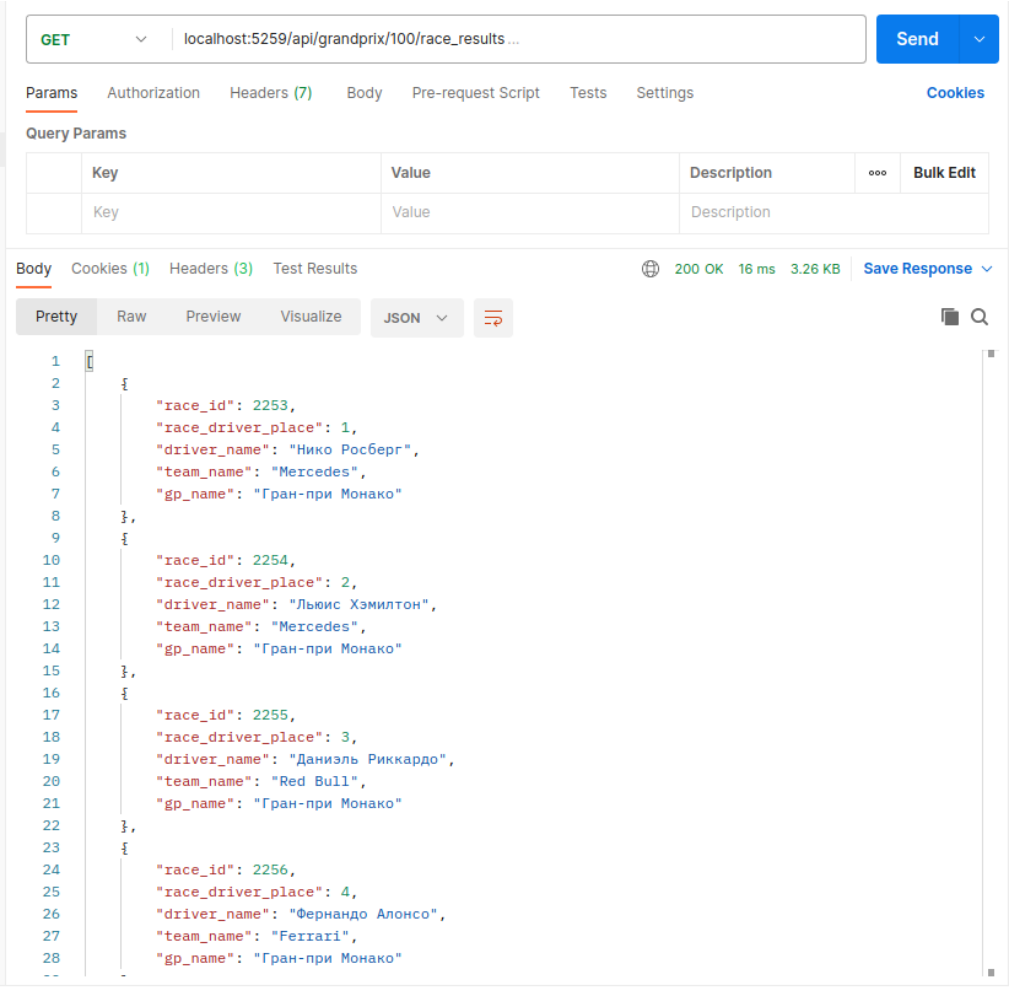

Рисунок 3.3 – Пример Get-запроса

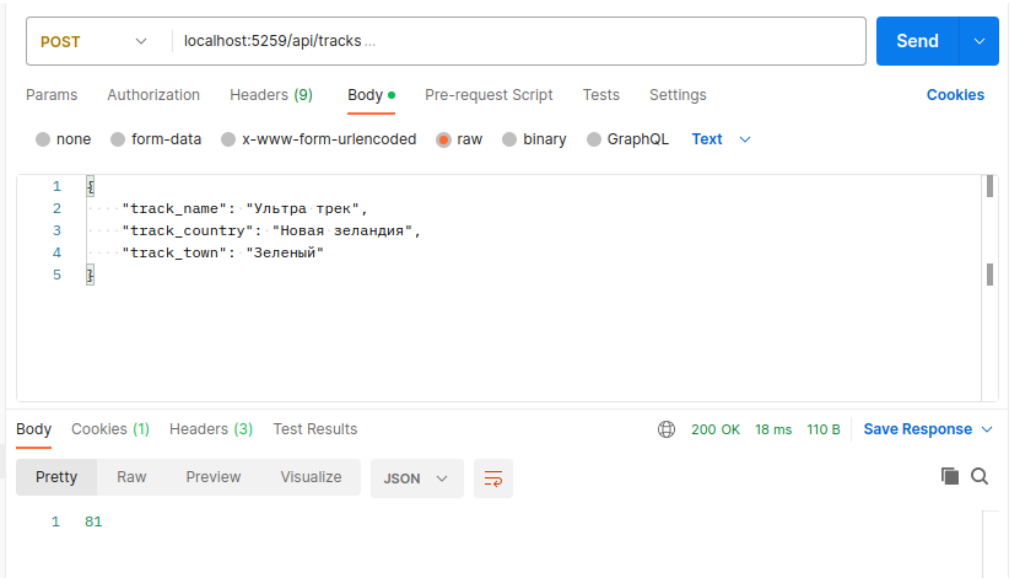

Рисунок 3.4 – Пример Post-запроса администратором для создания сущности

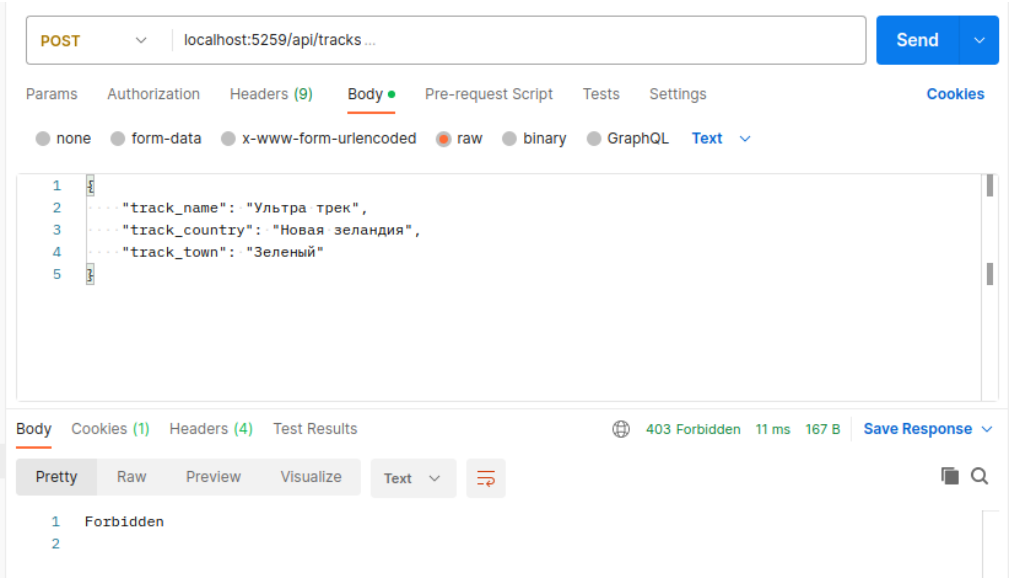

Рисунок 3.5 – Пример Post-запроса обычным пользователем для создания сущности

4 Исследовательский раздел

4.1 Эксперимент

В качестве исследования было проведено 3 эксперимента для анализа зависимости времени ответа сервера от количества пользователей с использованием инструмента для нагрузочного тестирования Locust[7], который позволяет задать сценарий обращения пользователя к серверу.

Тест №1

Пользовательский сценарий:

— POST(«/auth/login») - 1 запрос;

— GET(«/api/race_results») - 3 запроса;

— DELETE(«/auth/logout») - 1 запрос.

В первом тесте количество пользователей равномерно увеличивалось до 1000 (каждую секунду добавлялось 15 пользователей). Таблица, к которой выполнялся запрос, содержит 5849 строк. На рисунке 4.1 показан результат эксперимента.

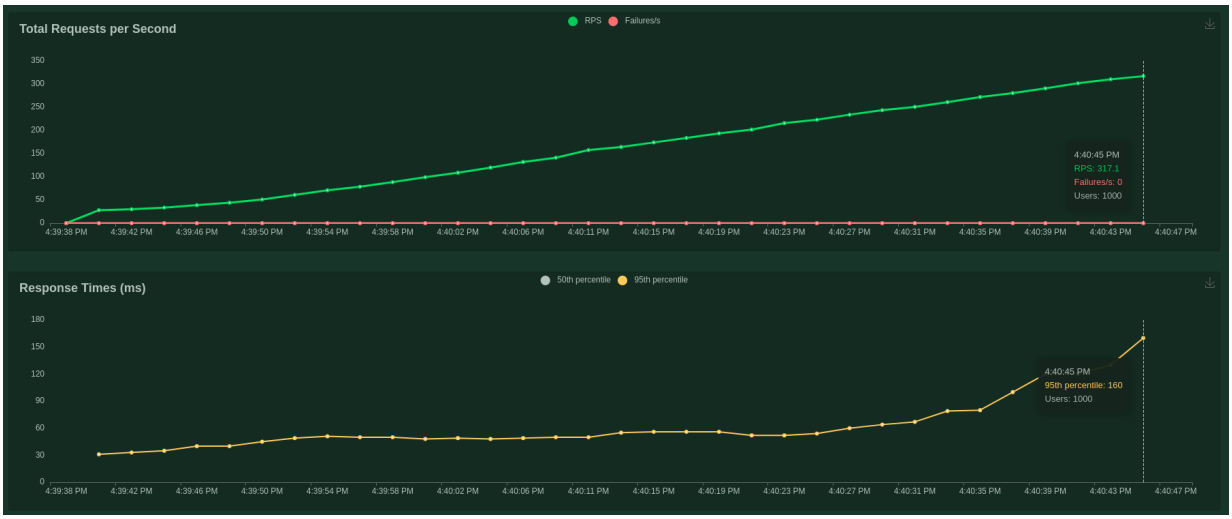

Рисунок 4.1 – Графики количества запросов в секунду и времени ответа сервера при росте количества пользователей для теста №1

Тест №2

Пользовательский сценарий:

— POST(«/auth/login») - 1 запрос;

— GET(«/api/grandprix») - 3 запроса;

— DELETE(«/auth/logout») - 1 запрос.

Во втором тесте количество пользователей также увеличивалось до 1000, как и в первом тесте. Таблица, к которой выполнялся запрос, содержит 277 строк. На рисунке 4.1 показан результат эксперимента.

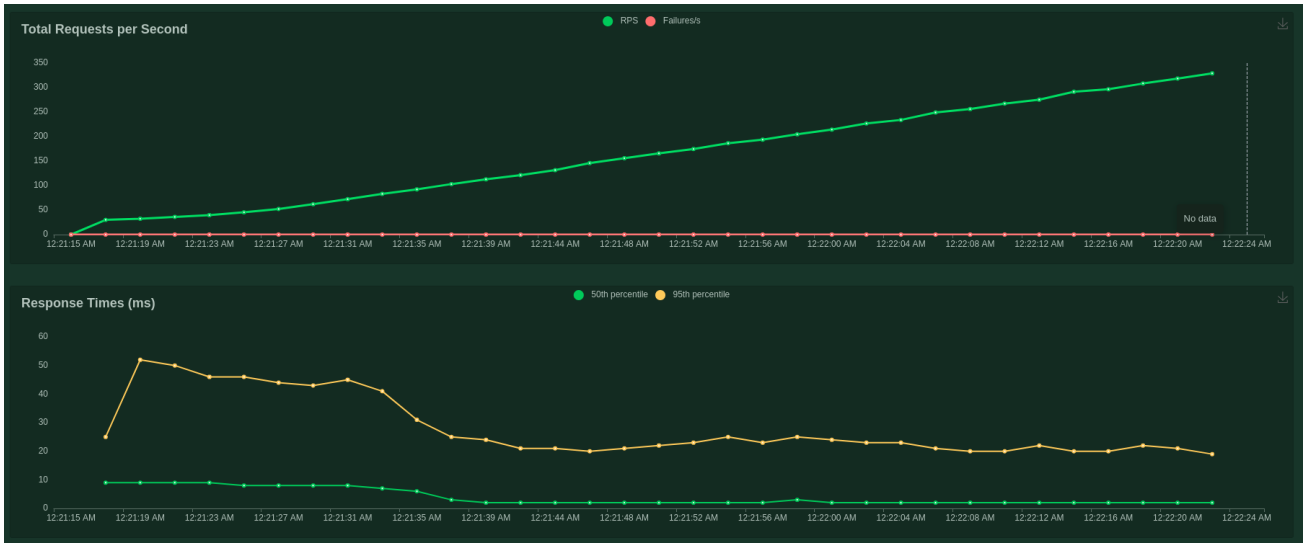

Рисунок 4.2 – Графики количества запросов в секунду и времени ответа сервера при росте количества пользователей для теста №2

Тест №3

Пользовательский сценарий:

— POST(«/auth/login») - 1 запрос;

— GET(«/api/grandprix») - 2 запроса;

— GET(«/api/drivers_of_season») - 1 запроса;

— GET(«/api/grandprix/season/2010») - 2 запроса;

— GET(«/api/grandprix/id/race_winner») - 3 запроса;

— GET(«/api/race_results») - 3 запроса;

— DELETE(«/auth/logout») - 1 запрос.

В данном тесте количество пользователей увеличивалось до 4000. На рисунке 4.3 показан результат эксперимента.

Рисунок 4.3 – Графики количества запросов в секунду и времени ответа сервера при росте количества пользователей для теста №3

4.2 Вывод

По результатам экспериментов можно выделить следующие аспекты:

— при обращении к таблице меньшего размера grandprix время ответа сервера в среднем составляет 30 миллисекунд, а при обращении к таблице race_results, количество строк в которой примерно в 21 раз больше, минимальное время ответа 30 миллисекунд, а максимальное — 160 миллисекунд, что в 5 раз больше, чем при получении всех записей из таблицы grandprix;

— максимальное количество запросов в секунду, которое может обработать сервер, примерно равно 560 запросам, и при достижении такого значения начинает стремительно увеличиваться время ответа сервера, что и отображено на рисунке 4.3;

— по графикам 4.1–4.3 видно, что в результате нагрузочного тестирования все запросы были выполнены корректно, не возникло ни одной ошибки.

ЗАКЛЮЧЕНИЕ

Цель, которая была поставлена в начале курсовой работы, была достигнута: разработана база данных для хранения и обработки информации о прошедших Гран-При чемпионата «Формула-1».

Решены все поставленные задачи:

— проанализирована предметная область и существующие решения;

— формализирована информация, которая будет храниться в базе данных;

— проанализированы существующие модели баз данных и СУБД и выбрана подходящая;

— спроектирована и разработана база данных;

— спроектировано и разработано API для доступа к базе данных.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Статистика Формулы 1 по итогам прошлого сезона - все новости Формулы 1 2023 [Электронный ресурс]. — URL: `https://www.f1news.ru/news/f1-158869.html` (дата обращения: 13.04.2023)

2. Бородина А. И. Реляционная модель данных [Электронный ресурс]. — URL: `http://bseu.by/it/tohod/lekcii2_3.htm` (дата обращения: 13.04.2023).

3. Бородина А. И. Постреляционная, многомерная и объектноориентированная модели данных [Электронный ресурс]. — URL: `http://bseu.by/it/tohod/lekcii2_4.htm` (дата обращения 13.04.2023).

4. The Go Programming Language [Электронный ресурс]. — URL: `https://go.dev/` (дата обращения 15.05.2023).

5. mux package - github.com/gorilla/mux - Go Packages [Электронный ресурс]. — URL: `https://pkg.go.dev/github.com/gorilla/mux` (дата обращения 15.05.2023).

6. The PostgreSQL Global Development Group [Электронный ресурс]. — URL: `https://www.postgresql.org/` (дата обращения 15.05.2023).

7. Locust - A modern load testing tool [Электронный ресурс]. — URL: `https://locust.io/` (дата обращения 28.05.2023).

ПРИЛОЖЕНИЕ А Презентация курсовой работы

Презентация курсовой работы содержит 12 слайдов, на которых представлено краткое описание курсовой работы.