Министерство науки и высшего образования РФ Федеральное государственное автономное образовательное учреждение высшего образования

«Национальный исследовательский технологический университет «МИСИС» Институт ИКН

Кафедра инфокоммуникационных технологий

Курсовая работа

Разработка клиент-серверных приложений

Создание базы данных PostgreSQL с интеграцией TypeORM

Выполнил студент группы БИВТ-21-10,

Зиядинов А.Р

Проверил:

Рзазаде Ульви Азар Оглы

Москва, 2023

Table of Contents

**No table of contents entries found.**

# 1. Предметная область

Тема "Создание базы данных PostgreSQL с интеграцией TypeORM" относится к области разработки программного обеспечения и баз данных. В данной предметной области осуществляется проектирование и реализация базы данных с использованием PostgreSQL, одной из популярных реляционных систем управления базами данных (СУБД).

TypeORM, в свою очередь, является ORM (Object-Relational Mapping) фреймворком для Node.js, который предоставляет средства для удобного взаимодействия с базой данных, включая создание моделей и сущностей, выполнение запросов и миграций.

Проблема(ы) в данной предметной области:

1. Сложность проектирования и создания эффективной базы данных: Создание хорошо структурированной базы данных, которая соответствует требованиям приложения и обеспечивает эффективное хранение и доступ к данным, является непростой задачей. Неправильное проектирование базы данных может привести к проблемам с производительностью и масштабируемостью.

2. Управление связями между сущностями: При проектировании базы данных часто возникает необходимость управлять связями между различными сущностями и их атрибутами. Это может включать связи один-к-одному, один-ко-многим и многие-ко-многим, а также работу с внешними ключами и целостностью данных.

3. Миграции базы данных: При разработке и обновлении приложения могут возникать изменения структуры базы данных. Это может быть добавление новых таблиц, изменение существующей схемы или внесение других изменений. Управление миграциями базы данных является важной задачей для обеспечения согласованности данных и обновления базы данных без потери данных или проблем с работой приложения.

Работа в данной предметной области требует понимания принципов проектирования баз данных, знания SQL (Structured Query Language) для выполнения запросов и манипуляции данными, а также практического опыта с использованием PostgreSQL и TypeORM для эффективной интеграции базы данных в приложения.

# 2. Постановка задачи

Необходимо разработать клиент-серверное приложение для администрации футбольной команды с использованием базы данных PostgreSQL и интеграции с TypeORM. Приложение должно предоставлять функции CRUD (создание, чтение, обновление и удаление) для трех основных сущностей: игроки, клубы и администрация.

Функциональные возможности разрабатываемого приложения должны включать:

- Создание, просмотр, обновление и удаление игроков команды. Необходимо хранить информацию о каждом игроке, включая имя, фамилию, возраст, позицию, контактные данные, статистику матчей и другую связанную информацию.

- Создание, просмотр, обновление и удаление записей о клубах, включая информацию о названии, местоположении, тренерах, игроках и других характеристиках.

- Управление администрацией команды, включая добавление, просмотр, обновление и удаление членов администрации. Информация о каждом члене администрации должна включать имя, фамилию, должность, контактные данные и другую связанную информацию.

Решение данной задачи включает разработку схемы базы данных PostgreSQL, настройку подключения с использованием TypeORM и реализацию соответствующих CRUD-операций для каждой из трех сущностей. Пользовательский интерфейс должен быть разработан для взаимодействия с базой данных и предоставления возможности выполнения необходимых действий по управлению игроками, клубами и администрацией футбольной команды.

# 3. Описание архитектуры

Для разработки клиент-серверного приложения с использованием базы данных PostgreSQL и интеграции с TypeORM, будет реализована следующая архитектура:

1. Клиентское приложение:

- Отвечает за представление данных и взаимодействие с пользователем.

- Отправляет запросы на сервер и обрабатывает полученные данные.

- Отображает информацию об игроках, клубах и администрации.

2. Серверное приложение:

- Обрабатывает запросы от клиентского приложения.

- Управляет взаимодействием с базой данных PostgreSQL через TypeORM.

- Выполняет операции CRUD для работы с сущностями игроков, клубов и администрации.

Основные сущности и их обязанности:

1. Игроки:

- Хранят информацию об игроках команды, включая персональные данные, статистику матчей, контактную информацию и дополнительные характеристики.

- Обязанности:

- Предоставление информации о себе: имя, фамилия, возраст, позиция, контактная информация и статистика матчей.

- Участие в тренировках и матчах команды.

- Соблюдение правил и норм командной дисциплины.

2. Клубы:

- Содержат информацию о футбольных клубах, включая название, местоположение, тренеров, достижения и другие характеристики.

- Обязанности:

- Управление клубом: руководство, тренеры и администрация.

- Участие в играх и соревнованиях.

- Заведение и обновление информации о клубе.

3. Администрация:

- Включает членов администрации команды, такие как руководители, менеджеры и другие.

- Обязанности:

- Управление командой: принятие стратегических решений, подписание контрактов, достижение финансовых и спортивных целей.

- Коммуникация с другими сущностями (игроки, клубы) и обслуживающим персоналом.

Эта архитектура позволяет эффективно взаимодействовать с базой данных PostgreSQL, используя ORM-фреймворк TypeORM для выполнения различных операций с сущностями игроков, клубов и администрации. Серверное приложение ответственно за выполнение запросов от клиентского приложения и обработку данных перед обращением к базе данных, а клиентское приложение предоставляет удобный интерфейс для работы с данными и отображение информации пользователю.

# 4. Описание структуры БД

Описание структуры базы данных для предметной области "Футбольная команда" с учетом сущностей игроки, администрация и футбольные клубы:

1. Сущность "Игроки":

- Атрибуты:

- ID: уникальный идентификатор игрока.

- ФИО: ФИО игрока.

- Позиция: позиция, на которой играет игрок.

- ID клуба: идентификатор клуба, в котором играет игрок.

2. Сущность "Футбольные клубы":

- Атрибуты:

- ID: уникальный идентификатор клуба.

- Название: название футбольного клуба.

- Местоположение: местоположение клуба.

- Слоган: слоган клуба.

- Администрация: информация о тренерах клуба.

- Игроки: информация об игроках клуба.

3. Сущность "Администрация":

- Атрибуты:

- ID: уникальный идентификатор администрации.

- ФИО: ФИО администрации.

- Должность: должность, занимаемая членом администрации.

- Контактная информация: контактные данные члена администрации.

В данной структуре базы данных сущности игроки, футбольные клубы и администрация связаны между собой следующим образом:

- Каждый игрок относится к определенному футбольному клубу (внешний ключ "ID клуба").

- Футбольные клубы могут иметь несколько игроков (отношение один-ко-многим).

- Администрация может относиться к определенному футбольному клубу.

Таким образом, база данных будет содержать таблицы для каждой из сущностей (игроки, футбольные клубы, администрация), где каждая запись представляет отдельного игрока, клуб или члена администрации, а связи между таблицами будут обеспечены с использованием внешних ключей для установления связей между соответствующими записями.

A diagram of a group of text

Description automatically generated with medium confidence

Рисунок 1 – схема базы данных

# 5. Описание серверной части

1. Контроллеры (Controllers):

- Контроллеры являются "мозгом" приложения и обрабатывают запросы от клиентской части.

- Они интерпретируют данные запроса, взаимодействуют с моделями (Models) и решают, какое представление (View) отправить обратно клиенту.

- Контроллеры служат промежуточным звеном между клиентской частью и моделями, осуществляют логику обработки запросов и форматирование данных для представления.

2. Модели (Models):

- Модели представляют данные и бизнес-логику приложения.

- Они определяют структуру данных и правила доступа к этим данным.

- Модели взаимодействуют с базой данных и выполняют операции чтения и записи данных.

- Они предоставляют контроллерам необходимую информацию для выполнения запросов и обновления представлений.

3. Представления (Views):

- Представления отвечают за отображение данных в удобном для пользователя виде.

- Они генерируют HTML-страницы, которые отображают данные, полученные от контроллеров и моделей.

- Представления могут использовать шаблоны, чтобы упростить процесс отображения данных.

Диаграмма классов приложения может включать следующие классы и их взаимосвязи:

- PlayerController: контроллер для управления действиями, связанными с игроками.

- PlayerModel: модель для работы с данными об игроках.

- ClubController: контроллер для управления действиями, связанными с футбольными клубами.

- ClubModel: модель для работы с данными о футбольных клубах.

- AdminController: контроллер для управления действиями администрации.

- AdminModel: модель для работы с данными администрации команды.

Взаимодействие между компонентами обычно следует принципу «Model-View-Controller»:

1. Когда пользователь отправляет запрос, контроллер принимает его и взаимодействует с моделью для получения необходимых данных.

2. Полученные данные передаются представлению, которое форматирует их в удобный вид для отображения.

3. Представление отображает данные и отправляет их обратно пользователю.

Например, при выполнении запроса на получение информации об игроке, контроллер PlayerController сначала обращается к модели PlayerModel для получения данных об игроке.

Вот пример кода на языке TypeScript для модели игроков:

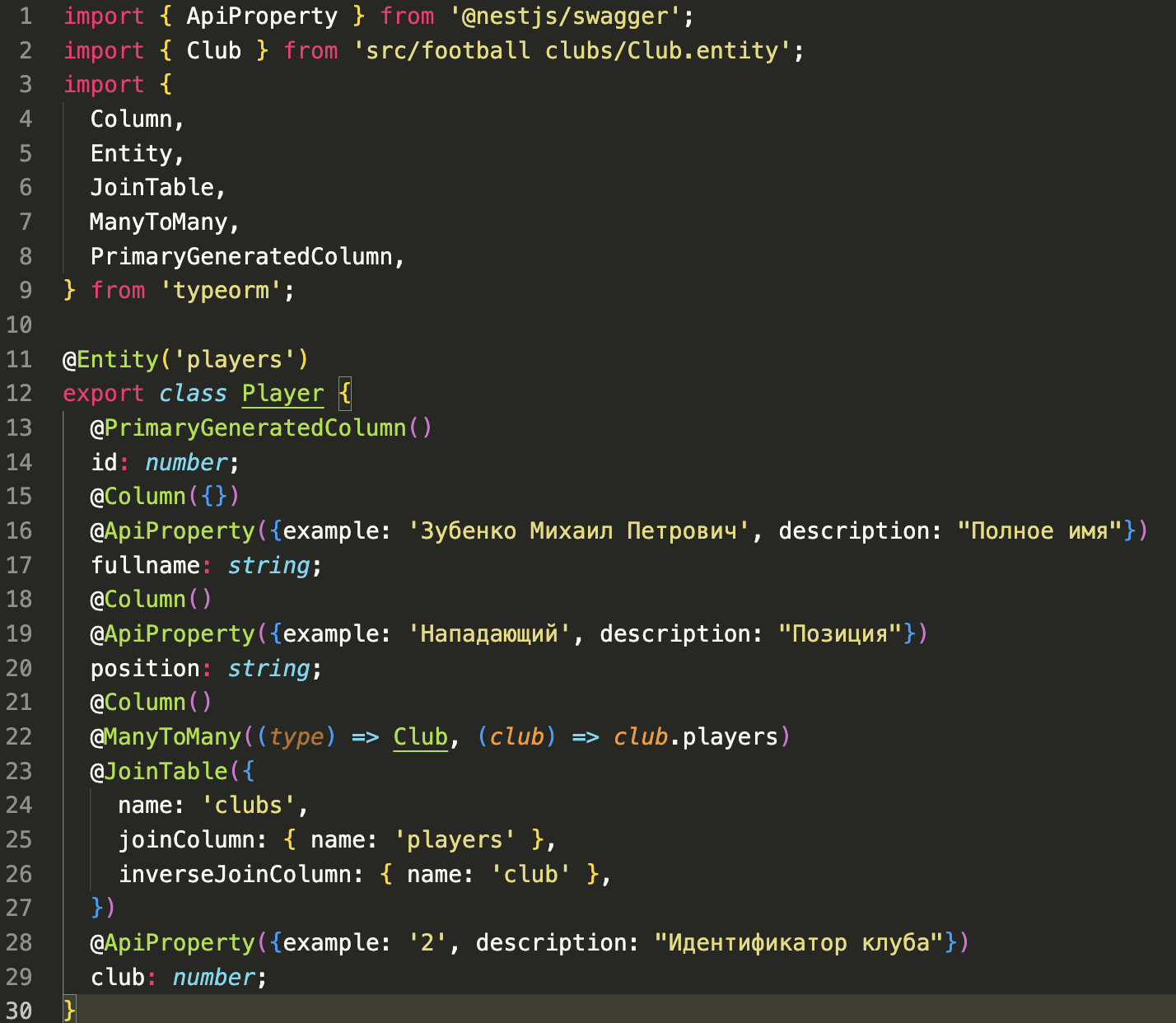


Рисунок 2 – модель Player.entity

Пример кода контроллера для модели игроков:



Рисунок 3 – котроллер для модели игроков

Пример кода сервиса для модели игроков:

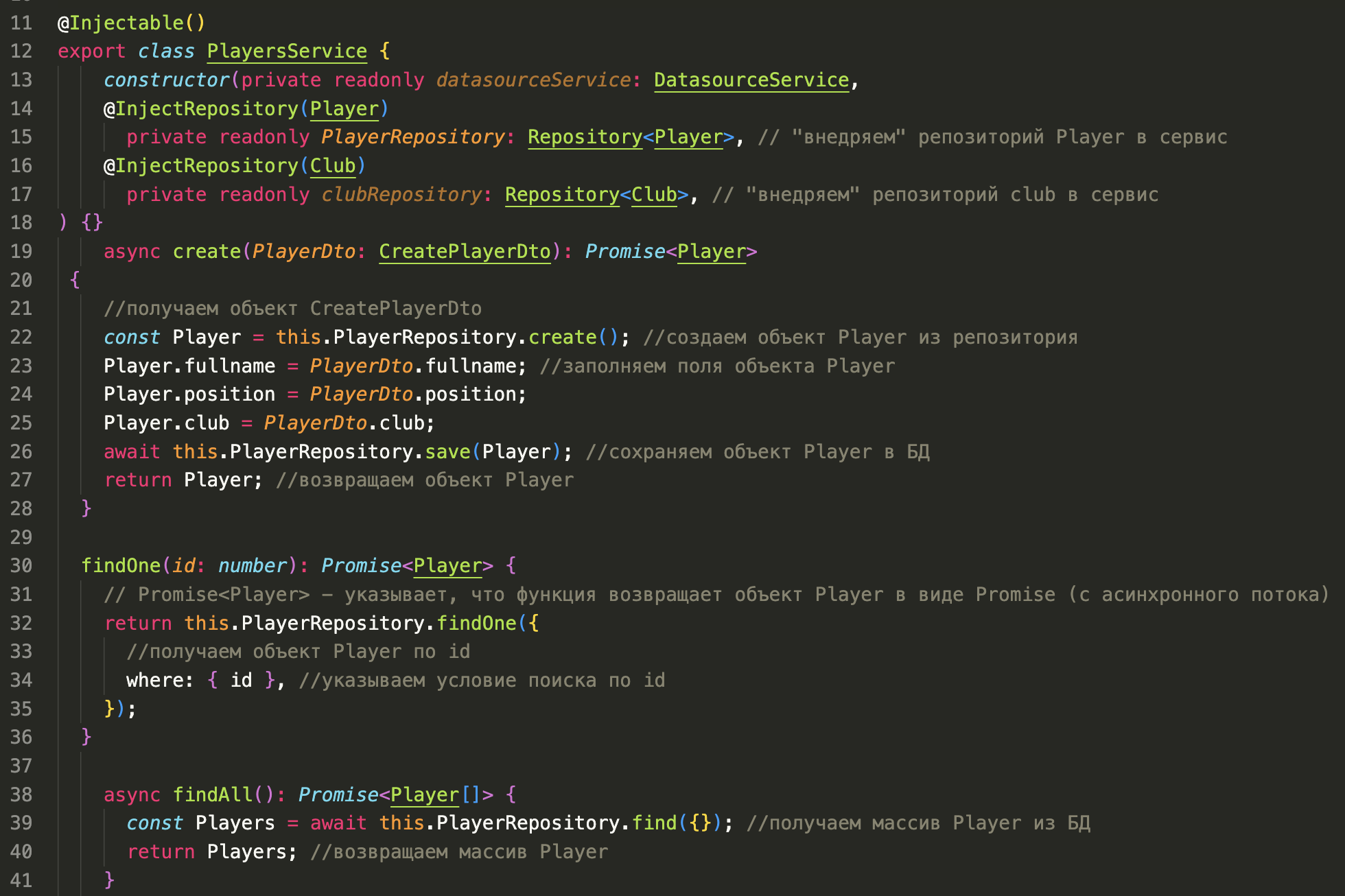


Рисунок 4 – сервис для модели игроков (1)

A screen shot of a computer program

Description automatically generated

Рисунок 5 – сервис для модели игроков (2)

# 6. Описание клиентской части

Описание клиентской части приложения включает элементы интерфейса для конечного пользователя и спецификацию API.

1. Элементы интерфейса (GUI):

- Формы ввода данных: это могут быть текстовые поля, раскрывающиеся списки, флажки или радиокнопки. Они предоставляют пользователю возможность вводить различные данные, такие как имя игрока, возраст, пол и другие.

- Кнопки действий: кнопки, которые выполняют определенные действия, например, "Сохранить", "Удалить" или "Отправить".

- Таблицы и списки: позволяют пользователю просматривать и отображать данные, такие как список игроков или футбольных клубов.

- Grafana visualization panel: это панель для визуализации данных, которая может отображать диаграммы, графики и другие типы визуализации.

2. Спецификация API:

Если приложение предоставляет API, можно включить спецификацию используемых эндпоинтов (URL-адресов) и методов HTTP, а также описать входные параметры и возвращаемые значения. Например:

- Эндпоинт: /api/players

- Метод: GET

- Входные параметры: отсутствуют

- Возвращаемое значение: список игроков в формате JSON

- Пример тестирования: При отправке GET-запроса по адресу /api/players, ожидается получение списка игроков

3. Пример тестирования API:

Можно предоставить примеры запросов и ожидаемых ответов для различных эндпоинтов API. Например:

- Тестирование эндпоинта /api/players:

- Запрос: GET /api/players

- Ожидаемый ответ: HTTP 200 OK

- Содержимое ответа: список игроков в формате JSON

Снизу предоставлен пример выведения списка игроков:

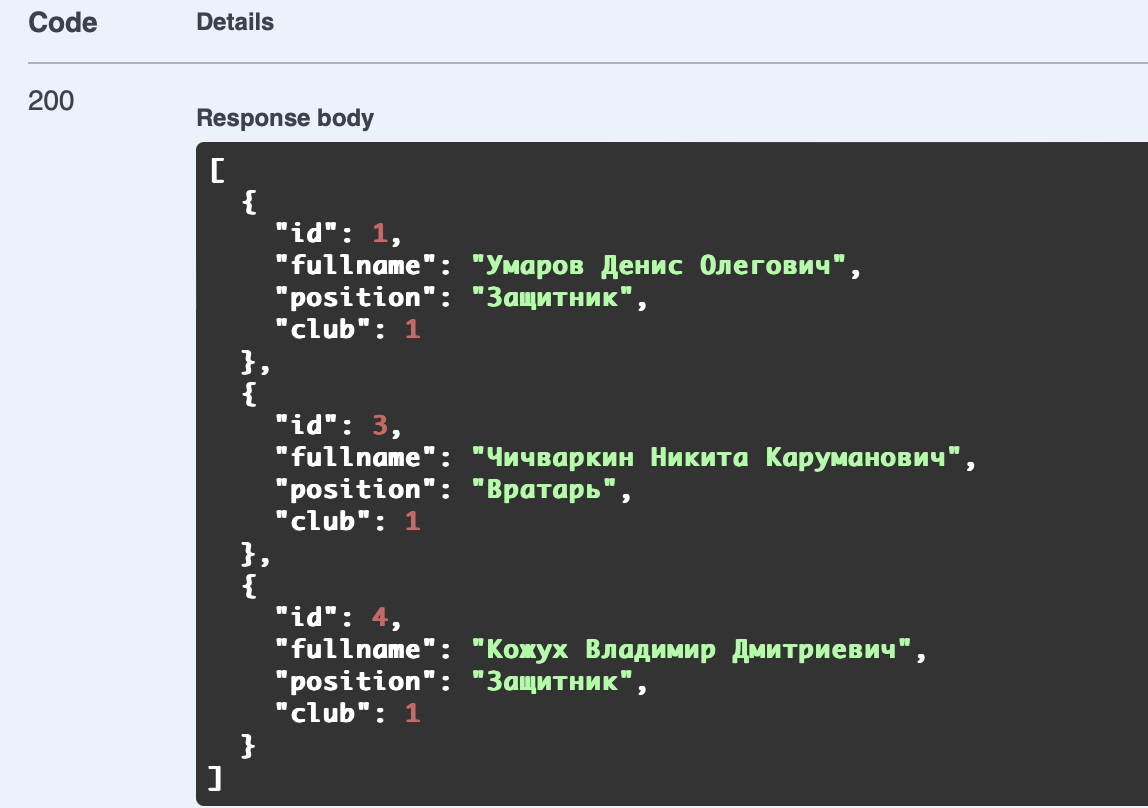


Рисунок 6 – пример GET для игроков

# 7. Заключение

В данном проекте рассматривалась предметная область "Футбольная команда" и разработка приложения для управления данными об игроках, администрации и футбольных клубах. Основная проблема, которую решало приложение, заключалась в эффективном управлении информацией о составе команды, футбольных клубах и администрации, а также обеспечении удобного доступа к этим данным.

Для решения данной проблемы была разработана клиент-серверная архитектура приложения с использованием базы данных PostgreSQL и интеграции с ORM-фреймворком TypeORM для обеспечения удобного взаимодействия с базой данных.

В результате работы была создана структура базы данных, включающая основные сущности: игроки, футбольные клубы и администрация. Каждая из сущностей была описана с указанием взаимосвязей между ними. Были разработаны модели для работы с данными, контроллеры для обработки запросов и представления для отображения данных.

Результаты работы включают разработанную структуру базы данных, реализацию серверной и клиентской частей приложения, а также возможность удобного взаимодействия с данными об игроках, футбольных клубах и администрации. Пользователи будут иметь возможность просматривать информацию о команде, добавлять и удалять игроков, обновлять данные и получать актуальную информацию о состоянии команды.

Используя PostgreSQL и TypeORM, мы обеспечили надежное хранение и управление данными о футбольной команде. Разработанная клиентская часть приложения позволяет конечным пользователям удобно взаимодействовать с данными через графический интерфейс или использовать предоставленное API для интеграции с другими системами.

Проект по созданию приложения для управления данными "Футбольная команда" важен для успешного функционирования футбольных клубов, обеспечивая эффективное управление информацией о игроках, клубах и администрации. Благодаря разработанной архитектуре и использованию соответствующих инструментов, было достигнуто качественное решение проблемы хранения и управления данными, а также предоставлен удобный функционал для конечных пользователей.

В целом, проект имеет значительный потенциал для оптимизации и автоматизации управления футбольными командами, а также интеграции с другими системами, что может повысить эффективность и успех команда в футбольной индустрии.

# 8. Список литературы

1. Martin Kleppmann. Designing Data-Intensive Applications. O'Reilly Media, 2017.

2. Ethan Brown. Web Development with Node.js, Express, and MongoDB. O'Reilly Media, 2014.

3. Alex Banks, Eve Porcello. Learning React: Modern Patterns for Developing React Apps.

O'Reilly Media, 2020.

4. Basarat Ali Syed. TypeScript Deep Dive. Leanpub, 2020.

5. Официальная документация PostgreSQL. https://www.postgresql.org/docs/

6. GitHub: