Пояснительная записка к проекту по Базам Данных

Выполнили:

- Алексеев Дмитрий Денисович, БПИ-236
- Лившиц Леонид Игоревич, БПИ-235
- Васюков Александр Владимирович, БПИ-235

Репозиторий на гитхабе: https://github.com/Dmitry-Alekseev01/hse-db-project

Описание проекта

В рамках проекта была реализована система управления футбольными клубами, предназначенная для использования в футбольных организациях, имеющих несколько команд и инфраструктуру для проведения матчей. Основными пользователями системы являются руководители клубов и администрация, отвечающая за управление командами, игроками, тренерским штабом.

Главной сущностью системы является футбольный клуб, который может включать в себя несколько команд, например, основная и молодежная. Каждая команда состоит из игроков и тренеров. Для тренеров предусмотрено разделение по специализациям, таким как главный тренер, тренер вратарей, тренер по игре в обороне, тренер по игре в атаке, тренер стандартных положений, тренер по физической подготовке и ассистент. Один тренер может совмещать несколько специализаций, а одна специализация может быть назначена нескольким тренерам.

Также важной частью системы являются игроки, которые привязываются к конкретной команде. Для каждого игрока хранится информация о его статусе, отражающем контракт с командой. В рамках одной команды может состоять неограниченное количество игроков, но при этом каждый игрок принадлежит ровно одной команде в текущий момент времени.

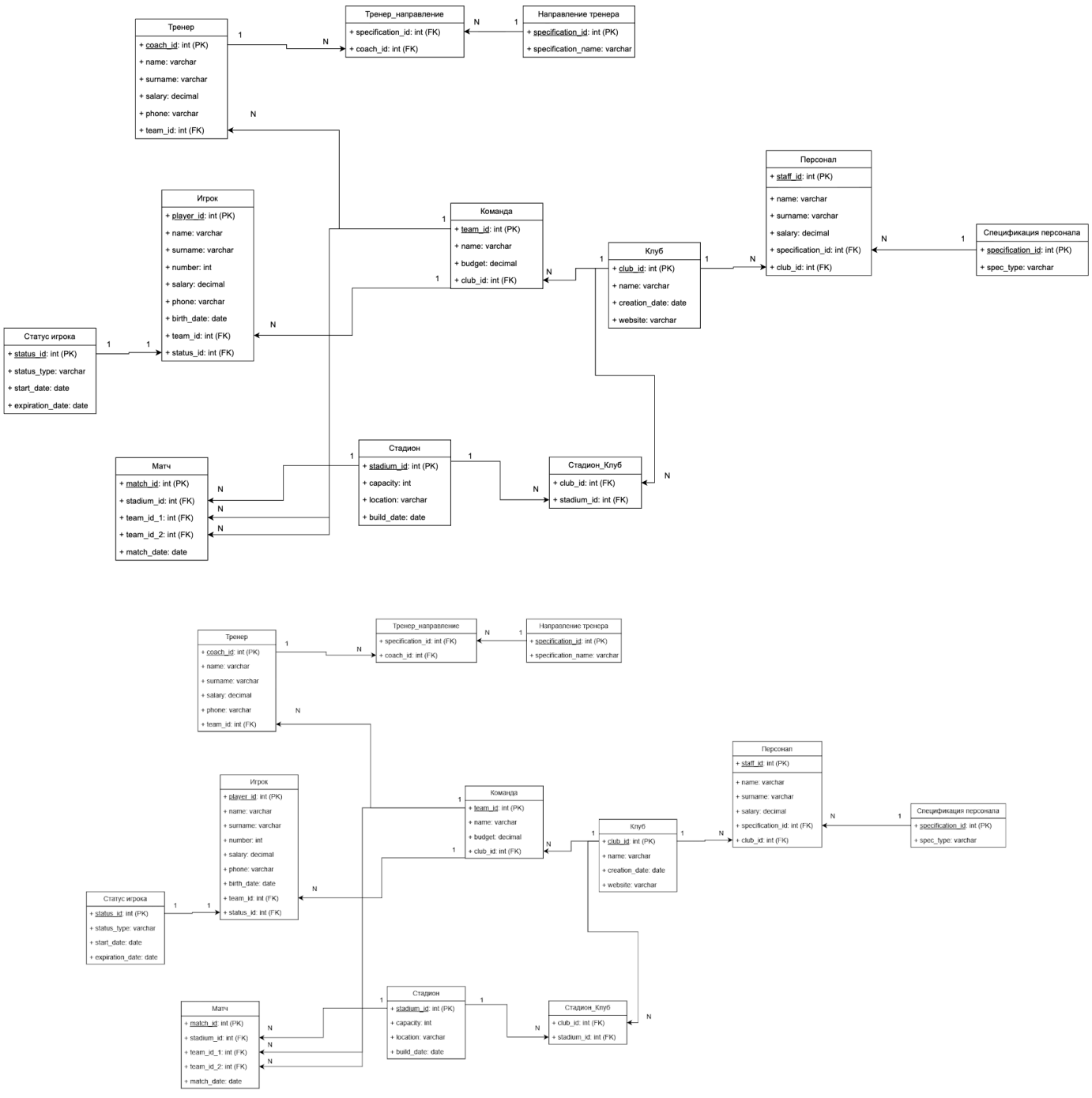
Помимо этого, в системе учитывается персонал футбольного клуба. Персонал разделяется по специализациям, например, уборщик, охранник, газонокосильщик, врач, директор и президент. Каждый сотрудник относится к одному клубу и имеет одну специализацию, это позволяет корректно учитывать структуру управления и обслуживания клуба.

Футбольные клубы могут использовать несколько стадионов, и в то же время один стадион может быть местом проведения матчей для разных клубов, как, например, итальянский стадион “Сан-Сиро”, который является домашним сразу для двух команд, Интера и Милана.

Между командами проводятся футбольные матчи. В каждом матче участвуют ровно две команды, при этом каждая команда может принимать участие в неограниченном количестве матчей. Для матча хранится информация о стадионе проведения, дате и посещаемости.

Диаграмма UML

> Выполнил Алексеев Дмитрий Денисович, БПИ-236



Ограничения:

- в таблице `CoachSpecification` могут быть только значения:
  - главный
  - тренер вратарей
  - тренер по игре в обороне
  - тренер по игре в атаке
  - тренер стандартных положений
  - тренер по физ. подготовке
  - ассистент
- в таблице `StaffSpecification` могут быть только значения:
  - уборщик
  - охранник
  - газонокосильщик
  - врач
  - директор
  - президент
- в таблице `Team` бюджет должен быть больше 100000
- в таблице `Staff` зарплата должна быть больше 30000
- в таблице `Stadium` вместительность должна быть больше 100
- в таблице `PlayerStatus` могут быть только значения:
  - в аренде

    - на контракте
- в таблице `Player` зарплата должна быть больше 30000
- в таблице `Club` у даты создания дефолтное значение — сегодняшний день

Все ограничение указаны в запросах DDL.

Схема БД приведена к третьей нормальной форме:

- каждая таблица имеет первичный ключ
- все неключевые атрибуты функционально зависят только от ключа
- транзитивные и многозначные зависимости устранены.

Проблемы из-за неденормализации

Если вместо отдельной таблицы `PlayerStatus`, состоящей из `status_id`, `status_type`, `start_date`, `expiration_date`, записать в таблицу `Player` все эти поля, будет несколько проблем:

1. Обновление: если изменяется контракт игрока, например, продлевается дата в `expiration_date`, то придется обновить много строк.
2. Вставка: нельзя создать запись о статусе игрока без игрока, то есть нельзя хранить какие-то абстрактные шаблоны контрактов.
3. Удаление: удаление игрока с каким-то статусом приведет к потере информации о самом статусе. То есть, если надо будет отдельно рассчитать статистику для бюджета команды, а игрок будет удален, то данные о контракте также удалятся.
4. Дублирование: строка `status_typ = 'на контракте'` будет храниться много раз.

Вынос информации о статусе игрока в отдельную таблицу `PlayerStatus` устраняет дублирование данных и предотвращает аномалии вставки, обновления и удаления, что соответствует требованиям нормализации до 3NF.

Функциональные зависимости

1. **Club**
   Зависимости: `club_id → name, creation_date, website`
   Ключ: `club_id`
2. **Team**
   Зависимости: `team_id → name, budget, club_id`
   Ключ: `team_id`
3. **Player**
   Зависимости: `player_id → name, surname, number, salary, phone, birth_date, team_id, status_id`
   Ключ: `player_id`
4. **PlayerStatus**
   Зависимости: `status_id → status_type, start_date, expiration_date`
   Ключ: `status_id`
5. **Coach**
   Зависимости: `coach_id → name, surname, salary, phone, team_id`
   Ключ: `coach_id`
6. **Coach_Specification**
   Зависимости: `(coach_id, specification_id) → ∅`
   Ключ: `(coach_id, specification_id)`
7. **CoachSpecification**
   Зависимости: `specification_id → specification_name`
   Ключ: `specification_id`
8. **Staff**
   Зависимости: `staff_id → name, surname, salary, specification_id, club_id`
   Ключ: `staff_id`

9. **StaffSpecification**
   Зависимости: `specification_id → spec_type`
   Ключ: `specification_id`
10. **Stadium_Club**
    Зависимости: `(club_id, stadium_id) → ø`
    Ключ: `(club_id, stadium_id)`
11. **Stadium**
    Зависимости: `stadium_id → capacity, location, build_date`
    Ключ: `stadium_id`
12. **Game**
    Зависимости: `game_id → stadium_id, team_id_1, team_id_2, match_date`
    Ключ: `game_id`

DDL

> Выполнил Алексеев Дмитрий Денисович, БПИ-236

```
create table CoachSpecification(
  specification_id serial primary key,
  specification_name varchar(50) not null check(specification_name in ('главный', 'тренер вратарей', 'трене
р по игре в обороне',
  'тренер по игре в атаке', 'тренер стандартных положений', 'тренер по физ. подготовке', 'ассистент'))
);

create table StaffSpecification(
  specification_id serial primary key,
  spec_type varchar(40) not null check(spec_type in ('уборщик', 'охранник', 'газонокосильщик',
  'врач', 'директор', 'президент'))
);

create table Club(
  club_id serial primary key,
  club_name varchar(50) not null,
  creation_date date not null default current_date,
  website varchar(100)
);

create table Team(
  team_id serial primary key,
  team_name varchar(50) not null,
  budget decimal not null check(budget > 100000),
  club_id int not null references Club(club_id)
);

create table Staff(
  staff_id serial primary key,
  staff_name varchar(30) not null,
  staff_surname varchar(30) not null,
  salary decimal check(salary > 30000),
  specification_id int not null references StaffSpecification(specification_id),
  club_id int not null references Club(club_id)
);

create table Stadium(
  stadium_id serial primary key,
  capacity int not null check(capacity > 100),
  stadium_location varchar(150) not null,
  build_date date not null
```

```
);

create table Stadium_Club(
  club_id int not null references Club(club_id),
  stadium_id int not null references Stadium(stadium_id)
);

create table Game(
  match_id serial primary key,
  stadium_id int not null references Stadium(stadium_id),
  team_1_id int not null references Team(team_id),
  team_2_id int not null references Team(team_id),
  match_date date not null default current_date,
  attendance int not null
);

create table PlayerStatus(
  status_id serial primary key,
  status_type varchar(20) not null check(status_type in ('в аренде', 'на контракте')),
  start_date date not null,
  expiration_date date not null
);

create table Player(
  player_id serial primary key,
  player_name varchar(30) not null,
  player_surname varchar(30) not null,
  player_number int not null,
  salary decimal not null check(salary > 30000),
  phone varchar(20) not null,
  birth_date date not null,
  team_id int not null references Team(team_id),
  status_id int not null references PlayerStatus(status_id)
);

create table Coach(
  coach_id serial primary key,
  coach_name varchar(30) not null,
  coach_surname varchar(30) not null,
  salary decimal not null,
  phone varchar(20) not null,
  team_id int not null references Team(team_id)
);

create table Coach_Specification(
  specification_id int not null references CoachSpecification(specification_id),
  coach_id int not null references Coach(coach_id)
);
```

DML

> Выполнил: Лившиц Леонид Игоревич, БПИ-235

Вставка:

```
BEGIN;

INSERT INTO club (club_name, creation_date, website) VALUES
('Реал Мадрид', '1902-03-06', 'https://www.realmadrid.com/en-US'),
```

```
('Манчестер Сити', '1880-11-23', 'https://www.mancity.com/');

INSERT INTO stadium (capacity, stadium_location, build_date) VALUES
(81044, 'Мадрид, Испания', '1947-12-14'),
(55097, 'Манчестер, Англия', '2003-08-10');

INSERT INTO stadium_club (club_id, stadium_id) VALUES (1,1), (2,2);

INSERT INTO team (team_name, budget, club_id) VALUES
('Реал Кастилья', 2000000.00, 1),
('Манчестер Молодежь', 1500000.00, 2);

INSERT INTO playerstatus (status_type, start_date, expiration_date) VALUES
('на контракте', '2023-01-01', '2026-12-31'),
('в аренде', '2024-07-01', '2025-06-30');

INSERT INTO player (player_name, player_surname, player_number, salary, phone, birth_date, team_id, status
_id) VALUES
('Иван', 'Компани', 9, 50000.00, '+37124949691', '1995-02-10', 1, 1),
('Петр', 'Стерлинг', 10, 60000.00, '+37128684642', '1993-06-20', 1, 1),
('Кевин', 'ДеБургер', 7, 45000.00, '+37123853783', '1998-11-05', 2, 1);

INSERT INTO coachspecification (specification_name) VALUES
('главный'), ('тренер вратарей'), ('тренер по игре в обороне'),
('тренер по игре в атаке'), ('тренер стандартных положений'),
('тренер по физ. подготовке'), ('ассистент');

INSERT INTO coach (coach_name, coach_surname, salary, phone, team_id) VALUES
('Хосеп', 'Гвардиола', 70000.00, '+37130000001', 1),
('Энцо', 'Мареска', 65000.00, '+37130000002', 2);

INSERT INTO coach_specification (specification_id, coach_id) VALUES
(1,1), (7,1), (3,2);

INSERT INTO staffspecification (spec_type) VALUES
('уборщик'), ('охранник'), ('газонокосильщик'),
('врач'), ('директор'), ('президент');

INSERT INTO staff (staff_name, staff_surname, salary, specification_id, club_id) VALUES
('Анна', 'Смит', 31000.00, 4, 1),
('Боб', 'Браун', 40000.00, 6, 2);

INSERT INTO game (stadium_id, team_1_id, team_2_id, match_date, attendance) VALUES
(1, 1, 2, '2025-12-20', 80000),
(2, 2, 1, '2026-01-10', 50000);

COMMIT;
```

Аренда игрока:

```
BEGIN;

DO $$
DECLARE
  p_player_id INT := 2; -- player_id
  p_loan_team_id INT := 2; -- команда, принимающая в аренду
  p_start_date DATE := '2025-12-08';
  p_end_date DATE := '2026-06-30';
  v_old_team INT;
```

```
  v_status_id INT;
BEGIN
  SELECT team_id INTO v_old_team FROM player WHERE player_id = p_player_id;
  IF NOT FOUND THEN RAISE EXCEPTION 'Игрок % не найден', p_player_id; END IF;

  IF v_old_team = p_loan_team_id THEN
    RAISE EXCEPTION 'Игрок уже в этой команде';
  END IF;

  -- статус "в аренде"
  SELECT status_id INTO v_status_id FROM playerstatus WHERE status_type = 'в аренде' LIMIT 1;
  IF NOT FOUND THEN
    INSERT INTO playerstatus (status_type, start_date, expiration_date)
      VALUES ('в аренде', p_start_date, p_end_date)
    RETURNING status_id INTO v_status_id;
  ELSE
    INSERT INTO playerstatus (status_type, start_date, expiration_date)
      VALUES ('в аренде', p_start_date, p_end_date)
    RETURNING status_id INTO v_status_id;
  END IF;

  -- Перевод игрока в команду арендатора и новый статус
  UPDATE player SET team_id = p_loan_team_id, status_id = v_status_id WHERE player_id = p_player_id;

  RAISE NOTICE 'Игрок % отдан в аренду команде % с % по %', p_player_id, p_loan_team_id, p_start_date,
p_end_date;
END $$ LANGUAGE plpgsql;

COMMIT;
```

Трансфер игрока:

```
BEGIN;

DO $$
DECLARE
  p_player_id INT := 1; -- player_id
  p_buyer_team_id INT := 2;  -- id покупающей команды
  p_transfer_fee NUMERIC := 100000.00; -- сумма трансфера
  seller_team_id INT;
  buyer_budget NUMERIC;
  seller_budget NUMERIC;
  v_status_id INT;
BEGIN
  -- текущая команда игрока
  SELECT team_id INTO seller_team_id FROM player WHERE player_id = p_player_id;
  IF seller_team_id IS NULL THEN
    RAISE EXCEPTION 'Игрок с id=% не привязан к команде', p_player_id;
  END IF;

  IF seller_team_id = p_buyer_team_id THEN
    RAISE EXCEPTION 'Покупающая команда = текущая команда';
  END IF;

  SELECT budget INTO buyer_budget FROM team WHERE team_id = p_buyer_team_id FOR UPDATE;
  IF NOT FOUND THEN RAISE EXCEPTION 'Покупающая команда % не найдена', p_buyer_team_id; END IF;

  SELECT budget INTO seller_budget FROM team WHERE team_id = seller_team_id FOR UPDATE;
  IF NOT FOUND THEN RAISE EXCEPTION 'Продающая команда % не найдена', seller_team_id; END IF;
```

```
  IF buyer_budget < p_transfer_fee THEN
    RAISE EXCEPTION 'Недостаточно бюджета у покупающей команды: % < %', buyer_budget, p_transfer_f
ee;
  END IF;

  -- Уменьшается бюджет покупающей
  UPDATE team SET budget = budget - p_transfer_fee WHERE team_id = p_buyer_team_id;

  -- увеличивается бюджет продающей
  UPDATE team SET budget = budget + p_transfer_fee WHERE team_id = seller_team_id;

  -- проверка, что есть статус "на контракте"
  SELECT status_id INTO v_status_id FROM playerstatus WHERE status_type = 'на контракте' LIMIT 1;
  IF NOT FOUND THEN
    INSERT INTO playerstatus (status_type, start_date, expiration_date)
    VALUES ('на контракте', current_date, current_date + INTERVAL '3 years')
    RETURNING status_id INTO v_status_id;
  END IF;

  -- статус "на контракте"
  UPDATE player SET team_id = p_buyer_team_id, status_id = v_status_id WHERE player_id = p_player_id;

  RAISE NOTICE 'Трансфер игрока % завершён: продавец team=% , покупатель team=%, сумма=%',
               p_player_id, seller_team_id, p_buyer_team_id, p_transfer_fee;
END $$ LANGUAGE plpgsql;

COMMIT;
```

Нанимаем на работу сотрудника:

```
BEGIN;

DO $$
DECLARE
  p_club_id INT := 1;
  p_name TEXT := 'Ольга';
  p_surname TEXT := 'Иванова';
  p_salary NUMERIC := 35000.00;
  p_spec_type TEXT := 'врач';
  v_spec_id INT;
BEGIN
  SELECT specification_id INTO v_spec_id FROM staffspecification WHERE spec_type = p_spec_type LIMIT 1;
  IF NOT FOUND THEN
    INSERT INTO staffspecification (spec_type) VALUES (p_spec_type) RETURNING specification_id INTO v_sp
ec_id;
  END IF;

  INSERT INTO staff (staff_name, staff_surname, salary, specification_id, club_id)
  VALUES (p_name, p_surname, p_salary, v_spec_id, p_club_id);

  RAISE NOTICE 'Сотрудник % % принят в клуб % как % (spec_id=%)', p_name, p_surname, p_club_id, p_sp
ec_type, v_spec_id;
END $$ LANGUAGE plpgsql;

COMMIT;
```

Вставка матча:

```
BEGIN;

DO $$
DECLARE
  p_stadium_id INT := 1;
  p_home_team INT := 1;
  p_away_team INT := 2;
  p_match_date DATE := '2026-03-15';
  p_expected_attendance INT := 50000;
  v_capacity INT;
  v_conflict_count INT;
BEGIN
  IF p_home_team = p_away_team THEN
    RAISE EXCEPTION 'Домашняя и гостевая команды совпадают (id=%).', p_home_team;
  END IF;

  SELECT capacity
  INTO v_capacity
  FROM stadium
  WHERE stadium_id = p_stadium_id
  FOR UPDATE;

  IF NOT FOUND THEN
    RAISE EXCEPTION 'Стадион с id=% не найден.', p_stadium_id;
  END IF;

  IF v_capacity < p_expected_attendance THEN
    RAISE EXCEPTION 'Ожидаемая посещаемость (%) больше вместимости стадиона (%).', p_expected_att
endance, v_capacity;
  END IF;

  SELECT COUNT(*) INTO v_conflict_count
  FROM game
  WHERE stadium_id = p_stadium_id
    AND match_date = p_match_date;

  IF v_conflict_count > 0 THEN
    RAISE EXCEPTION 'На стадионе % уже запланирован матч на дату %.', p_stadium_id, p_match_date;
  END IF;

  INSERT INTO game (stadium_id, team_1_id, team_2_id, match_date, attendance)
  VALUES (p_stadium_id, p_home_team, p_away_team, p_match_date, p_expected_attendance);

  RAISE NOTICE 'Матч успешно запланирован: stadium=% date=% home=% away=% attendance=%',
               p_stadium_id, p_match_date, p_home_team, p_away_team, p_expected_attendance;
END $$ LANGUAGE plpgsql;

COMMIT;
```

Проверка бюджета:

```
BEGIN;

DO $$
DECLARE
  rec RECORD;
  deficit NUMERIC;
BEGIN
```

```
  FOR rec IN
    SELECT t.team_id, t.team_name, t.budget, COALESCE(SUM(p.salary),0) AS total_salaries
    FROM team t
    LEFT JOIN player p ON p.team_id = t.team_id
    GROUP BY t.team_id, t.team_name, t.budget
    HAVING COALESCE(SUM(p.salary),0) > t.budget
  LOOP
    deficit := rec.total_salaries - rec.budget;
    RAISE NOTICE 'Команда % (id=%) имеет дефицит % (зарплаты=%, бюджет=%).', rec.team_name, rec.te
am_id, deficit, rec.total_salaries, rec.budget;

    UPDATE team SET budget = budget + deficit WHERE team_id = rec.team_id;
  END LOOP;

  RAISE NOTICE 'Проверка бюджета завершена.';
END $$ LANGUAGE plpgsql;

COMMIT;
```

Назначение тренера:

```
BEGIN;

DO $$
DECLARE
  p_team_id INT := 1;
  p_coach_name TEXT := 'Юрген';
  p_coach_surname TEXT := 'Клопп';
  p_salary NUMERIC := 500000.00;
  p_phone TEXT := '+3757583847692';
  p_specs TEXT[] := ARRAY['главный'];
  v_coach_id INT;
  v_spec_id INT;
  v_spec TEXT;
BEGIN
  INSERT INTO coach (coach_name, coach_surname, salary, phone, team_id)
  VALUES (p_coach_name, p_coach_surname, p_salary, p_phone, p_team_id)
  RETURNING coach_id INTO v_coach_id;

  FOREACH v_spec IN ARRAY p_specs
  LOOP
    SELECT specification_id INTO v_spec_id
    FROM coachspecification
    WHERE specification_name = v_spec
    LIMIT 1;

    IF NOT FOUND THEN
      RAISE EXCEPTION 'Спецификация "%" не найдена в coachspecification', v_spec;
    END IF;

    INSERT INTO coach_specification (specification_id, coach_id)
      VALUES (v_spec_id, v_coach_id);
  END LOOP;

  RAISE NOTICE 'Добавлен тренер % % с id=% и специализациями %',
               p_coach_name, p_coach_surname, v_coach_id, array_to_string(p_specs, ', ');
END $$ LANGUAGE plpgsql;
```

```
COMMIT;
```

Все стадионы, связанные с клубом:

```
SELECT s.stadium_id,
    s.stadium_location,
    s.capacity,
    c.club_name AS club_name
FROM stadium s
JOIN stadium_club sc ON s.stadium_id = sc.stadium_id
JOIN club c ON sc.club_id = c.club_id
WHERE c.club_id = 1;
```

Все игроки со статусом:

```
SELECT p.player_id,
    p.player_name,
    p.player_surname,
    p.player_number,
    ps.status_type
FROM player p
LEFT JOIN playerstatus ps ON p.status_id = ps.status_id
WHERE p.team_id = 1
ORDER BY p.player_number;
```

Вся зарплатная ведомость клуба:

```
SELECT t.team_id,
    t.team_name,
    COALESCE(SUM(p.salary), 0) AS total_players_salary,
    t.budget
FROM team t
LEFT JOIN player p ON t.team_id = p.team_id
GROUP BY t.team_id, t.team_name, t.budget;
```

API

> Выполнил Васюков Александр Владимирович, БПИ 235

Описание программной части

Была разработана серверная программа на языке Golang с использованием фреймворка Gin, реализующая REST-API для управления футбольными клубами и связанными с ними сущностями. Программа обеспечивает взаимодействие с реляционной базой данных и предоставляет единый интерфейс для работы с данными.

API предназначено для управления следующими сущностями:

- **Club** — футбольные клубы
- **Team** — команды внутри клуба
- **Player** — игроки команд
- **Coach** — тренеры команд
- **Staff** — персонал клуба
- **Stadium** — стадионы
- **Game** — футбольные матчи

Для всех сущностей реализованы стандартные CRUD-операции (создание, чтение, обновление и удаление), а также базовая серверная валидация входных данных в соответствии с ограничениями, заданным на уровне базы данных.

В качестве СУБД используется PostgreSQL 17. Приложение и база данных разворачивается в контейнерах с использованием Docker и Docker Compose, что обеспечивает воспроизводимость окружения и упрощает запуск проекта. Для автоматизации основных операций (создание секретов, сборка, запуск) есть Makefile.

При запуске приложения автоматически выполняются миграции, в ходе которых создается база данных и таблицы. После этого база данных заполняется тестовыми данными, что позволяет сразу приступить к работе с API и проверить корректность его функционирования.

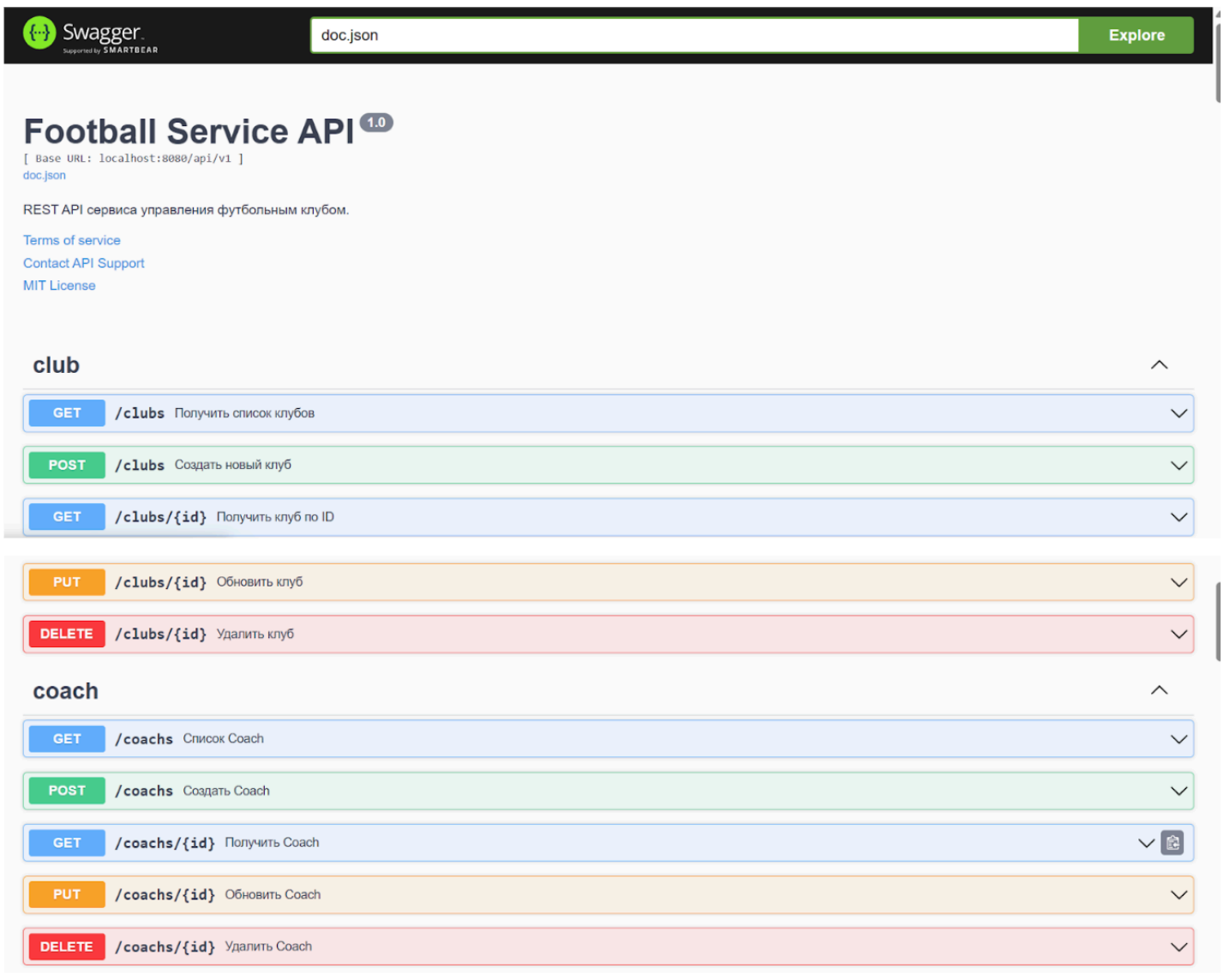
Для документирования AOI и удобного тестирования запросов используется Swagger, предоставляющий интерактивный интерфейс для просмотра доступных эндпоинтов и их параметров.

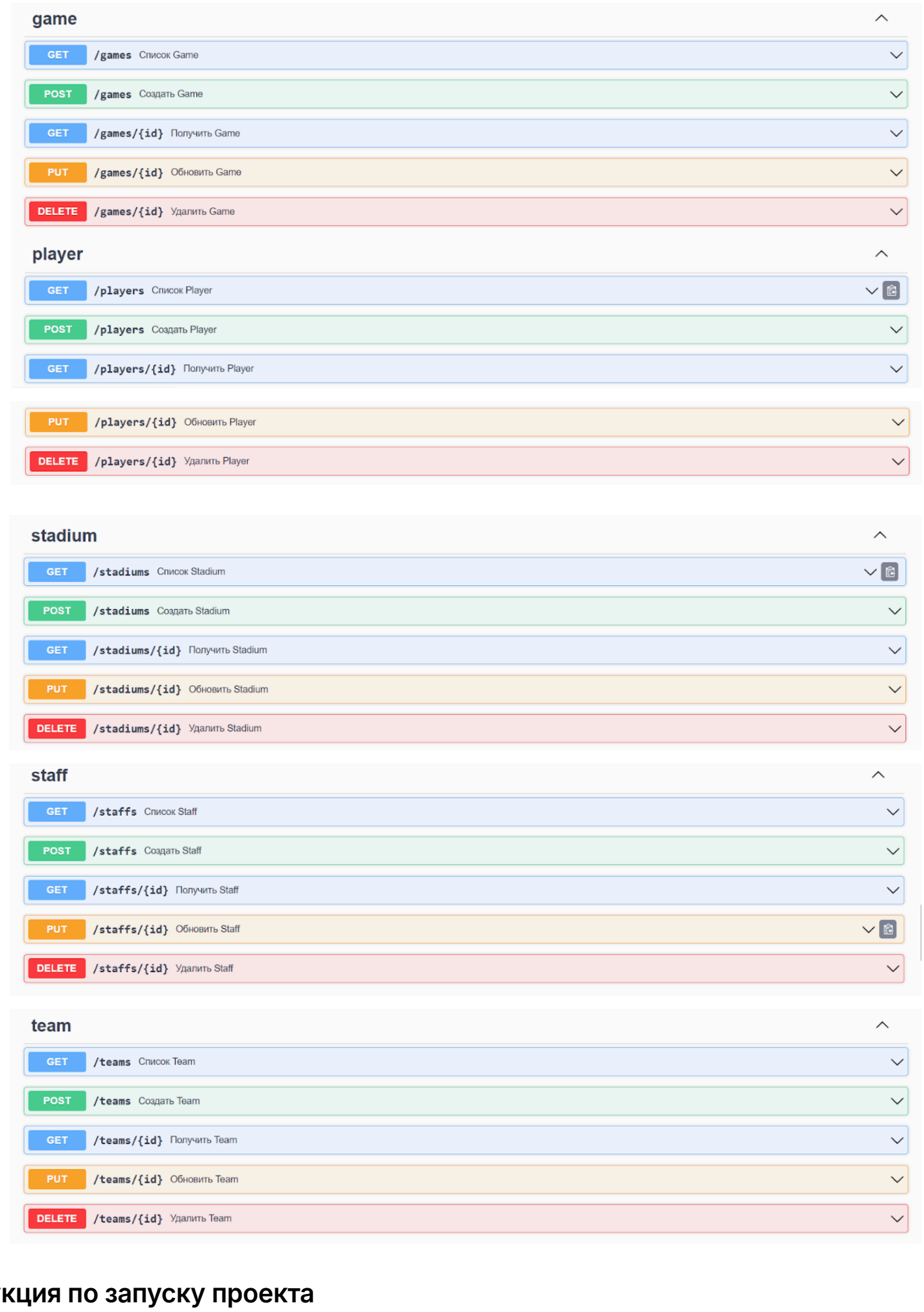
Базовый контракт

- Формат передачи данных: JSON
- Используются HTTP-коды состояния:
  - `200 OK` — успешное выполнение запроса
  - `201 Created` — успешное создание ресурса
  - `204 No Content` — успешное выполнение запроса без тела ответа
  - `400 Bad Request` — ошибка валидации или некорректный запрос
  - `404 Not Found` — запрашиваемый ресурс не найден
  - `500 Internal Server Error` — внутренняя ошибка сервера
- Валидация на всех ограничениях БД.

Эндпоинты REST

Все REST-эндпоинты доступны по базовому пути `/api/v1/`.





Инструкция по запуску проекта

1. Склонировать репозиторий:

```
git clone https://github.com/Dmitry-Alekseev01/hse-db-project.git
```

2. Перейти в директорию программы:

```
cd app
```

3. Запустить сервисы

```
make up
```

Для корректного запуска необходимо, чтобы был установлен и запущен Docker. При необходимости могут быть изменены данные в файле `.env`.

Нефункциональные требования

NFR-1. Целостность и непротиворечивость данных

Требование: База данных должна гарантировать логическую целостность данных и не позволять появление некорректных или противоречивых записей.

Зачем нужно: При ошибках приложения или некорректных запросах данные в БД не должны нарушать бизнес-логику предметной области. Например, не должно быть отрицательного бюджета, матча с невозможной вместимостью, зарплаты ниже уровня МРОТ.

Реализация: Используются первичные и внешние ключи для ссылочной целостности между сущностями, check-ограничения, например, `budget > 10000`, `capacity > 100`.

NFR-2. Транзакционная целостность бизнес-операций

Требование: Важные операции должны выполняться атомарно: либо выполняться полностью, либо не выполняться вовсе.

Зачем нужно: Операции вроде трансфера игрока задействуют игрока, его статус, две команды, то есть сразу несколько таблиц. Частичные выполнения таких действий приведут к логически некорректному состоянию БД.

Реализация: Используются явные транзакции (BEGIN, COMMIT, ROLLBACK), блоки DO, блокировки строк, откат транзакций при нарушении условий.

NFR-3. Минимизация аномалий обновлений из-за нормализации

Требование: Схема базы данных должна быть нормализована как минимум до третьей нормальной формы, чтобы исключить аномалии вставки, обновления и удаления.

Зачем нужно: Денормализованные структуры приводят к дублированию данных и ошибкам при изменениях, особенно при росте объема данных и числа операций.

Реализация: Повторяющиеся и логически независимые сущности вынесены в отдельные таблицы: статусы игроков, спецификации тренеров и персонала, связи многие-ко-многим.