



Минобрнауки России  
федеральное государственное бюджетное образовательное учреждение  
высшего образования  
«Санкт-Петербургский государственный технологический институт  
(технический университет)»

**Направление подготовки** 09.03.02 Информационные системы и  
технологии  
**Факультет** Информационных технологий и управления  
**Кафедра** Системного анализа и информационных  
технологий  
**Курс** 2 **Группа** 438

### **УПРАЖНЕНИЕ ПО ДИСЦИПЛИНЕ**

#### **«Разработка приложения»**

Выполнил обучающийся:

А. П. Черевков

Проверяла доцент:

С. А. Александрова

Санкт-Петербург  
2025

## СОДЕРЖАНИЕ

|                                                                        |    |
|------------------------------------------------------------------------|----|
| Введение .....                                                         | 3  |
| 1 Постановка задания.....                                              | 4  |
| 1.1 План реализации.....                                               | 5  |
| 2 Описание бизнес-процесса и ER-диаграмма.....                         | 6  |
| 3 Потребности бизнеса и таблица с функциями приложения и чат-бота..... | 11 |
| 4 Вид программы.....                                                   | 15 |
| 5 Вывод.....                                                           | 37 |
| 6 Список источников.....                                               | 38 |
| 7 Приложение .....                                                     | 39 |

## Введение

В современных информационных системах для эффективного взаимодействия с данными широко применяются веб-технологии и базы данных. Среди популярных инструментов выделяется Python с его фреймворком Flask, который позволяет быстро создавать легковесные и масштабируемые веб-приложения. Flask отличается простотой и гибкостью, что делает его отличным выбором для реализации серверной части, в том числе для поддержки чат-ботов.

Чат-боты, интегрируемые через Telegram API, становятся удобным средством взаимодействия пользователя с приложениями, обеспечивая быстрый и интуитивный доступ к информации. При этом базой данных MySQL служит надежным хранилищем данных, обеспечивая структурированное хранение и быстрый доступ к пользовательским данным и бизнес-логике. В рамках учебного проекта, где необходимо хранить ограниченный список — например, 30 игроков, — такая связка Python + Flask + MySQL является оптимальным решением для создания функционального, простой в сопровождении и расширяемого клиент-серверного приложения.

## **1 Постановка задания**

### **Задание:**

Заполнить базу данных информацией по выбранной теме. Разработать для базы данных графический интерфейс. В работе использовался **Python 3.12.6**, библиотека **Flask** и **MySQL Workbench 8.0 CE**. Работа была посвящена выбору победителя в номинации “**Золотой мяч**”.

**Основной язык: PYTHON 3.12.6.**

**Графический интерфейс: Flask.**

**База данных: MySQL Workbench 8.0 CE.**

**Слой документов: Football.txt.**

### **Основные цели:**

- Хранить все информацию на базе данных (10 таблиц).
- Вывести формулу для расчета победителя в номинации (голы + голевые передачи + сухие матчи + победы + ничьи + трофеи – поражения) \* Джентельменский коэффициент.
- Джентельменский коэффициент: от 1 до 5.
- Осуществить создание анкеты футболиста со всеми нужными полями: имя, фамилия, возраст, голы, голевые передачи, сухие матчи, победы, ничьи, поражения, джентельменский коэффициент, название клуба.
- Осуществить просмотр txt документа, там где это нужно.
- Очищать поля при надобности.
- Создать отдельную анкету под клуб с полями: название клуба, чемпионат страны, кубок страны, суперкубок, лига чемпионов.
- Создать аутентификацию для режима организатора и голосующего.
- В режиме организатора: добавить голосующего, создать запрос, удалить данные из таблицы, посмотреть список голосующих, посмотреть победителя, пересчитать победителя и награды.

- Количество игроков = 30.
- Использовать хешированный пароль.
- Применять логирование.
- Разработать удобный графический интерфейс.
- Добавить нужные проверки.
- Использовать Flask для написания интерфейса.
- Добавить чат-бота.

## **1.1 План реализации**

**Постановка задачи:** определение конкретных этапов разработки и критериев успеха.

**Проекты структур данных:** создание подробной ER-диаграммы, определение атрибутов каждой сущности.

**Разработка интерфейса:** Проектирование удобной формы подачи информации и удобства работы с приложением.

**Выбор технологии:** использование библиотеки **Flask** для графического интерфейса и **MySQL** для базы данных.

**Модульное тестирование:** проверка каждого компонента отдельно перед объединением.

**Финальное тестирование:** комплексное тестирование всего приложения.

**Запуск и мониторинг:** запуск программы, контроль её работоспособности и внесение доработок.

**Внедрение чат-бота:** использование чат-бота для голосования.  
Создание дополнительного слоя клиента.

## **2 Описание бизнес-процесса и ER-диаграмма**

Для эффективного управления голосами за “Золотой мяч” и решения поставленных задачий.

**Организатор:** отвечает за ввод данных о футболистах, формирование списков голосующих, обработку заявок, ведение статистики матчей и команд, управление базой данных и выбирает обладателя “**Золотого мяча**”.

**Голосующий:** выбирает одного или нескольких кандидатов на премию «**Золотой мяч**», это касается клубов и игроков, основываясь на предложенном списке спортсменов или вводит своих.

- Разделить голоса за клуб и за игрока.
- Создать удобные поля для ввода информации в каждом модуле.
- Разделить голосующих и организатора (админа).
- Реализовать функциональность для модуля админа: добавлять пользователя, сохранять пользователя, удалять записи, создавать SQL-запросы.
- 10 таблиц: Джентельменский коэффициент, Обладатель, Результаты + игрок, Трофеи + клуб, Личная статистика, Лучший бомбардир и ассистент, Трофеи + клуб, Футболисты, Логи, Пользователи и Администраторы.

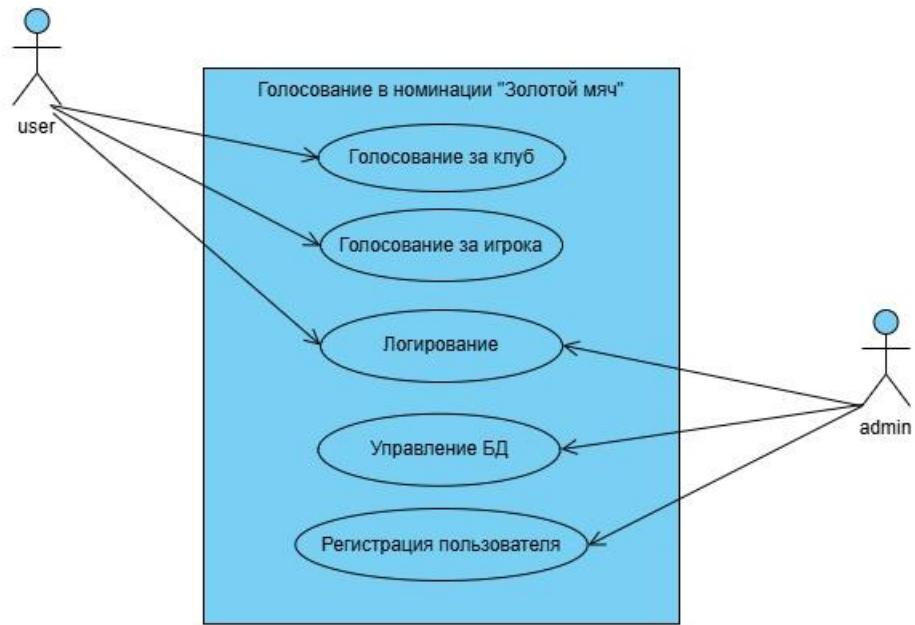


Рисунок 1 – UML-диаграмма

ER-диаграмма описывает структуру базы данных, необходимую для поддержки функционала системы. В нееходят следующие сущности:

### 1. Голосующие:

- Логин;
- Пароль;
- Поля для голосования и ввода информации.

### 2. Администратор:

- Логин;
- Пароль;
- Информация о голосующем;
- Удаление и запись в БД;
- Вывод победителя;
- Запросы в БД.

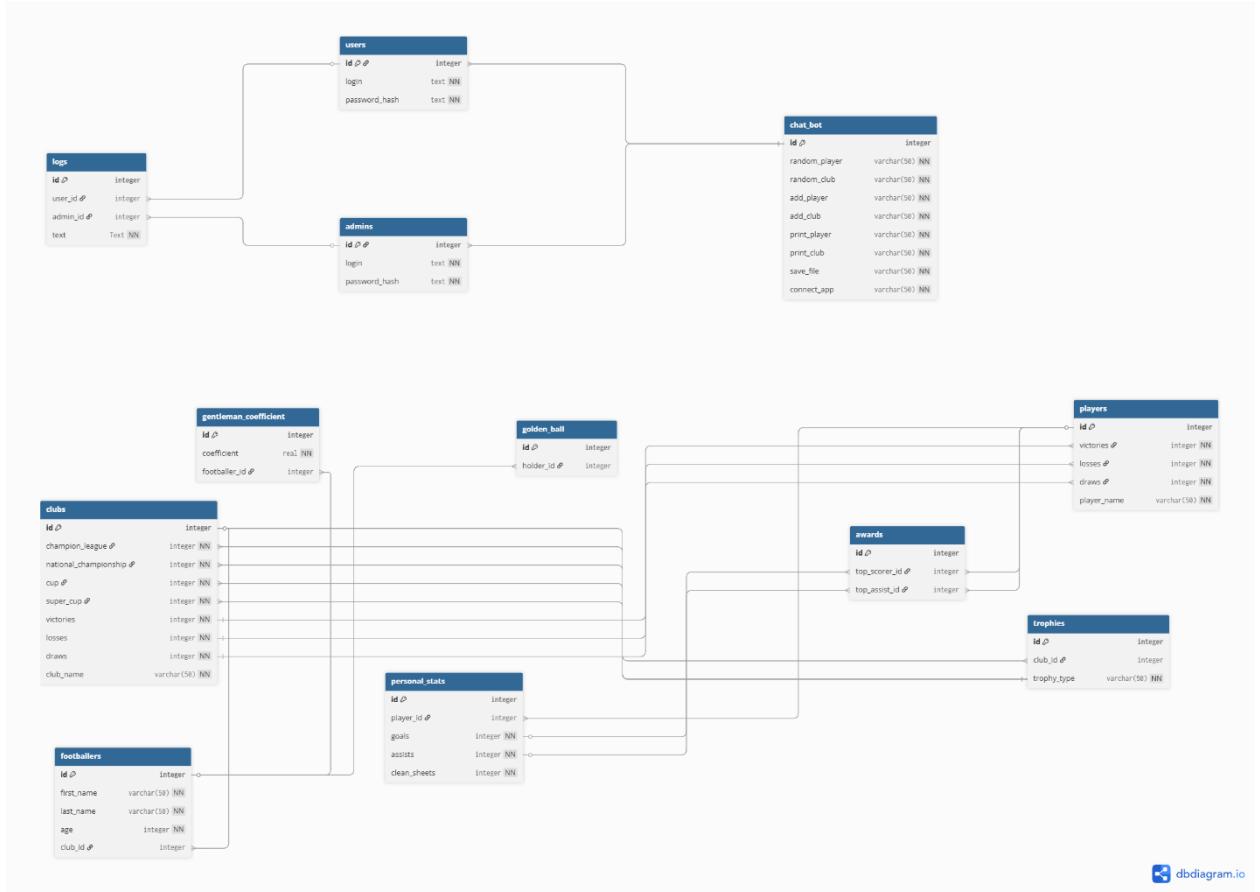


Рисунок 2 – ER-диаграмма

## 1. players:

- `id`;
- `victories`;
- `losses`;
- `draws`;
- `player_name`.

## 2. footballers:

- `id`;
- `first_name`;
- `last_name`;
- `age`;
- `club_id`.

### **3. clubs:**

- id;
- champions\_league;
- national\_championship;
- cup;
- super\_cup;
- losses;
- draws;
- victories;
- club\_name.

### **4. trophies:**

- id;
- club\_id;
- trophy\_type.

### **5. gentlemen\_coefficient**

- id;
- coefficient;
- footballer\_id.

### **6. awards**

- id;
- top\_scorer\_id;
- top\_assist\_id.

### **7. golden\_ball**

- id;
- holder\_id.

### **8. logs**

- id;
- user\_id;

- admin\_id;
- text.

## 9. personal\_stats

- id;
- player\_id;
- goals;
- assists;
- clean\_sheets.

## 10. users

- id;
- login;
- password\_hash.

## 11. admins

- id;
- login;
- password\_hash.

## 12. chat\_bot

- id;
- random\_player;
- random\_club;
- add\_player;
- add\_club;
- save\_txt;
- connect\_app.

### **3 Потребности бизнеса и таблица с функциями приложения и чат-бота**

Автоматизировать процедуру выбора обладателей престижной футбольной премии (**«Золотой мяч»**).

Организовать эффективное хранение и управление данными обо всех участниках, кубках и этапах процесса.

Повысить прозрачность процедуры голосования путём ведения детальных журналов активности и проверок целостности данных.

#### **Для функций приложения (app.py)**

| <b>Функции</b>                               | <b>Для чего нужна</b>                                                                                             |
|----------------------------------------------|-------------------------------------------------------------------------------------------------------------------|
| <code>def get_db_connection():</code>        | Функция возвращает соединение с базой данных MySQL<br>Возвращает объект соединения или None в случае неудачи.     |
| <code>def test_connection():</code>          | Тестирует работоспособность подключения к базе данных.<br>Возвращает True, если успешно подключился, иначе False. |
| <code>def initialize_database():</code>      | Инициализация базы данных и создание таблиц с нужными колонками                                                   |
| <code>def encrypt_password(password):</code> | Функция для хеширования пароля (SHA256)                                                                           |
| <code>def write_log(text):</code>            | Функция записи логов с обработкой ошибок блокировки базы                                                          |

|                                                                                                                                                                                                        |                                                                                |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------|
| <code>def vote_player():</code>                                                                                                                                                                        | Функция добавления нового футболиста вместе со статистикой и обновлением клуба |
| <code>def vote_team():</code>                                                                                                                                                                          | Функция добавления / обновления клуба с трофеями в базе                        |
| <code>def calculate_awards_and_winner():</code>                                                                                                                                                        | Функция добавления/обновления клуба с трофеями в базе                          |
| <code>def create_voting_player_window():</code>                                                                                                                                                        | Создание окна для голосования                                                  |
| <code>def validate_age(value):</code><br><code>def validate_goals_and_losses_draws_assist(value):</code><br><code>def validate_gentleman_coef(value):</code><br><code>def validate_name(value):</code> | Проверки на ввод                                                               |
| <code>def clear_input_fields():</code>                                                                                                                                                                 | Очистка полей                                                                  |
| <code>def submit_footballer_with_coef():</code>                                                                                                                                                        | Футболист + коэффициент                                                        |
| <code>def show_footballers_file():</code>                                                                                                                                                              | Показать файл с футболистами                                                   |
| <code>def create_voting_team_window():</code>                                                                                                                                                          | Создание окна для голосования за клуб                                          |
| <code>def validate_club_cups(value):</code>                                                                                                                                                            | Проверка на введенное значение клуба                                           |
| <code>def submit_team():</code>                                                                                                                                                                        | Добавить клуб                                                                  |
| <code>def open_admin_panel():</code>                                                                                                                                                                   | Открыть окно админа                                                            |
| <code>def add_voter():</code>                                                                                                                                                                          | Добавить голосующего                                                           |
| <code>def save_voter():</code>                                                                                                                                                                         | Сохранить голосующего                                                          |

|                                    |                                                 |
|------------------------------------|-------------------------------------------------|
| def view_voters():                 | Посмотреть голосующих                           |
| def view_golden_ball_holders():    | Посмотреть обладателя                           |
| def create_query():                | Сделать SQL-запрос                              |
| def run_query():                   | Осуществить запрос                              |
| def delete_record():               | Удалить запись                                  |
| def perform_delete():              | Проверка на удаление записи                     |
| def calculate_and_notify_awards(): | Расчет победителя                               |
| def authenticate_team_voter():     | Аутентификация голосующего за клуб              |
| def login_team_voter():            | Логирование входа в режим голосующего за клуб   |
| def authenticate_player_voter():   | Аутентификация голосующего за игрока            |
| def login_player_voter():          | Логирование входа в режим голосующего за игрока |
| def admin_auth_window():           | Вход в режим админа                             |
| def login_admin():                 | Логирование админа                              |
| def main():                        | Мейн                                            |

## Для функций бота (telegram\_bot.py)

|                                                                                    |                             |
|------------------------------------------------------------------------------------|-----------------------------|
| def add_player(date, player_name, goals, assists, clean_sheets)                    | Добавить игрока             |
| def add_club(date, club_name, super_cups, cups, championships, champions_leagues): | Добавить клуб               |
| def parse_name_and_params(parts, num_params):                                      | Проверка параметров         |
| def perform_delete():                                                              | Проверка на удаление записи |
| def get_user_state(user_id):                                                       | Получить игрока             |
| def set_user_state(user_id, state):                                                | Сформировать игрока         |
| def clear_user_state(user_id):                                                     | Очистить поле игрока        |
| def start_help_command(message):                                                   | Приветствие                 |
| def handle_menu_buttons(message):                                                  | Добавить кнопки             |
| def handle_add_club_step(message, state):                                          | Заполнение полей клуба      |
| def random_player(message):                                                        | Случайный игрок             |
| def random_club(message):                                                          | Случайный клуб              |
| def print_player_handler(message):                                                 | Показать игрока             |
| def print_club_handler(message):                                                   | Показать клуб               |
| def save_to_file(message):                                                         | Сохранение в файл           |
| def open_app_command(message):                                                     | Открыть приложение          |

## Для одновременного запуска 2 частей приложения (main.py)

|                         |                       |
|-------------------------|-----------------------|
| def run_flask():        | Запуск приложения     |
| def run_telegram_bot(): | Запуск телеграмм-бота |
| def main():             | Майн-функция          |

## 4 Вид программы

### Часть, связанная с сайтом:

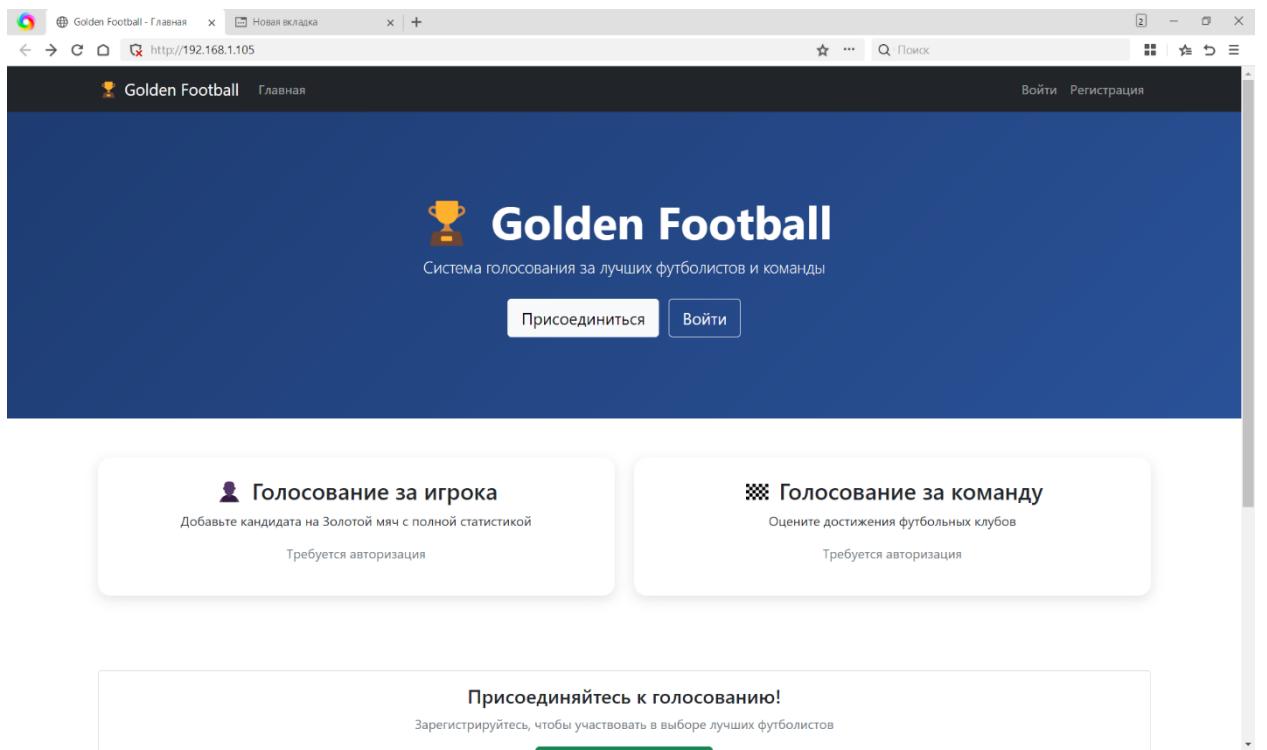
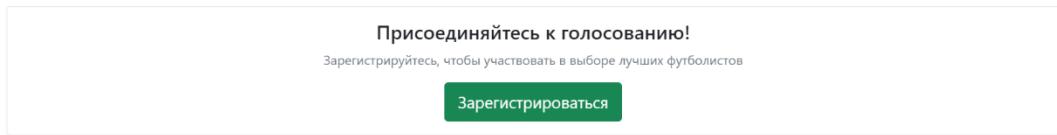


Рисунок 3 – Титульная страница



**О системе Golden Football**

**Возможности для пользователей:**

- Голосование за лучших футболистов
- Оценка достижений футбольных клубов
- Просмотр статистики и результатов
- Участие в определении победителей

**Возможности для организаторов:**

- Управление пользователями системы
- Подсчет результатов голосования
- Назначение наград и победителей
- Администрирование базы данных

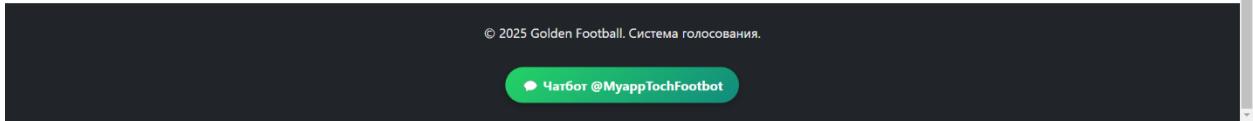


Рисунок 4 – Нижняя часть титульной страницы

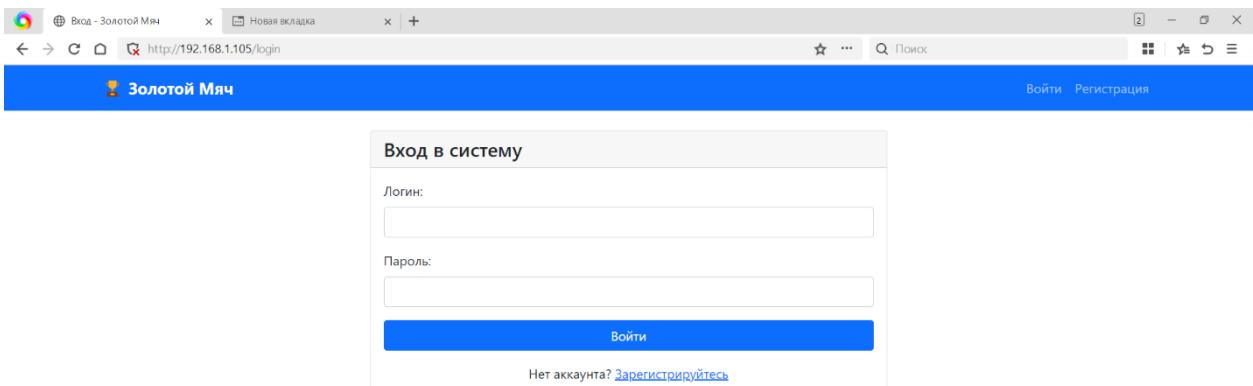


Рисунок 5 – Аутентификация

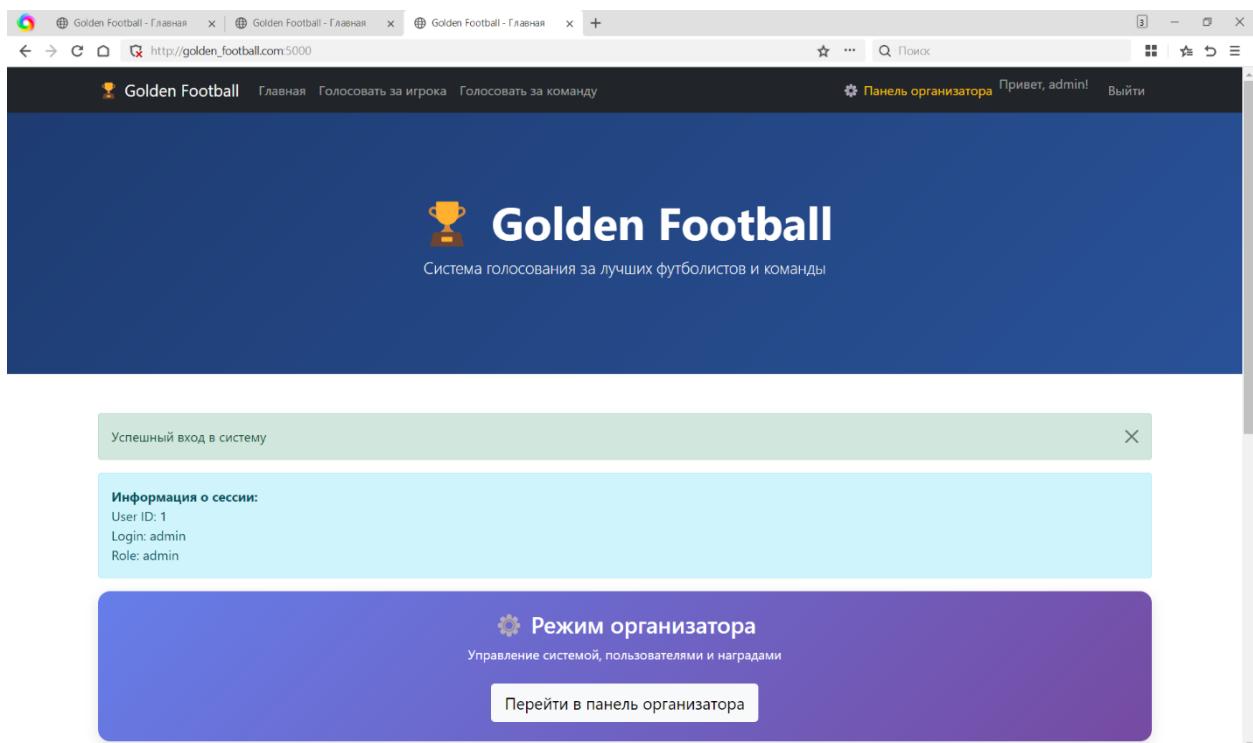


Рисунок 6 – Режим организатора

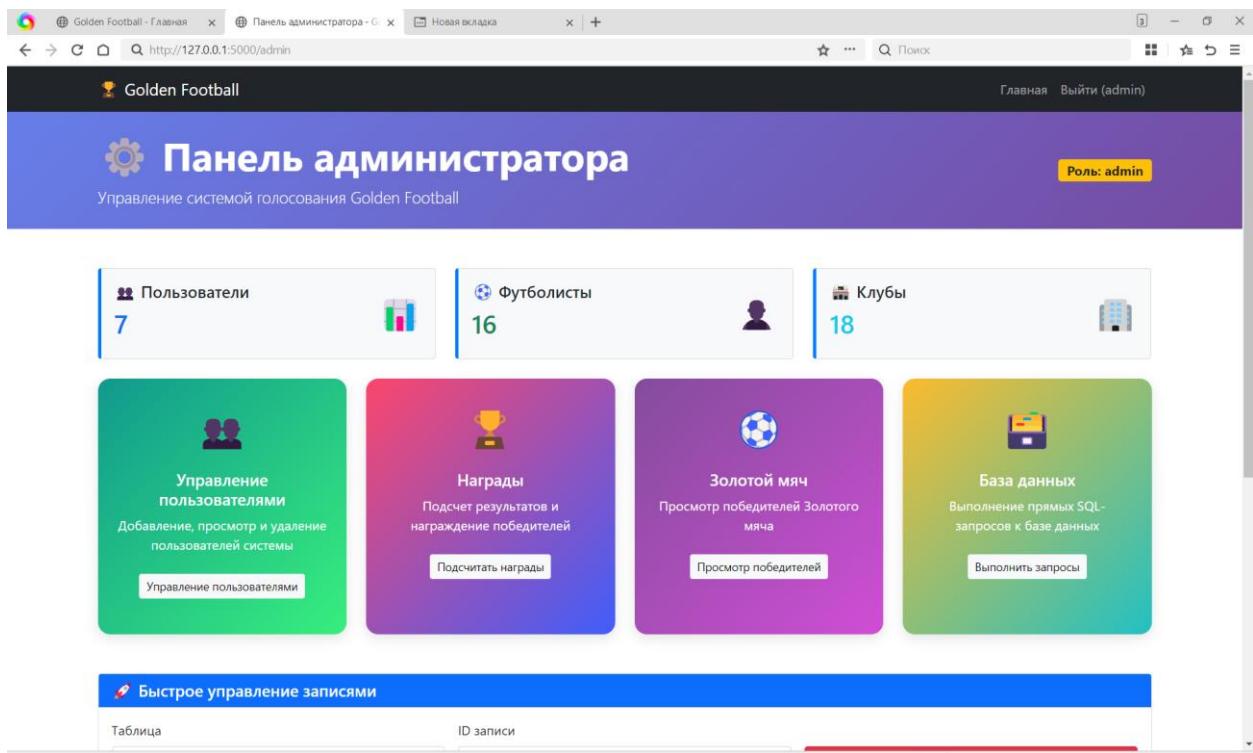


Рисунок 7 – Вид панели администратора

The screenshot shows a web-based application titled 'Golden Football'. The main title bar says 'Golden Football'. Below it, a green header bar contains the text 'Управление пользователями' (User Management) and a purple button labeled 'Добавить нового пользователя' (Add new user). A search bar with placeholder text 'Логин' (Login), a password field, and a dropdown menu for 'Пользователь' (User) are also present. A blue button labeled 'Добавить' (Add) is located at the bottom right of the header.

The main content area has a blue header bar with the text 'Список пользователей' (List of users). Below it is a table with columns: 'ID', 'Логин' (Login), 'Роль' (Role), and 'Действия' (Actions). The table lists seven users:

| ID | Логин   | Роль         | Действия                 |
|----|---------|--------------|--------------------------|
| 1  | admin   | Организатор  | Текущий пользователь     |
| 2  | Ильюха  | Пользователь | <button>Удалить</button> |
| 3  | Антон С | Пользователь | <button>Удалить</button> |
| 4  | Лиза    | Пользователь | <button>Удалить</button> |
| 5  | Саша    | Пользователь | <button>Удалить</button> |
| 6  | Максик  | Пользователь | <button>Удалить</button> |
| 7  | Милана  | Пользователь | <button>Удалить</button> |

At the bottom of the page, there is a black footer bar with the text '© 2025 Golden Football. Управление пользователями.'

Рисунок 8 – Список голосующих

The screenshot shows a web-based application titled 'Golden Football'. The main title bar says 'Golden Football'. Below it, a yellow header bar contains the text 'Победители Золотого мяча' (Golden Ball winners) and a blue button labeled 'История награждений' (History of awards). A search bar with placeholder text 'Победитель' (Winner) is also present.

The main content area has a blue header bar with the text 'Информация' (Information). Below it is a text block stating: 'Победитель Золотого мяча определяется на основе:' followed by a bulleted list:

- Количество голов и ассистов
- Количество "сухих" матчей (для вратарей)
- Побед и ничьих команды
- Джентльменского коэффициента

Below the list, there is a note: 'Для обновления результатов используйте кнопку "Подсчитать награды" в панели организатора.'

At the bottom of the page, there is a black footer bar with the text '© 2025 Golden Football. Золотой мяч.'

Рисунок 9 – Обладатель Золотого мяча

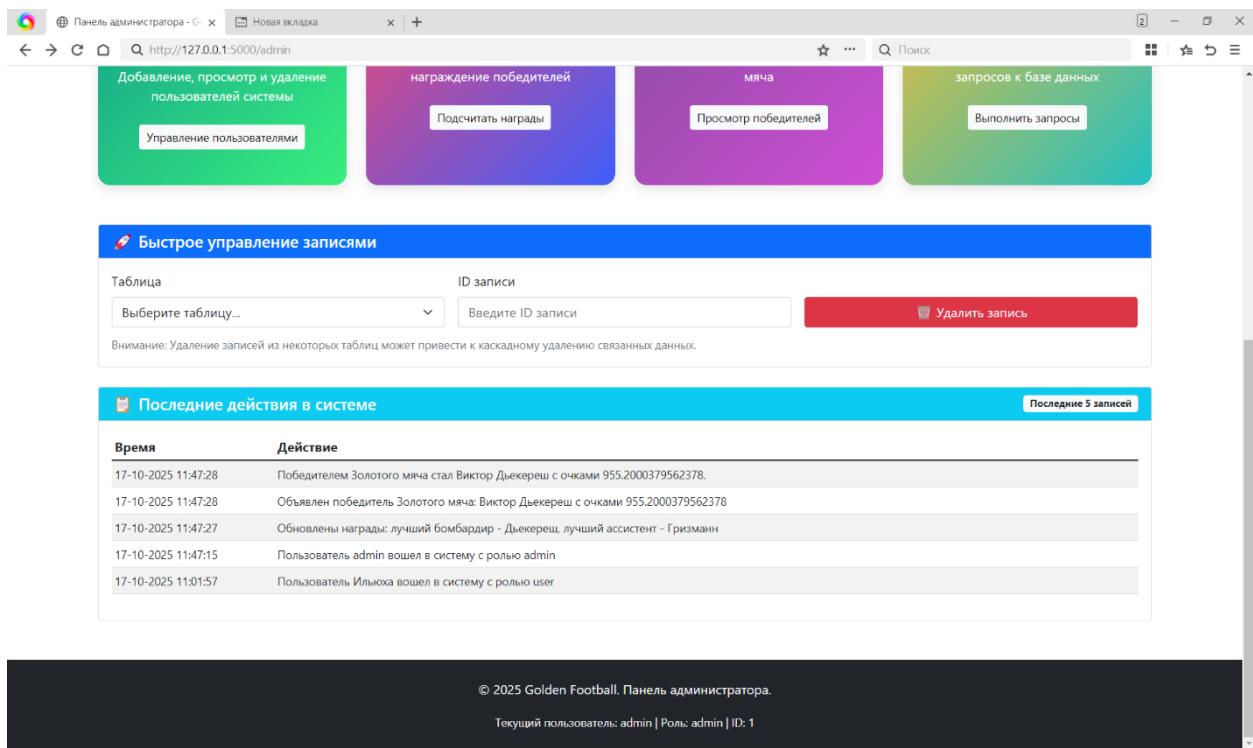


Рисунок 10 – Удалить запись и логирование

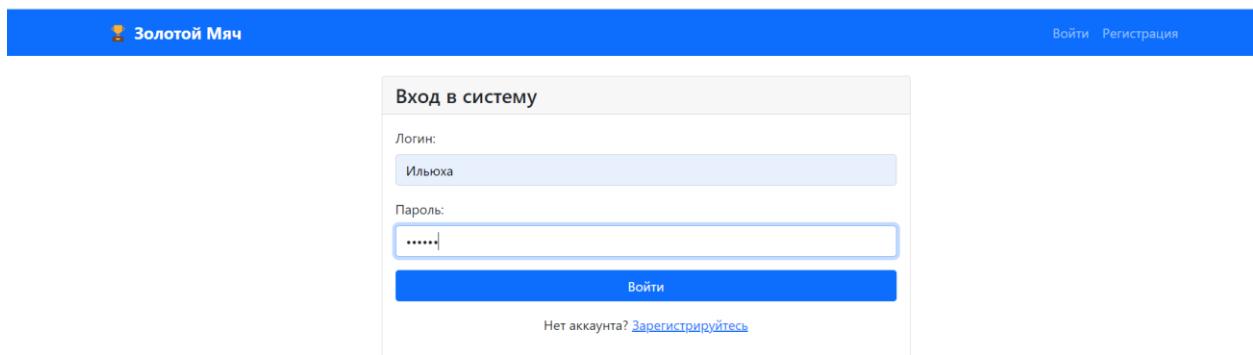


Рисунок 11 – Вход в режим голосующего

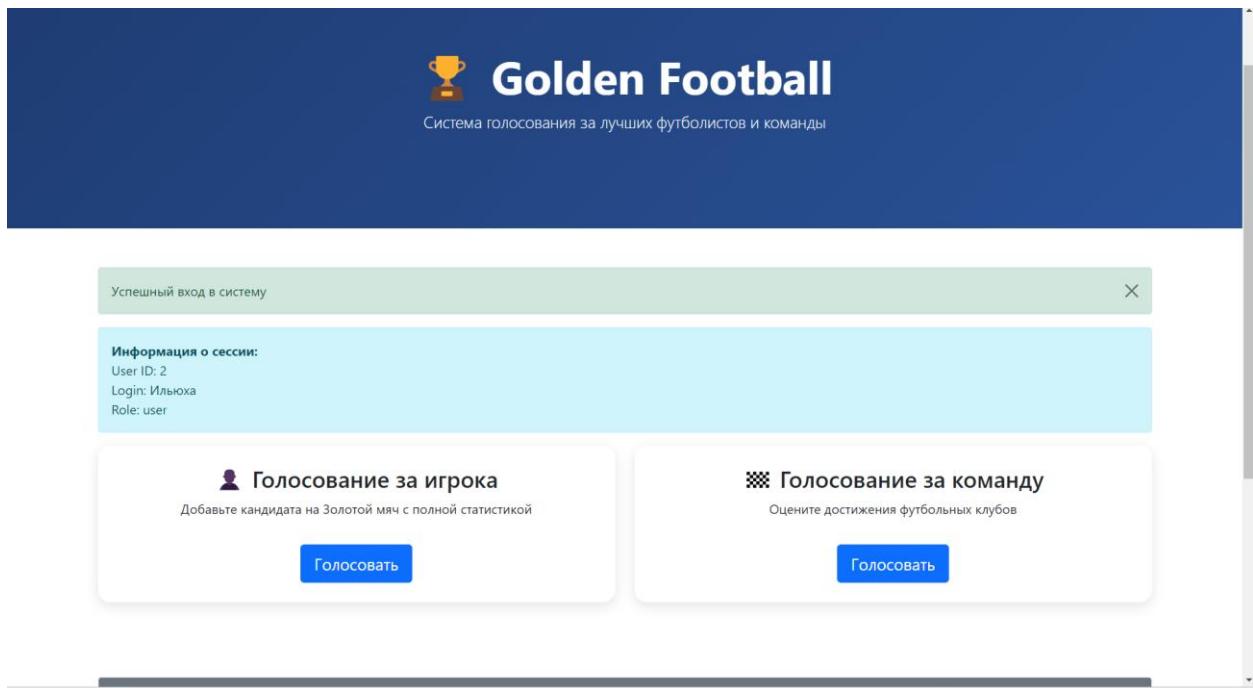


Рисунок 12 – Режим голосующего

This screenshot shows the 'Голосование за лучшего игрока' (Voting for player) page. At the top, it says 'Золотой Мяч' and 'Привет, Ильюха — Выйти'. The form fields include:

- Имя \*: input field
- Фамилия \*: input field
- Возраст \*: input field
- Клуб \*: input field
- Победы: input field (value 0)
- Поражения: input field (value 0)
- Ничьи: input field (value 0)
- Голы: input field (value 0)
- Асисты: input field (value 0)
- Сухие матчи: input field (value 0)
- Джентльменский коэффициент: input field (value 1,0)
- От 1,0 до 5,0
- Сохранить футболиста: blue button
- На главную: grey button

Рисунок 13 – Голосование за игрока

### Голосование за лучшую команду

Название клуба \*

Суперкубки

От 0 до 2

Лиги чемпионов

От 0 до 2

Чемпионаты страны

От 0 до 2

Кубки страны

От 0 до 2

[Сохранить клуб](#)[На главную](#)

Рисунок 14 – Голосование за команду

## Часть, связанная с базой данных:

The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' tree, with 'football' selected, revealing tables like awards, clubs, footballers, etc. The main area shows a query editor with the SQL command 'SELECT \* FROM football.awards;' and a result grid. The result grid contains one row of data:

|   | id   | top_scorer | top_assist |
|---|------|------------|------------|
| ▶ | 13   | Дъекереш   | Гризманн   |
| * | NULL | NULL       | NULL       |

Рисунок 15 – Таблицы в Базе данных и Бомбардир с ассистентом

MySQL Workbench

Local instance ×

File Edit View Query Database Server Tools Scripting Help

Navigator: clubs

SCHEMAS

- football
  - Tables: awards, clubs, footballers, gentleman\_coefficient, golden\_ball, logs, personal\_stats, players, trophies, users
  - Views
  - Stored Procedures
  - Functions
- sakila
- sys
- world

Administration Schemas

Information

Table: clubs

Columns:

|         | <b>id</b> | champion_league | national_championship | cup | super_cup | victories | losses | draws | club_name          |
|---------|-----------|-----------------|-----------------------|-----|-----------|-----------|--------|-------|--------------------|
| clubs 1 | 3         | 1               | 1                     | 1   | 1         | 0         | 0      | 0     | Барселона          |
|         | 5         | 0               | 0                     | 0   | 0         | 32        | 12     | 3     | АльНаср            |
|         | 6         | 0               | 0                     | 0   | 0         | 43        | 32     | 2     | Наполи             |
|         | 8         | 0               | 0                     | 0   | 0         | 32        | 2      | 3     | Манчестер Сити     |
|         | 9         | 0               | 1                     | 1   | 1         | 0         | 0      | 0     | Paris SaintGermain |
|         | 10        | 1               | 2                     | 1   | 1         | 0         | 0      | 0     | Bayern Munich      |
|         | 11        | 1               | 2                     | 1   | 2         | 0         | 0      | 0     | Juventus           |
|         | 12        | 2               | 2                     | 2   | 2         | 45        | 21     | 2     | Челси              |
|         | 13        | 0               | 0                     | 0   | 0         | 32        | 2      | 3     | Арсенал            |
|         | 14        | 0               | 0                     | 0   | 0         | 23        | 23     | 32    | Бенфика            |
|         | 15        | 0               | 0                     | 0   | 0         | 43        | 3      | 2     | Марсель            |
|         | 16        | 0               | 0                     | 0   | 0         | 60        | 20     | 5     | Олимпиакос         |

Output:

Рисунок 16 – Клубы

The screenshot shows the MySQL Workbench interface. In the top navigation bar, 'Local instance' is selected. The 'Navigator' pane on the left lists databases like 'football', 'sakila', 'sys', and 'world'. The 'Tables' section under 'football' contains 'awards', 'clubs', 'footballers', 'gentleman\_coefficient', 'golden\_ball', 'logs', 'personal\_stats', 'players', 'trophies', and 'users'. The 'Information' pane at the bottom shows 'Table: footballers'. The main area displays a query editor with the SQL command 'SELECT \* FROM football.footballers;' and a result grid. The result grid has columns: id, first\_name, last\_name, age, and club. The data is as follows:

|    | id | first_name | last_name   | age | club           |
|----|----|------------|-------------|-----|----------------|
| 1  | 1  | Антуан     | Гризманн    | 34  |                |
| 2  | 2  | Артем      | Дзюба       | 40  |                |
| 3  | 3  | Cristiano  | Ronaldo     | 39  | АльНаср        |
| 4  | 4  | Giovanni   | Di Lorenzo  | 25  | Наполи         |
| 5  | 5  | Robert     | Lewandowski | 31  |                |
| 6  | 6  | Erling     | Haaland     | 23  | Манчестер Сити |
| 7  | 7  | Ламин      | Ямаль       | 19  | Арсенал        |
| 8  | 8  | Бернардо   | Сильва      | 29  | Бенфика        |
| 9  | 9  | Ашраф      | Хакими      | 25  | Марсель        |
| 10 | 10 | Серхио     | Рамос       | 38  | Олимпиакос     |
| 11 | 11 | Кейлор     | Навас       | 32  | Майорка        |
| 12 | 12 | sda        | asd         | 23  | Милан          |

Рисунок 17 – Футболисты

MySQL Workbench

Local instance

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

football

- Tables
  - awards
  - clubs
  - footballers
  - gentleman\_coefficient
  - golden\_ball
  - logs
  - personal\_stats
  - players
  - trophies
  - users
- Views
- Stored Procedures
- Functions

sakila

sys

world

Administration Schemas

Information

Table: gentleman\_coefficient

Columns:

Query 1

players footballers footballers players gentleman\_

1 • SELECT \* FROM football.gentleman\_coefficient;

Result Grid | Filter Rows: Edit: Export/

|   | id | coefficient | footballer          |
|---|----|-------------|---------------------|
| ▶ | 1  | 3           | Антуан Гризманн     |
|   | 2  | 2           | Артем Дзюба         |
|   | 3  | 4.5         | Cristiano Ronaldo   |
|   | 4  | 3           | Giovanni Di Lorenzo |
|   | 5  | 2           | Robert Lewandowski  |
|   | 6  | 5           | Erling Haaland      |
|   | 7  | 3           | Ламин Ямаль         |
|   | 8  | 2           | Бернардо Силва      |
|   | 9  | 3.2         | Ашраф Хакими        |
|   | 10 | 2.8         | Серхио Рамос        |
|   | 11 | 4           | Кейлор Навас        |
|   | 12 | 5           | sda asd             |

Рисунок 18 – Коэффициенты

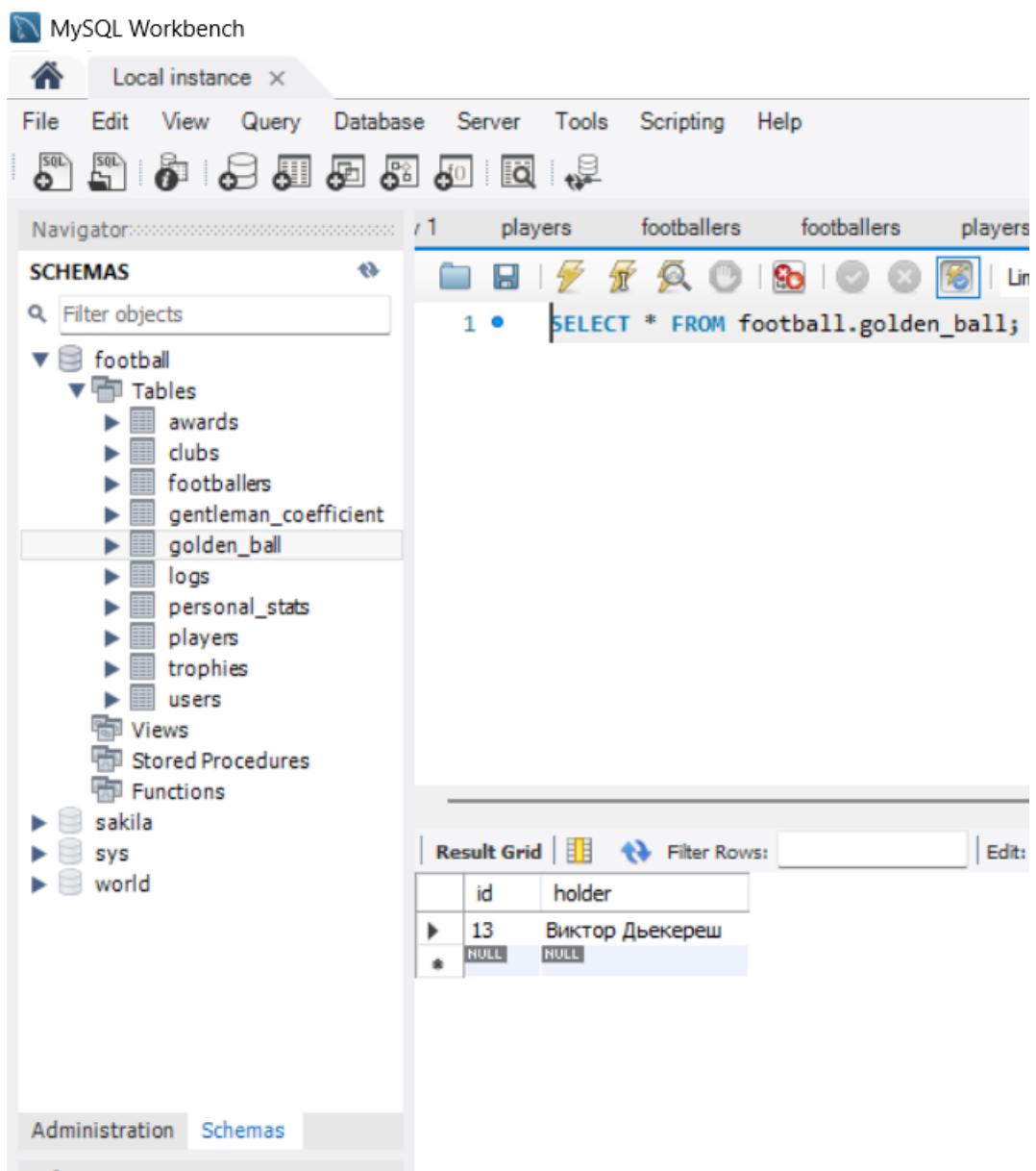


Рисунок 19 – Обладатель “Золотого мяча”

MySQL Workbench

Local instance ×

File Edit View Query Database Server Tools Scripting Help

Navigator: otballers footballers players gentleman\_coefficient trophies logs awards clubs footb

SCHEMAS

Filter objects

football

- Tables
  - awards
  - clubs
  - footballers
  - gentleman\_coefficient
  - golden\_ball
  - logs
  - personal\_stats
  - players
  - trophies
  - users
- Views
- Stored Procedures
- Functions

sakila

sys

world

Administration Schemas

Information

Table: logs

1 • | SELECT \* FROM football.logs;

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: |

|   | id | text                                          |
|---|----|-----------------------------------------------|
| ▶ | 1  | 13-10-2025 16:59:24: Запущено Flask прилож... |
|   | 2  | 13-10-2025 16:59:24: Запущено Flask прилож... |
|   | 3  | 13-10-2025 17:00:25: Зарегистрирован новый... |
|   | 4  | 13-10-2025 17:00:37: Пользователь Ильюха ...  |
|   | 5  | 13-10-2025 17:01:01: Добавлен футболист: ...  |
|   | 6  | 13-10-2025 17:01:22: Добавлен футболист: ...  |
|   | 7  | 13-10-2025 17:01:31: Добавлен клуб: Барсел... |
|   | 8  | 13-10-2025 17:01:34: Добавлен клуб: Зенит,... |
|   | 9  | 13-10-2025 17:01:39: Добавлен клуб: Ливер...  |
|   | 10 | 13-10-2025 17:01:45: Пользователь Ильюха ...  |
|   | 11 | 13-10-2025 17:01:52: Пользователь admin во... |
|   | 12 | 13-10-2025 17:02:02: Обновлены награды: л...  |

Рисунок 20 – Логи

MySQL Workbench

Local instance ×

File Edit View Query Database Server Tools Scripting Help

Navigator: footballers players gentleman\_coefficient trophies logs

SCHEMAS

Filter objects

football

Tables

- awards
- clubs
- footballers
- gentleman\_coefficient
- golden\_ball
- logs
- personal\_stats
- players
- trophies
- users

Views

Stored Procedures

Functions

sakila

sys

world

Administration Schemas

Information

Table: personal\_stats

1 • | SELECT \* FROM football.personal\_stats;

Result Grid | Filter Rows: | Edit: | Export:

|    | id | player_name | goals | assists | clean_sheets |
|----|----|-------------|-------|---------|--------------|
| 1  | 1  | Гризманн    | 23    | 21      | 2            |
| 2  | 2  | Дзюба       | 45    | 2       | 1            |
| 3  | 3  | Ronaldo     | 35    | 3       | 0            |
| 4  | 4  | Di Lorenzo  | 2     | 5       | 12           |
| 5  | 5  | Lewandowski | 48    | 9       | 0            |
| 6  | 6  | Haaland     | 36    | 8       | 0            |
| 7  | 7  | Ямаль       | 32    | 1       | 2            |
| 8  | 8  | Силва       | 5     | 3       | 3            |
| 9  | 9  | Хакими      | 6     | 6       | 0            |
| 10 | 10 | Рамос       | 3     | 3       | 0            |
| 11 | 11 | Навас       | 1     | 0       | 45           |
| 12 | 12 | asd         | 23    | 21      | 2            |

Рисунок 21 – Статистика

MySQL Workbench

Local instance

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

football

- Tables
  - awards
  - clubs
  - footballers
  - gentleman\_coefficient
  - golden\_ball
  - logs
  - personal\_stats
  - players
  - trophies
  - users
- Views
- Stored Procedures
- Functions

sakila

sys

world

Administration Schemas

Information

Table: players

Columns:

|   | id | victories | losses | draws | player_name |
|---|----|-----------|--------|-------|-------------|
| ▶ | 1  | 45        | 23     | 23    | Гризманн    |
|   | 2  | 12        | 56     | 2     | Дзюба       |
|   | 3  | 32        | 12     | 3     | Ronaldo     |
|   | 4  | 43        | 32     | 2     | Di Lorenzo  |
|   | 5  | 67        | 32     | 2     | Lewandowski |
|   | 6  | 32        | 2      | 3     | Haaland     |
|   | 7  | 32        | 2      | 3     | Ямаль       |
|   | 8  | 23        | 23     | 32    | Силва       |
|   | 9  | 43        | 3      | 2     | Хакими      |
|   | 10 | 60        | 20     | 5     | Рамос       |
|   | 11 | 70        | 10     | 5     | Навас       |
|   | 12 | 34        | 2      | 2     | asd         |
|   | 13 | 17        | 17     | 17    | -           |

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: |

1 • SELECT \* FROM football.players;

Limit to 1000 rows

1000 rows

Рисунок 22 – Игроки по победам, ничьим и поражениям

MySQL Workbench

Local instance

File Edit View Query Database Server Tools Scripting Help

Schemas: football (selected), sakila, sys, world

Tables: awards, clubs, footballers, gentleman\_coefficient, golden\_ball, logs, personal\_stats, players, trophies, users

Views, Stored Procedures, Functions

Information: Administration, Schemas

Table: trophies

Columns:

| <b>id</b>   | int AI PK          |                       |
|-------------|--------------------|-----------------------|
| club_name   | varchar(255)       |                       |
| trophy_type | varchar(100)       |                       |
| 1           | Барселона          | super_cup             |
| 2           | Барселона          | champion_league       |
| 3           | Барселона          | national_championship |
| 4           | Барселона          | cup                   |
| 7           | Paris SaintGermain | super_cup             |
| 8           | Paris SaintGermain | national_championship |
| 9           | Paris SaintGermain | cup                   |
| 10          | Bayern Munich      | super_cup             |
| 11          | Bayern Munich      | champion_league       |
| 12          | Bayern Munich      | national_championship |
| 13          | Bayern Munich      | national_championship |
| 14          | Bayern Munich      | cup                   |

Result Grid | Filter Rows: | Edit: | Export/Import: | w

Output:

Рисунок 23 – Трофеи

MySQL Workbench

Local instance X

File Edit View Query Database Server Tools Scripting Help

Navigator: ent trophies logs awards clubs footballers gentleman\_coefficient

SCHEMAS

Filter objects

football

Tables

- awards
- clubs
- footballers
- gentleman\_coefficient
- golden\_ball
- logs
- personal\_stats
- players
- trophies
- users

Views

Stored Procedures

Functions

sakila

sys

world

Administration Schemas

Information

Result Grid | Filter Rows: | Edit: | Export/Import: |

|   | id   | login   | password_hash                               | role  |
|---|------|---------|---------------------------------------------|-------|
| ▶ | 1    | admin   | e042a46d4d1c710f18dbeca2072f96297c4b6be...  | admin |
|   | 2    | Ильюха  | 9bad8ce29f9fcf77699f61fbbb49070aa4bbdbf4... | user  |
|   | 3    | Антон С | ff007c72aabfc7f2ce80f178686c34618e71b451... | user  |
|   | 4    | Лиза    | e3a98205bdfa10a9a5e10324bc1e8674ae9bbc9...  | user  |
|   | 5    | Саша    | 28fd4ee3ea8aa9b9902e0bfc25e00e19c367e06...  | user  |
|   | 6    | Максик  | 5700fe8178e4792d0abbff7a7b862379f4e1f98d... | user  |
|   | 7    | Милана  | c08463bfb1b020211b7abfba012245729d8e308...  | user  |
| * | HULL | HULL    | NULL                                        | NULL  |

Table: users

The screenshot shows the MySQL Workbench interface. The left sidebar displays the database schema with the 'football' database selected, containing tables like awards, clubs, footballers, etc. The main area shows a query editor with the SQL command 'SELECT \* FROM football.users;' and a result grid displaying the contents of the 'users' table. The table has columns: id, login, password\_hash, and role. The data shows 7 rows, including an administrator ('admin') and several regular users ('Ильюха', 'Антон С', 'Лиза', 'Саша', 'Максик', 'Милана'). The 'password\_hash' column contains hashed password values.

Рисунок 24 – Пользователи и администраторы

## Часть, связанная с телеграм-ботом.

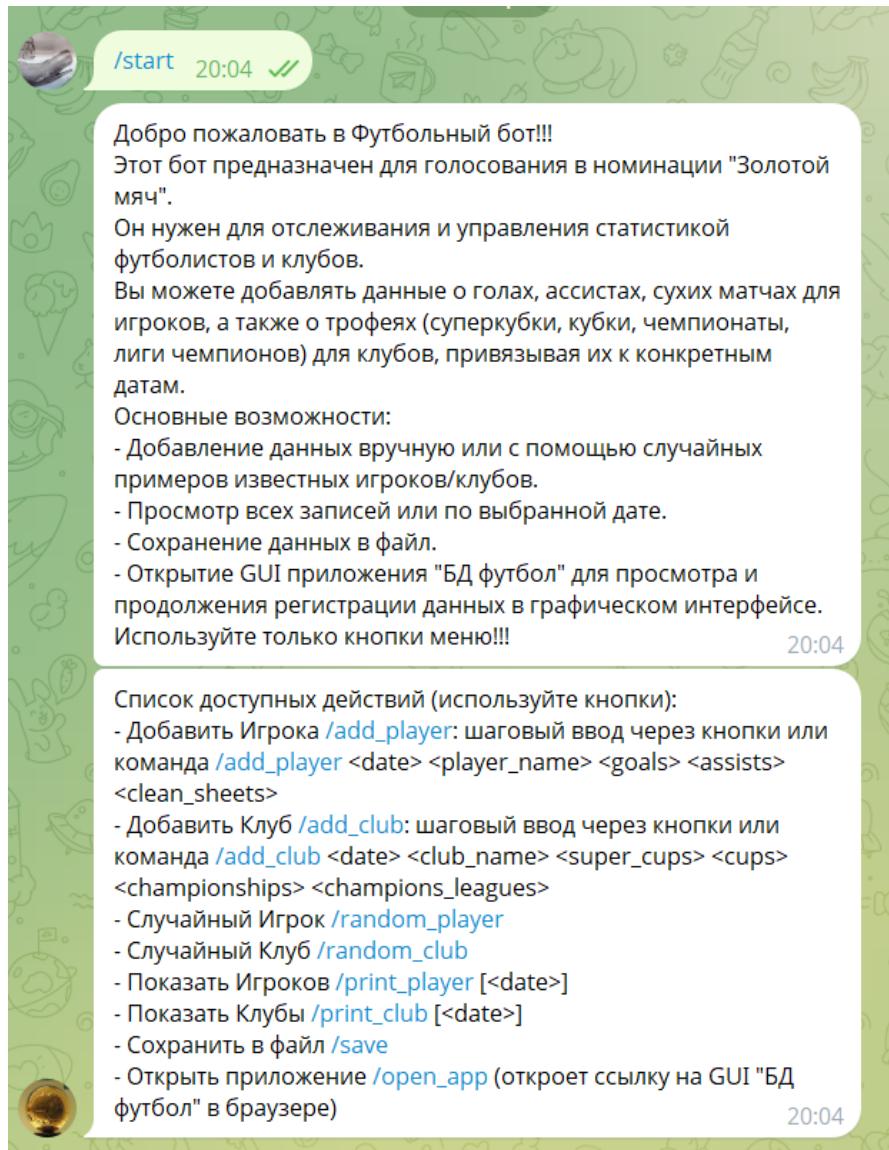


Рисунок 25 – Приветствие

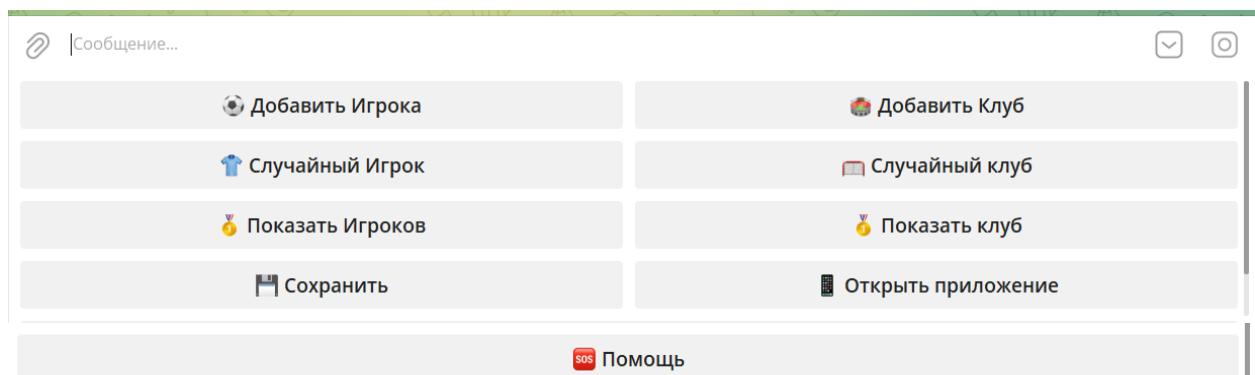


Рисунок 26 – Кнопки и команды



📎 Сообщение...

Рисунок 27 – Добавить игрока

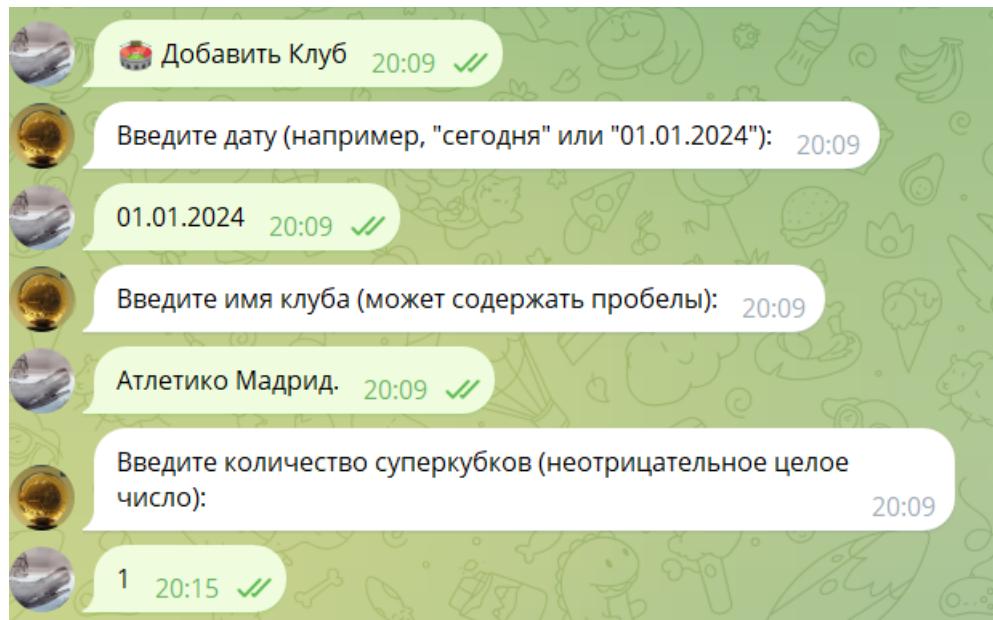


Рисунок 28 – Добавить клуб

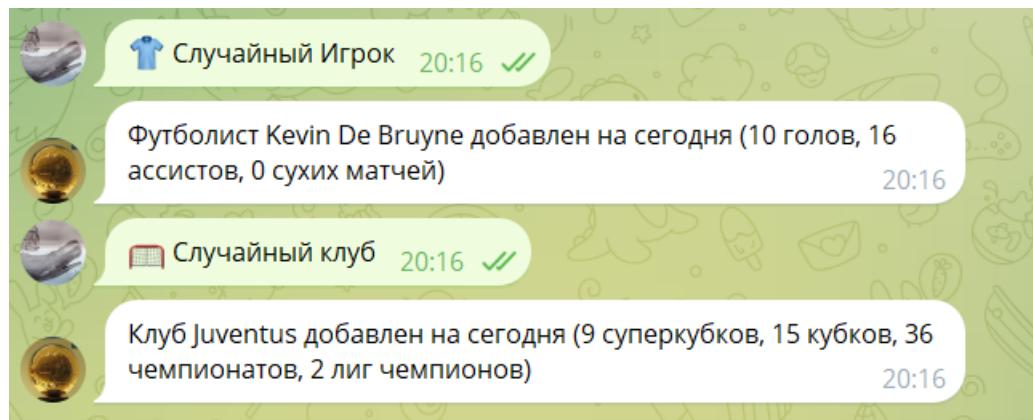


Рисунок 29 – Случайный игрок или клуб

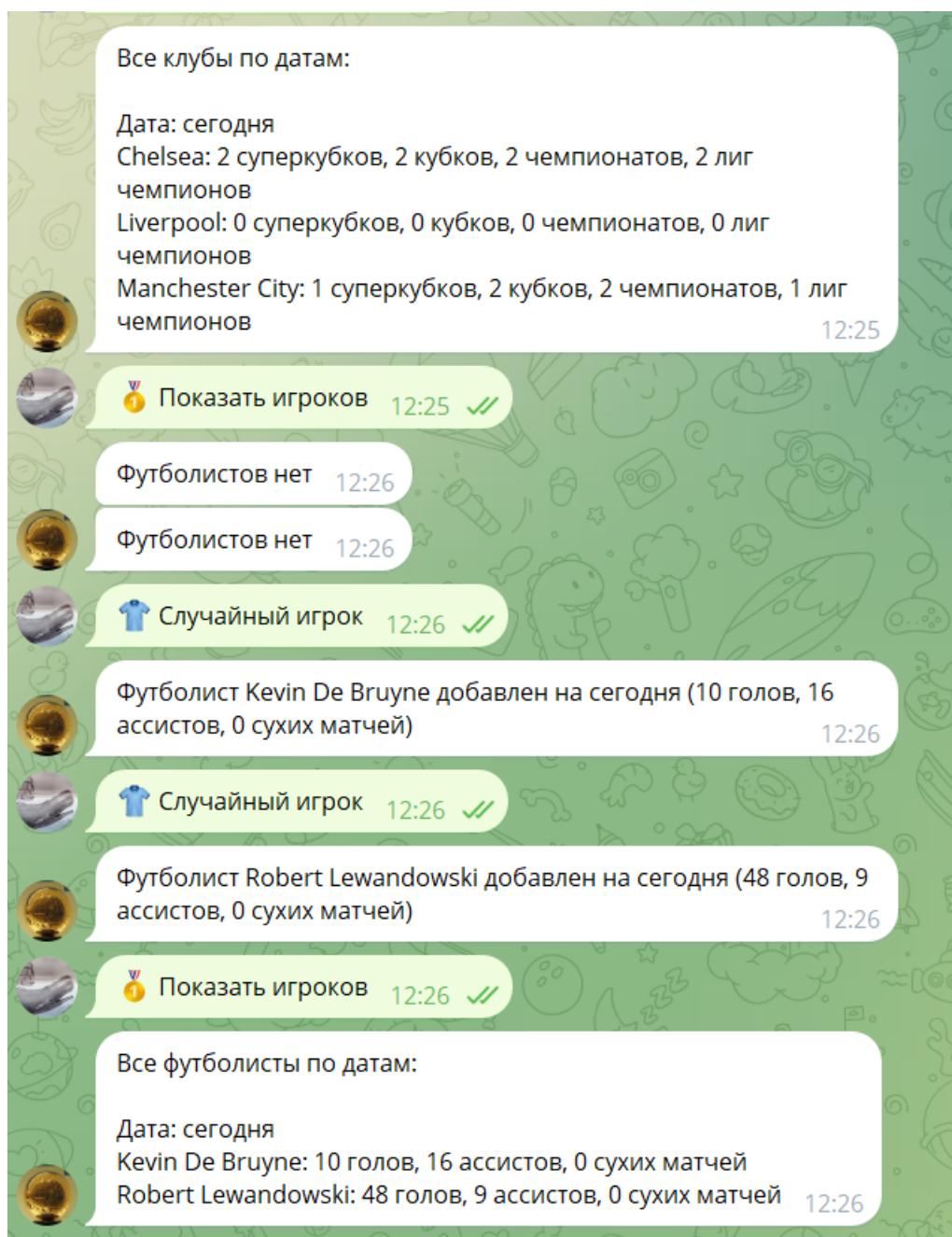


Рисунок 30 – Показать клуб или игрока

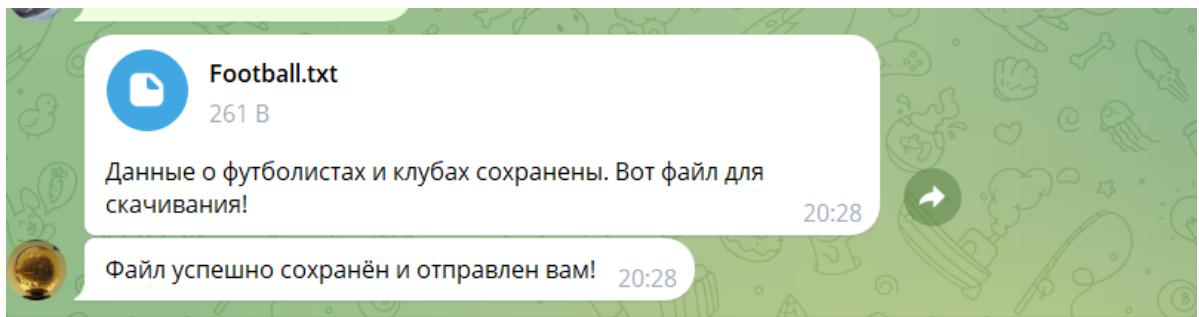


Рисунок 31 – Сохранение в файл

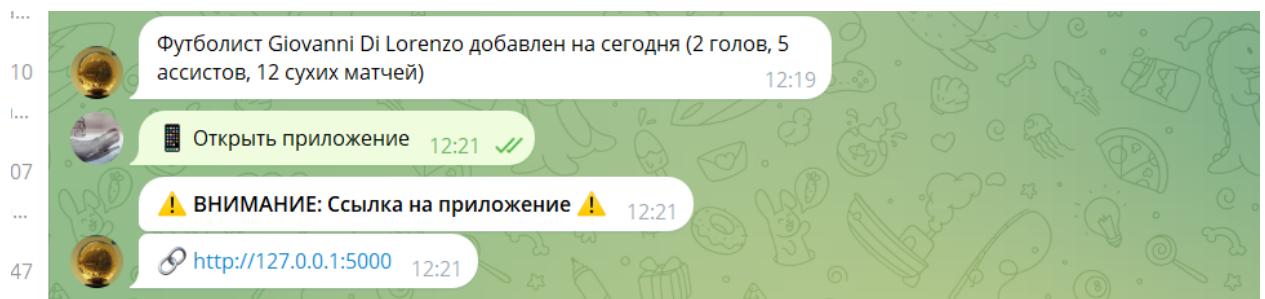


Рисунок 32 – Ссылка на приложение

```
File Изменить Просмотр
==== Players ====
сегодня:
- Giovanni Di Lorenzo: 2 голов, 5 ассистов, 12 сухих матчей
==== Clubs ====
сегодня:
- Chelsea: 2 суперкубков, 2 кубков, 2 чемпионатов, 2 лиг чемпионов
- Liverpool: 0 суперкубков, 0 кубков, 0 чемпионатов, 0 лиг чемпионов
```

Строка 1, столбец 1 | 250 символов | 100% | Windows (CRLF) | UTF-8

Рисунок 33 – Содержимое файла

Windows PowerShell  
(C) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

Установите последнюю версию PowerShell для новых функций и улучшения! <https://aka.ms/PSWindows>

```
PS C:\Users\alexe> ssh -i C:\Users\alexe\.ssh\ssh-key-1760976323804 -l alexeycherevkov 158.160.203.139
Welcome to Ubuntu 24.04.3 LTS (GNU/Linux 6.8.0-85-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

System information as of Mon Oct 20 19:51:26 UTC 2025

System load: 0.0          Processes:           146
Usage of /:   34.4% of 9.04GB  Users logged in:    1
Memory usage: 31%          IPv4 address for eth0: 10.130.0.5
Swap usage:   0%         

* Strictly confined Kubernetes makes edge and IoT secure. Learn how MicroK8
s
just raised the bar for easy, resilient and secure K8s cluster deployment
.

https://ubuntu.com/engage/secure-kubernetes-at-the-edge

Expanded Security Maintenance for Applications is not enabled.

2 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

Last login: Mon Oct 20 19:45:36 2025 from 5.18.215.78
alexeycherevkov@applicationcherevkov:~$ cd my_application
alexeycherevkov@applicationcherevkov:~/my_application$ cd my_project_dir
alexeycherevkov@applicationcherevkov:~/my_application/my_project_dir$ |
```

Рисунок 34 – Проект на виртуальной машине

```
(venv) alexeycherevkov@applicationcherevkov:~/my_application/my_project_dir$ nano main.py
(venv) alexeycherevkov@applicationcherevkov:~/my_application/my_project_dir$ nano app.py
(venv) alexeycherevkov@applicationcherevkov:~/my_application/my_project_dir$ python3 main.py
Traceback (most recent call last):
  File "/home/alexeycherevkov/my_application/my_project_dir/main.py", line 2, in <module>
    import app
  File "/home/alexeycherevkov/my_application/my_project_dir/app.py", line 742
    app.run(host='0.0.0.0'
           ^
SyntaxError: unterminated string literal (detected at line 742)
(venv) alexeycherevkov@applicationcherevkov:~/my_application/my_project_dir$ nano app.py
(venv) alexeycherevkov@applicationcherevkov:~/my_application/my_project_dir$ python3 main.py
Старт Football App and Telegram Bot...
Старт Flask app...
Старт Telegram bot...
База данных Football создана или уже существует.
* Serving Flask app 'app'
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:5000
* Running on http://10.130.0.5:5000
Press CTRL+C to quit
5.18.215.78 - - [20/Oct/2025 17:38:13] "GET / HTTP/1.1" 200 -
5.18.215.78 - - [20/Oct/2025 17:38:14] "GET /favicon.ico HTTP/1.1" 404 -
27.115.124.6 - - [28/Oct/2025 17:38:22] "GET / HTTP/1.1" 200 -
27.115.124.38 - - [28/Oct/2025 17:38:29] "GET /register HTTP/1.1" 200 -
5.18.215.78 - - [28/Oct/2025 17:40:43] "GET /login HTTP/1.1" 200 -
5.18.215.78 - - [28/Oct/2025 17:40:50] "POST /Login HTTP/1.1" 302 -
5.18.215.78 - - [28/Oct/2025 17:40:50] "GET / HTTP/1.1" 200 -
5.18.215.78 - - [28/Oct/2025 17:40:52] "GET /admin HTTP/1.1" 200 -
5.18.215.78 - - [28/Oct/2025 17:40:56] "GET /vote/player HTTP/1.1" 200 -
123.6.49.44 - - [28/Oct/2025 17:40:59] "GET / HTTP/1.1" 200 -
123.6.49.12 - - [28/Oct/2025 17:41:04] "GET /favicon.ico HTTP/1.1" 404 -
123.6.49.15 - - [28/Oct/2025 17:41:04] "GET /favicon.ico HTTP/1.1" 404 -
5.18.215.78 - - [28/Oct/2025 17:41:24] "POST /vote/player HTTP/1.1" 200 -
5.18.215.78 - - [28/Oct/2025 17:41:25] "GET / HTTP/1.1" 200 -
5.18.215.78 - - [28/Oct/2025 17:41:26] "GET /admin HTTP/1.1" 200 -
5.18.215.78 - - [28/Oct/2025 17:41:29] "GET /admin/awards HTTP/1.1" 302 -
5.18.215.78 - - [28/Oct/2025 17:41:29] "GET /admin HTTP/1.1" 200 -
5.18.215.78 - - [28/Oct/2025 17:41:30] "GET /admin/golden_ball HTTP/1.1" 200 -
5.18.215.78 - - [28/Oct/2025 17:41:35] "GET /admin/query HTTP/1.1" 200 -
5.18.215.78 - - [28/Oct/2025 17:41:47] "POST /admin/query HTTP/1.1" 200 -
^C
Прерывание.
Все процессы остановлены.
(venv) alexeycherevkov@applicationcherevkov:~/my_application/my_project_dir$ vim |
```

Рисунок 35 – Запуск на виртуальной машине

## 5 Вывод

В ходе работы была разработана двухзвенная информационная система с использованием **PYTHON 3.12.6**, библиотеки **Flask** и **MySQL Workbench 8.0 CE**. Тестирование программы проводилось на компьютере LENOVO IdeaPad Gaming 3.

Операционная система: Windows 11 Сборка 26100. Информационная система была протестирована в 3 режимах. В перспективе эту базу данных можно внедрить учет результатов за Сборные страны, коэффициент прогресса и стабильности.

## **6 Список источников**

1. Бородина Е.А., Даценко Н.В., Никитин Б.Е., Мачтаков С.Г., Хромых Е.А. Проектирование баз данных. Учебное пособие для подготовки обучающихся по направлениям 09.03.02 - «Информационные системы и технологии», 09.03.03 «Прикладная информатика» / Воронеж, 2023.
2. Бутенко Ю.И. Использование базы данных моделей структурных переводческих трансформаций для извлечения многокомпонентных терминологических единиц // Системы и средства информатики. 2023. Т. 33. № 35-44.
3. Варакута П.С., Козлов Р.К. Имитационное моделирование пропускной способности пулов соединений к базе данных POSTGRESQL // Трибуна ученого. 2022. № 5. С. 48-53.
4. Федоров, Д. Ю. Программирование на языке высокого уровня Python : учебное пособие для прикладного бакалавриата / Д. Ю. Федоров. – 2-е изд., перераб. и доп. – Москва : Издательство Юрайт, 2019. – 161 с. – (Бакалавр. Прикладной курс). – ISBN 978-5-534-10971-9.
5. Шелудько, В. М. Основы программирования на языке высокого уровня Python: учебное пособие / В. М. Шелудько. – Ростов-на-Дону, Таганрог: Издательство Южного федерального университета, 2017. – 146 с. – ISBN 978-5-9275-2649-9.

## 7 Приложение

```
import os
import mysql.connector as mysql
from mysql.connector import Error
from flask import Flask, render_template, request, redirect,
url_for, session, flash
from functools import wraps
import hashlib
from datetime import datetime

app = Flask(__name__)
app.secret_key = os.getenv('SECRET_KEY',
'your_default_secret_key')

# Прямо в коде задаём необходимые параметры подключения к базе
# данных
HOST = 'localhost'          # Адрес сервера баз данных
PORT = 3306                  # Порт подключения
USER = 'root'                # Имя пользователя базы данных
PASSWORD = 'Tochankau110574' # Пароль пользователя
DATABASE_NAME = 'Football'   # Название вашей базы данных

def get_db_connection():
    """
    Функция возвращает соединение с базой данных MySQL.
    Возвращает объект соединения или None в случае неудачи.
    """
    try:
        connection = mysql.connect(
            host=HOST,
            port=PORT,
            user=USER,
            password=PASSWORD,
            database=DATABASE_NAME
        )
        return connection
    except Error as e:
        print(f"Ошибка подключения к базе данных: {e}")
        return None

def test_connection():
    """
    Тестирует работоспособность подключения к базе данных.
    Возвращает True, если успешно подключился, иначе False.
    """
```

```

"""
try:
    conn = get_db_connection()
    if conn and conn.is_connected():
        conn.close()
        return True
    else:
        return False
except Error:
    return False

def login_required(role='user'):
    def decorator(f):
        @wraps(f)
        def decorated_function(*args, **kwargs):
            if 'user_id' not in session:
                flash('Требуется авторизация', 'error')
                return redirect(url_for('login'))
            if role == 'admin' and session.get('role') != 'admin':
                flash('Доступ запрещен. Требуются права администратора.', 'error')
                return redirect(url_for('index'))
            return f(*args, **kwargs)

        return decorated_function

    return decorator

def initialize_database():
    if not test_connection():
        print("Не удалось подключиться к MySQL. Проверьте пароль root и запуск сервера.")
        return False
    try:
        temp_conn = mysql.connect(
            host=os.getenv('MYSQL_HOST', 'localhost'),
            port=int(os.getenv('MYSQL_PORT', 3306)),
            user=os.getenv('MYSQL_USER', 'root'),
            password=os.getenv('MYSQL_PASSWORD',
'Tochankau110574')
        )
        with temp_conn.cursor() as temp_cursor:
            temp_cursor.execute(f"CREATE DATABASE IF NOT EXISTS

```

```

{DATABASE_NAME}")
    temp_conn.close()
    print(f"База данных {DATABASE_NAME} создана или уже
существует.")

    with get_db_connection() as connection:
        with connection.cursor() as cursor:
            tables = [ # Список таблиц
                'CREATE TABLE IF NOT EXISTS
gentleman_coefficient (id INT PRIMARY KEY AUTO_INCREMENT,
coefficient FLOAT NOT NULL, footballer VARCHAR(255) NOT NULL);',
                'CREATE TABLE IF NOT EXISTS golden_ball (id
INT PRIMARY KEY AUTO_INCREMENT, holder VARCHAR(255) NOT NULL);',
                'CREATE TABLE IF NOT EXISTS players (id INT
PRIMARY KEY AUTO_INCREMENT, victories INT NOT NULL DEFAULT 0,
losses INT NOT NULL DEFAULT 0, draws INT NOT NULL DEFAULT 0,
player_name VARCHAR(255) NOT NULL);',
                'CREATE TABLE IF NOT EXISTS clubs (id INT
PRIMARY KEY AUTO_INCREMENT, champion_league INT NOT NULL DEFAULT
0, national_championship INT NOT NULL DEFAULT 0, cup INT NOT
NULL DEFAULT 0, super_cup INT NOT NULL DEFAULT 0, victories INT
NOT NULL DEFAULT 0, losses INT NOT NULL DEFAULT 0, draws INT NOT
NULL DEFAULT 0, club_name VARCHAR(255) NOT NULL);',
                'CREATE TABLE IF NOT EXISTS personal_stats
(id INT PRIMARY KEY AUTO_INCREMENT, player_name VARCHAR(255) NOT
NULL, goals INT NOT NULL DEFAULT 0, assists INT NOT NULL DEFAULT
0, clean_sheets INT NOT NULL DEFAULT 0);',
                'CREATE TABLE IF NOT EXISTS awards (id INT
PRIMARY KEY AUTO_INCREMENT, top_scorer VARCHAR(255) NOT NULL,
top_assist VARCHAR(255) NOT NULL);',
                'CREATE TABLE IF NOT EXISTS trophies (id INT
PRIMARY KEY AUTO_INCREMENT, club_name VARCHAR(255) NOT NULL,
trophy_type VARCHAR(100) NOT NULL);',
                'CREATE TABLE IF NOT EXISTS footballers (id
INT PRIMARY KEY AUTO_INCREMENT, first_name VARCHAR(100) NOT
NULL, last_name VARCHAR(100) NOT NULL, age INT NOT NULL, club
VARCHAR(255) NOT NULL);',
                'CREATE TABLE IF NOT EXISTS logs (id INT
PRIMARY KEY AUTO_INCREMENT, text TEXT NOT NULL);',
                'CREATE TABLE IF NOT EXISTS users (id INT
PRIMARY KEY AUTO_INCREMENT, login VARCHAR(100) NOT NULL UNIQUE,
password_hash VARCHAR(255) NOT NULL, role VARCHAR(50) DEFAULT
\'user\');");
]

```

```

        for table_sql in tables:
            cursor.execute(table_sql)
            cursor.execute("SELECT COUNT(*) FROM users WHERE
login = %s;", ('admin',))
            if cursor.fetchone()[0] == 0:
                admin_password =
hashlib.sha256("Админчик".encode()).hexdigest()
                cursor.execute("INSERT INTO users(login,
password_hash, role) VALUES (%s, %s, %s);",
('admin', admin_password,
'admin'))
                connection.commit()
                return True
            except Error as e:
                print(f"Ошибка инициализации: {e}")
                return False

def encrypt_password(password):
    return hashlib.sha256(password.encode()).hexdigest()

def write_log(text):
    try:
        with get_db_connection() as connection:
            with connection.cursor() as cursor:
                timestamp = datetime.now().strftime("%d-%m-%Y
%H:%M:%S")
                cursor.execute("INSERT INTO logs(text) VALUES
(%s);", (f'{timestamp}: {text}',))
                connection.commit()
    except Error as e:
        print(f"Ошибка записи лога: {e}")

# Маршрут Flask
@app.route('/')
def index():
    current_year = datetime.now().year
    return render_template('index.html',
current_year=current_year)

@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        login_input = request.form['login']
        password = request.form['password']

```

```

try:
    connection = get_db_connection()
    cursor = connection.cursor()
    cursor.execute("SELECT id, password_hash, role FROM
users WHERE login = %s;", (login_input,))
    user = cursor.fetchone()
    cursor.close()
    connection.close()

    if user and user[1] == encrypt_password(password):
        session['user_id'] = user[0]
        session['login'] = login_input
        session['role'] = user[2] # Важно: сохраняем
роль в сессии
        write_log(f"Пользователь {login_input} вошел в
систему с ролью {user[2]}")
        flash('Успешный вход в систему', 'success')
        return redirect(url_for('index'))
    else:
        flash('Неверный логин или пароль', 'error')
except Error as e:
    flash(f'Ошибка подключения к БД: {str(e)}', 'error')
    print(f"Ошибка в login: {e}")

return render_template('login.html')

@app.route('/register', methods=['GET', 'POST'])
def register():
    if request.method == 'POST':
        login_input = request.form['login']
        password = request.form['password']
        confirm_password = request.form['confirm_password']

        if password != confirm_password:
            flash('Пароли не совпадают', 'error')
            return render_template('register.html')

        if len(password) < 4:
            flash('Пароль должен быть не менее 4 символов',
'error')
            return render_template('register.html')

        hashed_password = encrypt_password(password)

```

```

connection = None
cursor = None
try:
    connection = get_db_connection()
    cursor = connection.cursor()
    cursor.execute("INSERT INTO users (login,
password_hash) VALUES (%s, %s)", (login_input, hashed_password))
    connection.commit()
    flash('Регистрация успешна. Теперь вы можете
войти.', 'success')
    write_log(f"Зарегистрирован новый пользователь:
{login_input}")
    return redirect(url_for('login'))
except Error as e:
    if connection:
        connection.rollback()
    if "Duplicate entry" in str(e):
        flash('Пользователь с таким логином уже
существует', 'error')
    else:
        flash('Ошибка регистрации', 'error')
        print(f"Ошибка в register: {e}")
finally:
    if cursor:
        cursor.close()
    if connection:
        connection.close()

return render_template('register.html')

@app.route('/logout')
def logout():
    if 'login' in session:
        write_log(f"Пользователь {session.get('login')} вышел из
системы")
        session.clear()
    flash('Вы вышли из системы', 'success')
    return redirect(url_for('index'))

@app.route('/vote/player', methods=['GET', 'POST'])
@login_required()
def vote_player():
    if request.method == 'POST':

```

```

# Валидация данных
first_name = request.form['first_name'].strip()
last_name = request.form['last_name'].strip()
age = request.form['age'].strip()
club = request.form['club'].strip()
wins = request.form.get('wins', '0').strip()
losses = request.form.get('losses', '0').strip()
draws = request.form.get('draws', '0').strip()
goals = request.form.get('goals', '0').strip()
assists = request.form.get('assists', '0').strip()
clean_sheets = request.form.get('clean_sheets',
'0').strip()
gentleman_coef = request.form.get('gentleman_coef',
'1.0').strip()

# Проверка обязательных полей
if not all([first_name, last_name, age, club]):
    flash('Заполните все обязательные поля', 'error')
    return render_template('vote_player.html')

try:
    # Преобразование и валидация числовых значений
    age_value = int(age)
    wins_value = int(wins) if wins else 0
    losses_value = int(losses) if losses else 0
    draws_value = int(draws) if draws else 0
    goals_value = int(goals) if goals else 0
    assists_value = int(assists) if assists else 0
    clean_sheets_value = int(clean_sheets) if
clean_sheets else 0
    gentleman_coef_value = float(gentleman_coef) if
gentleman_coef else 1.0

    # Проверка диапазонов
    validations = [
        (0 <= age_value <= 100, "Возраст должен быть от
0 до 100 лет"),
        (0 <= wins_value <= 100, "Количество побед
должно быть от 0 до 100"),
        (0 <= losses_value <= 100, "Количество поражений
должно быть от 0 до 100"),
        (0 <= draws_value <= 100, "Количество ничьих
должно быть от 0 до 100"),
        (0 <= goals_value <= 100, "Количество голов
должно быть от 0 до 100")
    ]

```

```

должно быть от 0 до 100"),
(0 <= assists_value <= 100, "Количество ассистов
должно быть от 0 до 100"),
(0 <= clean_sheets_value <= 100, "Количество
сухих матчей должно быть от 0 до 100"),
(1 <= gentleman_coef_value <= 5, "Джентльменский
коэффициент должен быть от 1 до 5")
]

for condition, error_msg in validations:
    if not condition:
        flash(error_msg, 'error')
        return render_template('vote_player.html')

except ValueError:
    flash('Некорректные числовые значения', 'error')
    return render_template('vote_player.html')

# Проверка максимального количества футболистов
connection = None
cursor = None
try:
    connection = get_db_connection()
    cursor = connection.cursor()
    cursor.execute("SELECT COUNT(*) FROM footballers;")
    count = cursor.fetchone()[0]
    if count >= 30:
        flash('Достигнуто максимальное количество
футболистов (30)', 'error')
        return render_template('vote_player.html')

    # Добавление футболиста в базу
    cursor.execute('''INSERT INTO
footballers(first_name, last_name, age, club) VALUES (%s, %s,
%s, %s);''',
                           (first_name, last_name, age_value,
club))
    cursor.execute(
        '''INSERT INTO personal_stats(player_name,
goals, assists, clean_sheets) VALUES (%s, %s, %s, %s);''',
        (last_name, goals_value, assists_value,
clean_sheets_value))
    cursor.execute('''INSERT INTO players(player_name,
victories, losses, draws) VALUES (%s, %s, %s, %s);''',

```

```

        (last_name, wins_value, losses_value,
draws_value))

        # Обновление коэффициента джентльменства
        cursor.execute("INSERT INTO
gentleman_coefficient(coefficient, footballer) VALUES (%s,
%s);",
                           (gentleman_coef_value, f"{first_name}"
{last_name})) 

        # Обновление статистики клуба
        cursor.execute("SELECT id, victories, losses, draws
FROM clubs WHERE club_name = %s;", (club,))
        club_data = cursor.fetchone()
        if club_data:
            club_id, club_victories, club_losses, club_draws
= club_data
            new_victories = club_victories + wins_value
            new_losses = club_losses + losses_value
            new_draws = club_draws + draws_value
            cursor.execute("UPDATE clubs SET victories = %s,
losses = %s, draws = %s WHERE id = %s;",
                           (new_victories, new_losses,
new_draws, club_id))
        else:
            cursor.execute('''INSERT INTO
clubs(champion_league, national_championship, cup, super_cup,
victories, losses, draws, club_name)
VALUES (0, 0, 0, 0, %s, %s, %s,
%s)''', (wins_value, losses_value, draws_value, club))

            connection.commit()
            write_log(f"Добавлен футболист: {first_name}"
{last_name}, клуб: {club}, возраст: {age_value}")
            flash('Футболист успешно добавлен', 'success')

    except Error as e:
        if connection:
            connection.rollback()
            flash(f'Ошибка при добавлении футболиста: {str(e)}',
'e')
            print(f"Ошибка в vote_player: {e}")
    finally:
        if cursor:

```

```

        cursor.close()
    if connection:
        connection.close()

    return render_template('vote_player.html')

@app.route('/vote/team', methods=['GET', 'POST'])
@login_required()
def vote_team():
    if request.method == 'POST':
        club_name = request.form['club_name'].strip()
        super_cup = request.form.get('super_cup', '0').strip()
        champion_league = request.form.get('champion_league',
                                         '0').strip()
        national_championship =
request.form.get('national_championship', '0').strip()
        cup = request.form.get('cup', '0').strip()

        if not club_name:
            flash('Название клуба обязательно для заполнения',
'error')
            return render_template('vote_team.html')

    try:
        super_cup_value = int(super_cup) if super_cup else 0
        champion_league_value = int(champion_league) if
champion_league else 0
        national_championship_value =
int(national_championship) if national_championship else 0
        cup_value = int(cup) if cup else 0

        # Проверка диапазонов
        for val, name in [(super_cup_value, "суперкубков"),
(champion_league_value, "лиг чемпионов"),
(national_championship_value,
"чемпионатов страны"), (cup_value, "кубков")]:
            if val < 0 or val > 2:
                flash(f'Количество {name} должно быть от 0
до 2', 'error')
                return render_template('vote_team.html')

    except ValueError:
        flash('Поля статистики должны быть числовыми',
'error')

```

```

        return render_template('vote_team.html')

# Добавление клуба в базу
connection = None
cursor = None
try:
    connection = get_db_connection()
    cursor = connection.cursor()

    cursor.execute("SELECT id FROM clubs WHERE club_name = %s;", (club_name,))
    club_id_result = cursor.fetchone()

    if club_id_result:
        club_id = club_id_result[0]
        cursor.execute(
            "SELECT super_cup, champion_league,
national_championship, cup FROM clubs WHERE club_name = %s;",
            (club_name,))
        old = cursor.fetchone()
        super_cup_total = old[0] + super_cup_value
        champion_league_total = old[1] +
champion_league_value
        national_championship_total = old[2] +
national_championship_value
        cup_total = old[3] + cup_value
        cursor.execute(
            '''UPDATE clubs SET super_cup=%s,
champion_league=%s, national_championship=%s, cup=%s WHERE
club_name=%s;''',
            (super_cup_total, champion_league_total,
national_championship_total, cup_total, club_name))
    else:
        cursor.execute('''INSERT INTO clubs(super_cup,
champion_league, national_championship, cup, victories, losses,
draws, club_name)
VALUES (%s, %s, %s, %s, 0, 0, 0,
%s);''',
            (super_cup_value,
champion_league_value, national_championship_value, cup_value,
club_name))
        club_id = cursor.lastrowid

# Добавление трофеев

```

```

        trophy_map = {
            "super_cup": super_cup_value,
            "champion_league": champion_league_value,
            "national_championship":
national_championship_value,
            "cup": cup_value
        }
        for trophy_type, count in trophy_map.items():
            for _ in range(count):
                cursor.execute('INSERT INTO
trophies(club_name, trophy_type) VALUES (%s, %s);',
                           (club_name, trophy_type))

        connection.commit()
        write_log(
            f"Добавлен клуб: {club_name}, Суперкубки:
{super_cup_value}, Лига Чемпионов: {champion_league_value},
Чемпионат: {national_championship_value}, Кубок: {cup_value}")
            flash('Клуб успешно добавлен', 'success')

    except Error as e:
        if connection:
            connection.rollback()
        flash(f'Ошибка при добавлении клуба: {str(e)}',
              'error')
        print(f"Ошибка в vote_team: {e}")
    finally:
        if cursor:
            cursor.close()
        if connection:
            connection.close()

    return render_template('vote_team.html')

@app.route('/admin')
@login_required('admin')
def admin_panel():
    try:
        connection = get_db_connection()
        cursor = connection.cursor()

        # Получаем статистику
        cursor.execute("SELECT COUNT(*) FROM users;")
        users_count = cursor.fetchone()[0]

```

```

cursor.execute("SELECT COUNT(*) FROM footballers;")
players_count = cursor.fetchone()[0]

cursor.execute("SELECT COUNT(*) FROM clubs;")
clubs_count = cursor.fetchone()[0]

# Получаем последние логи (последние 5 записей)
cursor.execute("SELECT text FROM logs ORDER BY id DESC
LIMIT 5;")
logs_data = cursor.fetchall()

cursor.close()
connection.close()

# Парсим логи для отображения
recent_logs = []
for log in logs_data:
    if ':' in log[0]:
        timestamp, message = log[0].split(':', 1)
        recent_logs.append({'timestamp': timestamp,
'message': message})
    else:
        recent_logs.append({'timestamp': 'N/A',
'message': log[0]})

return render_template('admin_panel.html',
                      users_count=users_count,
                      players_count=players_count,
                      clubs_count=clubs_count,
                      recent_logs=recent_logs)

except Error as e:
    flash(f'Ошибка загрузки панели: {str(e)}', 'error')
    print(f"Ошибка в admin_panel: {e}")
    return redirect(url_for('index'))

@app.route('/admin/users')
@login_required('admin')
def admin_users():
    try:
        connection = get_db_connection()
        cursor = connection.cursor()
        cursor.execute("SELECT id, login, role FROM users;")
        users = cursor.fetchall()

```

```

        cursor.close()
        connection.close()
        return render_template('admin_users.html', users=users)
    except Error as e:
        flash(f'Ошибка загрузки пользователей: {str(e)}',
        'error')
        print(f"Ошибка в admin_users: {e}")
        return redirect(url_for('admin_panel'))

@app.route('/admin/add_user', methods=['POST'])
@login_required('admin')
def admin_add_user():
    login = request.form['login']
    password = request.form['password']
    role = request.form.get('role', 'user')

    if not login or not password:
        flash('Введите логин и пароль', 'error')
        return redirect(url_for('admin_users'))

    if len(password) < 4:
        flash('Пароль должен быть не менее 4 символов', 'error')
        return redirect(url_for('admin_users'))

    hashed = encrypt_password(password)
    connection = None
    cursor = None
    try:
        connection = get_db_connection()
        cursor = connection.cursor()
        cursor.execute("INSERT INTO users(login, password_hash,
role) VALUES (%s, %s, %s);", (login, hashed, role))
        connection.commit()
        flash('Пользователь успешно добавлен', 'success')
        write_log(f"Администратор добавил пользователя:
{login}")
    except Error as e:
        if connection:
            connection.rollback()
        if "Duplicate entry" in str(e):
            flash('Такой логин уже существует', 'error')
        else:
            flash(f'Ошибка добавления пользователя: {str(e)}',
            'error')

```

```

        print(f"Ошибка в admin_add_user: {e}")
    finally:
        if cursor:
            cursor.close()
        if connection:
            connection.close()

    return redirect(url_for('admin_users'))

@app.route('/admin/delete_user/<int:user_id>')
@login_required('admin')
def admin_delete_user(user_id):
    if user_id == session.get('user_id'):
        flash('Нельзя удалить самого себя', 'error')
        return redirect(url_for('admin_users'))

    connection = None
    cursor = None
    try:
        connection = get_db_connection()
        cursor = connection.cursor()
        cursor.execute("DELETE FROM users WHERE id = %s;",
(user_id,))
        connection.commit()
        flash('Пользователь удален', 'success')
        write_log(f"Администратор удалил пользователя с ID:
{user_id}")
    except Error as e:
        if connection:
            connection.rollback()
        flash(f'Ошибка удаления пользователя: {str(e)}',
'error')
        print(f"Ошибка в admin_delete_user: {e}")
    finally:
        if cursor:
            cursor.close()
        if connection:
            connection.close()

    return redirect(url_for('admin_users'))

@app.route('/admin/awards')
@login_required('admin')
def admin_awards():

```

```

    calculate_awards_and_winner()
    flash('Награды и победитель Золотого мяча обновлены',
'success')
    return redirect(url_for('admin_panel'))

@app.route('/admin/golden_ball')
@login_required('admin')
def admin_golden_ball():
    try:
        connection = get_db_connection()
        cursor = connection.cursor()
        cursor.execute("SELECT holder FROM golden_ball;")
        holders = cursor.fetchall()
        cursor.close()
        connection.close()
        return render_template('admin_golden_ball.html',
holders=holders)
    except Error as e:
        flash(f'Ошибка загрузки Золотого мяча: {str(e)}',
'error')
        print(f"Ошибка в admin_golden_ball: {e}")
        return redirect(url_for('admin_panel'))

@app.route('/admin/query', methods=['GET', 'POST'])
@login_required('admin')
def admin_query():
    if request.method == 'POST':
        query = request.form['query']
        if not query:
            flash('Введите SQL-запрос', 'error')
            return render_template('admin_query.html')

        connection = None
        cursor = None
        try:
            connection = get_db_connection()
            cursor = connection.cursor()
            cursor.execute(query)
            if query.lower().strip().startswith('select'):
                rows = cursor.fetchall()
                headers = [description[0] for description in
cursor.description] if cursor.description else []
                result = {
                    'headers': headers,

```

```

        'rows': rows
    }
    write_log(f"Администратор выполнил запрос:
{query}")
    return render_template('admin_query.html',
result=result)
else:
    connection.commit()
    flash('Запрос выполнен успешно', 'success')
    write_log(f"Администратор выполнил запрос:
{query}")
except Error as e:
    if connection:
        connection.rollback()
    flash(f'Ошибка выполнения запроса: {str(e)}',
'error')
    print(f"Ошибка в admin_query: {e}")
finally:
    if cursor:
        cursor.close()
    if connection:
        connection.close()

return render_template('admin_query.html')

@app.route('/admin/delete_record', methods=['POST'])
@login_required('admin')
def admin_delete_record():
    table = request.form['table']
    record_id = request.form['record_id']

    if not record_id.isdigit():
        flash('ID записи должен быть числом', 'error')
        return redirect(url_for('admin_panel'))

    record_id = int(record_id)
    connection = None
    cursor = None
    try:
        connection = get_db_connection()
        cursor = connection.cursor()

        cursor.execute(f"SELECT COUNT(*) FROM {table} WHERE id =
%s;", (record_id,))

```

```

        count = cursor.fetchone()[0]
        if count == 0:
            flash(f'Запись с id={record_id} не найдена в таблице {table}', 'error')
            return redirect(url_for('admin_panel'))

        if table == "footballers":
            cursor.execute("SELECT last_name, first_name, club
FROM footballers WHERE id = %s;", (record_id,))
            player_data = cursor.fetchone()
            if player_data:
                last_name, first_name, club = player_data
                cursor.execute("DELETE FROM footballers WHERE id = %s;", (record_id,))
                cursor.execute("DELETE FROM personal_stats WHERE
player_name = %s;", (last_name,))
                cursor.execute("DELETE FROM players WHERE
player_name = %s;", (last_name,))
                full_name = f'{first_name} {last_name}'
                cursor.execute("DELETE FROM
gentleman_coefficient WHERE footballer = %s;", (full_name,))
            elif table == "clubs":
                cursor.execute("SELECT club_name FROM clubs WHERE id = %s;", (record_id,))
                club_data = cursor.fetchone()
                if club_data:
                    club_name = club_data[0]
                    cursor.execute("DELETE FROM clubs WHERE id = %s;", (record_id,))
                    cursor.execute("DELETE FROM trophies WHERE
club_name = %s;", (club_name,))
                    cursor.execute("UPDATE footballers SET club=''
WHERE club = %s;", (club_name,))
                else:
                    cursor.execute(f"DELETE FROM {table} WHERE id = %s;", (record_id,))

            connection.commit()
            flash(f'Запись с id={record_id} из таблицы {table}
успешно удалена', 'success')
            write_log(f"Удалена запись id={record_id} из таблицы
{table}")

except Error as e:

```

```

        if connection:
            connection.rollback()
        flash(f'Не удалось удалить запись: {str(e)}', 'error')
        print(f"Ошибка в admin_delete_record: {e}")
    finally:
        if cursor:
            cursor.close()
    if connection:
        connection.close()

    return redirect(url_for('admin_panel'))

def calculate_awards_and_winner():
    connection = None
    cursor = None
    try:
        connection = get_db_connection()
        cursor = connection.cursor()

        cursor.execute("DELETE FROM awards;")
        cursor.execute("DELETE FROM golden_ball;")

        # Найти лучшего бомбардира (игрок с максимальным goals)
        cursor.execute("SELECT player_name FROM personal_stats
WHERE goals = (SELECT MAX(goals) FROM personal_stats) LIMIT 1;")
        top_scorer_data = cursor.fetchone()
        top_scorer = top_scorer_data[0] if top_scorer_data else
None

        # Найти лучшего ассистента (игрок с максимальным
assists)
        cursor.execute("SELECT player_name FROM personal_stats
WHERE assists = (SELECT MAX(assists) FROM personal_stats) LIMIT
1;")
        top_assist_data = cursor.fetchone()
        top_assist = top_assist_data[0] if top_assist_data else
None

        if top_scorer and top_assist:
            cursor.execute("INSERT INTO awards(top_scorer,
top_assist) VALUES (%s, %s);", (top_scorer, top_assist))
            write_log(f"Обновлены награды: лучший бомбардир -
{top_scorer}, лучший ассистент - {top_assist}")

```

```

query = '''
        SELECT f.first_name, f.last_name, f.club,
               ps.goals, ps.assists, ps.clean_sheets,
               p.victories, p.draws, p.losses,
               c.victories, c.draws, c.losses,
               COALESCE(fc.coefficient, 1.0) as
gentleman_coef
        FROM footballers f
        JOIN personal_stats ps ON ps.player_name =
f.last_name
        JOIN players p ON p.player_name = f.last_name
        JOIN clubs c ON c.club_name = f.club
        LEFT JOIN gentleman_coefficient fc ON fc.footballer
= CONCAT(f.first_name, ' ', f.last_name)
        '''

cursor.execute(query)
players = cursor.fetchall()

best_score = None
best_player = None
for player in players:
    (first_name, last_name, club,
     goals, assists, clean_sheets,
     p_victories, p_draws, p_losses,
     c_victories, c_draws, c_losses,
     gentleman_coef_val) = player

    score = (goals + assists + clean_sheets +
             c_victories + c_draws +
             p_victories + p_draws -
             c_losses - p_losses) * gentleman_coef_val

    if best_score is None or score > best_score:
        best_score = score
        best_player = f"{first_name} {last_name}"

if best_player:
    cursor.execute("INSERT INTO golden_ball(holder)
VALUES (%s);", (best_player,))
    write_log(f"Объявлен победитель Золотого мяча:
{best_player} с очками {best_score}")
    write_log(f"Победителем Золотого мяча стал
{best_player} с очками {best_score}.")

```

```

        connection.commit()

    except Error as e:
        if connection:
            connection.rollback()
        print(f"Ошибка в calculate_awards_and_winner: {e}")
    finally:
        if cursor:
            cursor.close()
        if connection:
            connection.close()

if __name__ == '__main__':
    success = initialize_database()
    if success:
        print("Flask приложение инициализировано и запущено.")
        app.run(host='0.0.0.0', port=5000, debug=True) # Запуск
на всех интерфейсах
    else:
        print("Инициализация БД failed. Приложение не
запустится.")
        exit(1)

```

```

from random import choice
from telebot.types import ReplyKeyboardMarkup, KeyboardButton,
InputFile
import telebot
import os

token = '8315061997:AAFEeHeoS16xB119HDNk5AMQwCKeZ64Y1ek'
bot = telebot.TeleBot(token)

GUI_APP_PATH = 'http://127.0.0.1:5000'

RANDOM_TASKS_PLAYERS = [
    {'name': 'Erling Haaland', 'goals': 36, 'assists': 8,
'clean_sheets': 0},
    {'name': 'Giovanni Di Lorenzo', 'goals': 2, 'assists': 5,
'clean_sheets': 12},
    {'name': 'Kylian Mbappé', 'goals': 44, 'assists': 10,
'clean_sheets': 0},
    {'name': 'Lionel Messi', 'goals': 20, 'assists': 15,
'clean_sheets': 0},

```

```

        {'name': 'Cristiano Ronaldo', 'goals': 35, 'assists': 3,
'clean_sheets': 0},
        {'name': 'Virgil van Dijk', 'goals': 1, 'assists': 2,
'clean_sheets': 20},
        {'name': 'Kevin De Bruyne', 'goals': 10, 'assists': 16,
'clean_sheets': 0},
        {'name': 'Robert Lewandowski', 'goals': 48, 'assists': 9,
'clean_sheets': 0}
]

RANDOM_TASKS_CLUBS = [
    {'name': 'Manchester City', 'super_cups': 1, 'cups': 2,
'championships': 2, 'champions_leagues': 1},
    {'name': 'Real Madrid', 'super_cups': 1, 'cups': 2,
'championships': 1, 'champions_leagues': 1},
    {'name': 'Bayern Munich', 'super_cups': 1, 'cups': 1,
'championships': 0, 'champions_leagues': 0},
    {'name': 'Paris Saint-Germain', 'super_cups': 1, 'cups': 1,
'championships': 1, 'champions_leagues': 0},
    {'name': 'Liverpool', 'super_cups': 0, 'cups': 0,
'championships': 0, 'champions_leagues': 0},
    {'name': 'Juventus', 'super_cups': 2, 'cups': 1,
'championships': 2, 'champions_leagues': 1},
    {'name': 'Chelsea', 'super_cups': 2, 'cups': 2,
'championships': 2, 'champions_leagues': 2},
    {'name': 'Barcelona', 'super_cups': 1, 'cups': 0,
'championships': 1, 'champions_leagues': 0}
]

WELCOME = '''
Добро пожаловать в Футбольный бот!!!
Этот бот предназначен для голосования в номинации "Золотой мяч".
Он нужен для отслеживания и управления статистикой футболистов и
клубов.
Вы можете добавлять данные о голах, ассистах, сухих матчах для
игроков, а также о трофеях (суперкубки, кубки, чемпионаты, лиги
чемпионов) для клубов, привязывая их к конкретным датам.
Основные возможности:
- Добавление данных вручную или с помощью случайных примеров
известных игроков/клубов.
- Просмотр всех записей или по выбранной дате.
- Сохранение данных в файл.
- Открытие GUI приложения "БД футбол" для просмотра и
продолжения регистрации данных в графическом интерфейсе.

```

```

Используйте только кнопки меню!!!
'''

players = dict() # date -> list of {'name': str, 'goals': int,
'assists': int, 'clean_sheets': int}
clubs = dict() # date -> list of {'name': str, 'super_cups': int,
'cups': int, 'championships': int, 'champions_leagues': int}

user_states = {}

MAIN_MENU = ReplyKeyboardMarkup(resize_keyboard=True)
MAIN_MENU.add(KeyboardButton('⚽ Добавить игрока'),
KeyboardButton('👤 Добавить клуб'))
MAIN_MENU.add(KeyboardButton('🎲 Случайный игрок'),
KeyboardButton('🏀 Случайный клуб'))
MAIN_MENU.add(KeyboardButton('👤 Показать игроков'),
KeyboardButton('🏆 Показать клуб'))
MAIN_MENU.add(KeyboardButton('💾 Сохранить'), KeyboardButton('📱 Открыть приложение'))
MAIN_MENU.add(KeyboardButton('🆘 Помощь'))

HELP = '''
Список доступных действий (используйте кнопки):
- Добавить Игрока /add_player: шаговый ввод через кнопки или команда /add_player <date> <player_name> <goals> <assists> <clean_sheets>
- Добавить Клуб /add_club: шаговый ввод через кнопки или команда /add_club <date> <club_name> <super_cups> <cups> <championships> <champions_leagues>
- Случайный Игрок /random_player
- Случайный Клуб /random_club
- Показать Игроков /print_player [<date>]
- Показать Клубы /print_club [<date>]
- Сохранить в файл /save
- Открыть приложение /open_app (откроет ссылку на GUI "БД футбол" в браузере)
'''


def add_player(date, player_name, goals, assists, clean_sheets):
    date = date.lower().strip()
    if not date:
        raise ValueError("Дата не может быть пустой")
    if date not in players:

```

```

        players[date] = []
    players[date].append({'name': player_name, 'goals': goals,
'assists': assists, 'clean_sheets': clean_sheets})

def add_club(date, club_name, super_cups, cups, championships,
champions_leagues):
    date = date.lower().strip()
    if not date:
        raise ValueError("Дата не может быть пустой")
    if date not in clubs:
        clubs[date] = []
    clubs[date].append({'name': club_name, 'super_cups': super_cups,
'cups': cups, 'championships': championships,
'champions_leagues': champions_leagues})

def parse_name_and_params(parts, num_params):
    if len(parts) < 2 + num_params:
        return None, None
    params = parts[-num_params:]
    name_parts = parts[2:-num_params]
    name = ' '.join(name_parts).strip()
    if not name:
        return None, None
    try:
        param_values = [int(p) for p in params]
        if any(param < 0 for param in param_values):
            return None, None
    except ValueError:
        return None, None
    return name, param_values

def get_user_state(user_id):
    return user_states.get(user_id, {})

def set_user_state(user_id, state):
    user_states[user_id] = state

def clear_user_state(user_id):
    if user_id in user_states:
        del user_states[user_id]

@bot.message_handler(commands=['start', 'help'])
def start_help_command(message):
    bot.send_message(message.chat.id, WELCOME,

```

```

    reply_markup=MAIN_MENU)
        bot.send_message(message.chat.id, HELP,
    reply_markup=MAIN_MENU)

@bot.message_handler(func=lambda message: True)
def handle_menu_buttons(message):
    user_id = message.from_user.id
    text = message.text.strip()

    if text == '🆘 Помощь':
        bot.send_message(message.chat.id, HELP,
    reply_markup=MAIN_MENU)
        return

    if text == '⚽ Добавить игрока':
        set_user_state(user_id, {'action': 'add_player', 'step': 'date'})
        bot.send_message(message.chat.id, 'Введите дату
(например, "сегодня" или "01.01.2024")')
        return

    if text == '🏀 Добавить клуб':
        set_user_state(user_id, {'action': 'add_club', 'step': 'date'})
        bot.send_message(message.chat.id, 'Введите дату
(например, "сегодня" или "01.01.2024")')
        return

    if text == '🏆 Случайный игрок':
        player_data = choice(RANDOM_TASKS_PLAYERS)
        add_player('сегодня', player_data['name'],
player_data['goals'], player_data['assists'],
player_data['clean_sheets'])
        bot.send_message(message.chat.id, f'Футболист
{player_data["name"]} добавлен на сегодня
({player_data["goals"]}) голов, {player_data["assists"]}
ассистов, {player_data["clean_sheets"]} сухих матчей',
    reply_markup=MAIN_MENU)
        return

    if text == '📃 Случайный клуб':
        club_data = choice(RANDOM_TASKS_CLUBS)
        add_club('сегодня', club_data['name'],
club_data['super_cups'], club_data['cups'],

```

```

club_data['championships'], club_data['champions_leagues'])
    bot.send_message(message.chat.id, f'Клуб
{club_data["name"]} добавлен на сегодня
({club_data["super_cups"]}) суперкубков, ({club_data["cups"]})
кубков, {club_data["championships"]} чемпионатов,
{club_data["champions_leagues"]} лиг чемпионов',
reply_markup=MAIN_MENU)
    return

if text == '👤 Показать игроков':
    if not players:
        output = 'Футболистов нет'
    else:
        output = "Все футболисты по датам:\n\n"
        for date in sorted(players.keys()):
            output += f"Дата: {date}\n"
            for p in players[date]:
                output += f'{p["name"]}: {p["goals"]} голов,
{p["assists"]} ассистов, {p["clean_sheets"]} сухих матчей\n'
                output += '\n'
    bot.send_message(message.chat.id, output,
reply_markup=MAIN_MENU)
    return

if text == '🏆 Показать клуб':
    if not clubs:
        output = 'Клубов нет'
    else:
        output = "Все клубы по датам:\n\n"
        for date in sorted(clubs.keys()):
            output += f"Дата: {date}\n"
            for c in clubs[date]:
                output += f'{c["name"]}: {c["super_cups"]}
суперкубков, {c["cups"]} кубков, {c["championships"]}
чемпионатов, {c["champions_leagues"]} лиг чемпионов\n'
                output += '\n'
    bot.send_message(message.chat.id, output,
reply_markup=MAIN_MENU)
    return

if text == '💾 Сохранить':
    save_to_file(message)
    return

```

```

    if text == '📱 Открыть приложение':
        bot.send_message(message.chat.id, "⚠ <b>ВНИМАНИЕ:</b>\nСсылка на приложение</b> ⚠", parse_mode='HTML')
        bot.send_message(message.chat.id, f"🔗 {GUI_APP_PATH}", reply_markup=MAIN_MENU)
        return

    state = get_user_state(user_id)
    if not state:
        bot.send_message(message.chat.id, 'Неизвестная команда.\nИспользуйте кнопки меню.', reply_markup=MAIN_MENU)
        return

    if state['action'] == 'add_player':
        handle_add_player_step(message, state)
    elif state['action'] == 'add_club':
        handle_add_club_step(message, state)

def handle_add_player_step(message, state):
    user_id = message.from_user.id
    text = message.text.strip()

    if state['step'] == 'date':
        if not text:
            bot.send_message(message.chat.id, 'Дата не может быть пустой. Введите дату:')
            return
        state['date'] = text.lower()
        state['step'] = 'name'
        bot.send_message(message.chat.id, 'Введите имя футболиста (может содержать пробелы):')

    elif state['step'] == 'name':
        if not text:
            bot.send_message(message.chat.id, 'Имя не может быть пустым. Введите имя:')
            return
        state['name'] = text
        state['step'] = 'goals'
        bot.send_message(message.chat.id, 'Введите количество голов (неотрицательное целое число):')

    elif state['step'] == 'goals':
        try:
            goals = int(text)
            if goals < 0:

```

```

        raise ValueError
    state['goals'] = goals
    state['step'] = 'assists'
    bot.send_message(message.chat.id, 'Введите
количество ассистов (неотрицательное целое число):')
except ValueError:
    bot.send_message(message.chat.id, 'Неверное
значение. Введите неотрицательное целое число для голов:')
    return
elif state['step'] == 'assists':
    try:
        assists = int(text)
        if assists < 0:
            raise ValueError
        state['assists'] = assists
        state['step'] = 'clean_sheets'
        bot.send_message(message.chat.id, 'Введите
количество сухих матчей (неотрицательное целое число):')
    except ValueError:
        bot.send_message(message.chat.id, 'Неверное
значение. Введите неотрицательное целое число для ассистов:')
        return
    elif state['step'] == 'clean_sheets':
        try:
            clean_sheets = int(text)
            if clean_sheets < 0:
                raise ValueError
            add_player(state['date'], state['name'],
state['goals'], state['assists'], clean_sheets)
            bot.send_message(message.chat.id, f'Футболист
"{state["name"]}" добавлен на дату {state["date"]}
({state["goals"]}) голов, {state["assists"]} ассистов,
{clean_sheets} сухих матчей', reply_markup=MAIN_MENU)
            clear_user_state(user_id)
        except ValueError as e:
            bot.send_message(message.chat.id, f'Ошибка:
{str(e)}')
            clear_user_state(user_id)

def handle_add_club_step(message, state):
    user_id = message.from_user.id
    text = message.text.strip()

    if state['step'] == 'date':

```

```

    if not text:
        bot.send_message(message.chat.id, 'Дата не может
быть пустой. Введите дату:')
        return
    state['date'] = text.lower()
    state['step'] = 'name'
    bot.send_message(message.chat.id, 'Введите имя клуба
(может содержать пробелы):')
elif state['step'] == 'name':
    if not text:
        bot.send_message(message.chat.id, 'Имя не может быть
пустым. Введите имя:')
        return
    state['name'] = text
    state['step'] = 'super_cups'
    bot.send_message(message.chat.id, 'Введите количество
суперкубков (неотрицательное целое число):')
elif state['step'] == 'super_cups':
    try:
        super_cups = int(text)
        if super_cups < 0:
            raise ValueError
        state['super_cups'] = super_cups
        state['step'] = 'cups'
        bot.send_message(message.chat.id, 'Введите
количество кубков (неотрицательное целое число):')
    except ValueError:
        bot.send_message(message.chat.id, 'Неверное
значение. Введите неотрицательное целое число для суперкубков:')
        return
elif state['step'] == 'cups':
    try:
        cups = int(text)
        if cups < 0:
            raise ValueError
        state['cups'] = cups
        state['step'] = 'championships'
        bot.send_message(message.chat.id, 'Введите
количество чемпионатов (неотрицательное целое число):')
    except ValueError:
        bot.send_message(message.chat.id, 'Неверное
значение. Введите неотрицательное целое число для кубков:')
        return
elif state['step'] == 'championships':

```

```

try:
    championships = int(text)
    if championships < 0:
        raise ValueError
    state['championships'] = championships
    state['step'] = 'champions_leagues'
    bot.send_message(message.chat.id, 'Введите количество лиг чемпионов (неотрицательное целое число):')
except ValueError:
    bot.send_message(message.chat.id, 'Неверное значение. Введите неотрицательное целое число для чемпионатов:')
    return
elif state['step'] == 'champions_leagues':
    try:
        champions_leagues = int(text)
        if champions_leagues < 0:
            raise ValueError
        add_club(state['date'], state['name'],
state['super_cups'], state['cups'], state['championships'],
champions_leagues)
        bot.send_message(message.chat.id, f'Клуб {state["name"]} добавлен на дату {state["date"]}'
({state["super_cups"]}) суперкубков, {state["cups"]} кубков,
{state["championships"]} чемпионатов, {champions_leagues} лиг
чемпионов)', reply_markup=MAIN_MENU)
        clear_user_state(user_id)
    except ValueError as e:
        bot.send_message(message.chat.id, f'Ошибка: {str(e)}')
        clear_user_state(user_id)

@bot.message_handler(commands=['random_player'])
def random_player(message):
    player_data = choice(RANDOM_TASKS_PLAYERS)
    add_player('сегодня', player_data['name'],
player_data['goals'], player_data['assists'],
player_data['clean_sheets'])
    bot.send_message(message.chat.id, f'Футболист {player_data["name"]} добавлен на сегодня
({player_data["goals"]} голов, {player_data["assists"]} ассистов, {player_data["clean_sheets"]} сухих матчей)',
reply_markup=MAIN_MENU)

@bot.message_handler(commands=['random_club'])

```

```

def random_club(message):
    club_data = choice(RANDOM_TASKS_CLUBS)
    add_club('сегодня', club_data['name'],
club_data['super_cups'], club_data['cups'],
club_data['championships'], club_data['champions_leagues'])
    bot.send_message(message.chat.id, f'Клуб {club_data["name"]}'
добавлен на сегодня {club_data["super_cups"]} суперкубков,
{club_data["cups"]} кубков, {club_data["championships"]}
чемпионатов, {club_data["champions_leagues"]} лиг чемпионов',
reply_markup=MAIN_MENU)

@bot.message_handler(commands=['add_player'])
def add_player_handler(message):
    parts = message.text.split()
    if len(parts) < 6:
        bot.send_message(message.chat.id, "Неправильный формат.
Используйте: /add_player <date> <player_name> <goals> <assists>
<clean_sheets>")
        return
    date = parts[1].lower().strip()
    if not date:
        bot.send_message(message.chat.id, "Дата не может быть
пустой!")
        return
    name, params = parse_name_and_params(parts, 3)
    if name is None or len(params) != 3:
        bot.send_message(message.chat.id, "Неправильный формат.
Укажите имя, затем три неотрицательных целых числа (goals,
assists, clean_sheets).")
        return
    goals, assists, clean_sheets = params
    try:
        add_player(date, name, goals, assists, clean_sheets)
        bot.send_message(message.chat.id, f'Футболист "{name}"
добавлен на дату {date} ({goals} голов, {assists} ассистов,
{clean_sheets} сухих матчей)', reply_markup=MAIN_MENU)
    except ValueError as e:
        bot.send_message(message.chat.id, f'Ошибка: {str(e)}')

@bot.message_handler(commands=['add_club'])
def add_club_handler(message):
    parts = message.text.split()
    if len(parts) < 7:
        bot.send_message(message.chat.id, "Неправильный формат.

```

```

Используйте: /add_club <date> <club_name> <super_cups> <cups>
<championships> <champions_leagues>")
    return
    date = parts[1].lower().strip()
    if not date:
        bot.send_message(message.chat.id, "Дата не может быть
пустой!")
        return
    name, params = parse_name_and_params(parts, 4)
    if name is None or len(params) != 4:
        bot.send_message(message.chat.id, "Неправильный формат.
Укажите имя, затем четыре неотрицательных целых числа
(super_cups, cups, championships, champions_leagues).")
        return
    super_cups, cups, championships, champions_leagues = params
try:
    add_club(date, name, super_cups, cups, championships,
champions_leagues)
    bot.send_message(message.chat.id, f'Клуб "{name}"'
добавлен на дату {date} ({super_cups} суперкубков, {cups} кубков, {championships} чемпионатов, {champions_leagues} лиг
чемпионов)', reply_markup=MAIN_MENU)
except ValueError as e:
    bot.send_message(message.chat.id, f'Ошибка: {str(e)}')

@bot.message_handler(commands=['print_player'])
def print_player_handler(message):
    parts = message.text.split()
    date_input = parts[1].lower().strip() if len(parts) > 1 else
None
    if date_input is not None and not date_input:
        bot.send_message(message.chat.id, "Дата не может быть
пустой!")
        return
    is_specific_date = date_input is not None

    if is_specific_date:
        date = date_input
        if date in players and players[date]:
            output = f"Футболисты на дату {date}:\n"
            for p in players[date]:
                output += f'{p["name"]}: {p["goals"]} голов,
{p["assists"]} ассистов, {p["clean_sheets"]} сухих матчей\n'
            else:

```

```

        output = f'Футболистов на дату {date} нет'
    else:
        if not players:
            output = 'Футболистов нет'
        else:
            output = "Все футболисты по датам:\n\n"
            for date in sorted(players.keys()):
                output += f"Дата: {date}\n"
                for p in players[date]:
                    output += f'{p["name"]}: {p["goals"]} голов,' \
{p["assists"]} ассистов, {p["clean_sheets"]} сухих матчей\n'
                    output += '\n'
            bot.send_message(message.chat.id, output,
reply_markup=MAIN_MENU)

@bot.message_handler(commands=['print_club'])
def print_club_handler(message):
    parts = message.text.split()
    date_input = parts[1].lower().strip() if len(parts) > 1 else
None
    if date_input is not None and not date_input:
        bot.send_message(message.chat.id, "Дата не может быть
пустой!")
        return
    is_specific_date = date_input is not None

    if is_specific_date:
        date = date_input
        if date in clubs and clubs[date]:
            output = f"Клубы на дату {date}:\n"
            for c in clubs[date]:
                output += f'{c["name"]}: {c["super_cups"]}'
            supercups, {c["cups"]} кубков, {c["championships"]}
            чемпионатов, {c["champions_leagues"]} лиг чемпионов\n'
        else:
            output = f'Клубов на дату {date} нет'
    else:
        if not clubs:
            output = 'Клубов нет'
        else:
            output = "Все клубы по датам:\n\n"
            for date in sorted(clubs.keys()):
                output += f"Дата: {date}\n"
                for c in clubs[date]:

```

```

        output += f'{c["name"]}: {c["super_cups"]}'
    суперкубков, {c["cups"]} кубков, {c["championships"]}
    чемпионатов, {c["champions_leagues"]} лиг чемпионов\n'
        output += '\n'
    bot.send_message(message.chat.id, output,
reply_markup=MAIN_MENU)

def save_to_file(message):
    try:
        with open('Football.txt', 'w', encoding='utf-8') as f:
            f.write("== Players ==\n")
            for date, plist in sorted(players.items()):
                f.write(f"\n{date}:\n")
                for p in plist:
                    f.write(f" - {p['name']}: {p['goals']}\
голов, {p['assists']} ассистов, {p['clean_sheets']} сухих
матчей\n")
            f.write("\n== Clubs ==\n")
            for date, clist in sorted(clubs.items()):
                f.write(f"\n{date}:\n")
                for c in clist:
                    f.write(f" - {c['name']}: {c['super_cups']}\
суперкубков, {c['cups']} кубков, {c['championships']}
чемпионатов, {c['champions_leagues']} лиг чемпионов\n")

        if os.path.exists('Football.txt') and
os.path.getsize('Football.txt') > 0:
            with open('Football.txt', 'rb') as file:
                try:
                    document = InputFile(file,
filename='Football.txt')
                    bot.send_document(message.chat.id, document,
caption='Данные о футболистах и клубах сохранены. Вот файл для
скачивания!')
                except Exception:
                    file.seek(0)
                    bot.send_document(message.chat.id, file,
caption='Данные о футболистах и клубах сохранены. Вот файл для
скачивания!')
                bot.send_message(message.chat.id, 'Файл успешно
сохранён и отправлен вам!', reply_markup=MAIN_MENU)
        else:
            bot.send_message(message.chat.id, 'Ошибка: файл не
создался корректно.', reply_markup=MAIN_MENU)

```

```

        except Exception as e:
            bot.send_message(message.chat.id, f'Ошибка при
сохранении: {str(e)}', reply_markup=MAIN_MENU)

def start_bot():
    """Функция для запуска polling. Вызывайте ее только в
основном скрипте."""
    bot.polling(none_stop=True, timeout=60) # timeout для
стабильности

if __name__ == '__main__':
    start_bot()

from multiprocessing import Process
import app
import telegram_bot

# Запуск всех приложений
def run_flask():
    print("Старт Flask app...")
    try:
        if app.initialize_database():
            app.app.run(host="0.0.0.0", port=5000, debug=False)
        else:
            print("База данных не инициализирована!")
    except Exception as e:
        print(f"Flask error: {e}")

def run_telegram_bot():
    print("Старт Telegram bot...")
    try:
        telegram_bot.start_bot()
    except Exception as e:
        print(f"Ошибка телеграм-бота: {e}")

if __name__ == '__main__':
    print("Старт Football App and Telegram Bot...")

    flask_process = Process(target=run_flask)
    telegram_process = Process(target=run_telegram_bot)

    flask_process.start()
    telegram_process.start()

```

```
try:  
    flask_process.join()  
    telegram_process.join()  
except KeyboardInterrupt:  
    print("\nПрерывание.")  
    flask_process.terminate()  
    telegram_process.terminate()  
    flask_process.join()  
    telegram_process.join()  
    print("Все процессы остановлены.")
```