МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение высшего образования

«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ   
АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

КАФЕДРА № 44

КУРСОВАЯ РАБОТА   
ЗАЩИЩЕНА С ОЦЕНКОЙ

РУКОВОДИТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ст.преп. |  |  |  | А.В. Аксенов |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К КУРСОВОЙ РАБОТЕ |
| Веб-приложение «сайт для спортсменов» |
| по дисциплине: БАЗЫ ДАННЫХ |
|  |
|  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ гр. № | 4243 |  |  |  | В.С. Суковатицин |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2024

**СОДЕРЖАНИЕ**

ВВЕДЕНИЕ…………………………………………………………….…………3

Техническое задание…………………………………………….……………….4

Проектирование базы данных…………………………………………………...7

Описание выбранных технологий реализации, развертывания и методов взаимодействия с базой данных…………………………………………………13

Листинги модулей……………………………………………………………….16

Тестирование функционала…………………………………………………..…49

ЗАКЛЮЧЕНИЕ………………………………………………………………….55

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ…………………………….56

**ВВЕДЕНИЕ**

Данная работа предназначена для закрепления учебного материала, изученного по курсу «Базы данных».

Веб-приложение представляет собой программу с подключенной базой данных. Взаимодействие происходит через браузер.

Цель курсовой работы – выработать и закрепить навыки проектирования реляционных баз данных на основе требований и ограничений предметной области, а также освоение принципов использования баз данных в качестве хранилищ данных для приложений. Задачей курсовой работы является разработка приложения, осуществляющего взаимодействие с базой данных для выполнения своих функций.

Кроме этого, курсовая работа предназначена для приобретения навыков оформления сопроводительной документации к разработанному программному средству.

**1 Тема курсовой работы**

Веб-приложение «сайт для спортсменов»

**2 Словесное описание предметной области и актуальность**

Данное приложение будет полезно, как и начинающим, так и опытным спортсменам или даже тренерам. С его помощью спортсмены смогут записывать свои результаты, отслеживать свой прогресс и сравнивать себя с силовыми стандартами. Тренеры могут выкладывать свои тренировочные программы и советы спортсменам.

**3 Описание данных, хранящихся в базе данных**

База данных должна содержать данные о:

* Спортсменах, зарегистрировавшихся в системе, и их данных (пол, возраст, рост, вес, ссылки на соцсети).
* Тренерах, зарегистрировавшихся в системе, и их данных (пол, возраст, рост, вес, ссылки на соцсети).
* О результатах выполнения упражнений (дата и время, результат упражнения)
* О тренировочных программах и советах (заголовок, текст, рейтинг, комментарии)
* Об упражнениях (название, техника выполнения)
* Нормативы упражнений

**4 Роли пользователей приложения**

* Спортсмен
* Тренер

**5 Развернутое описание функционала приложения для каждой из ролей**

Система доступна только для зарегистрированных пользователей. Система изначально содержит в себе данные об упражнениях (название, техника выполнения). Эти данные обычный пользователь изменять не может.

* Спортсмен

Спортсмен может зарегистрироваться или войти в уже существующую учётную запись. После авторизации ему будет доступен профиль, где он сможет изменить уже имеющиеся или внести после регистрации данные о себе (Имя, пол, возраст, рост, вес, ссылка на соцсети).

Спортсмен может внести свои результаты выполнения упражнений. Для этого нужно выбрать упражнение из списка, затем ввести свой результат в зависимости от выбранного упражнения (например, вес и количество повторений или только количество повторений). Далее проводится подсчёт одноповторного максимума и результат сохраняется в профиле спортсмена.

Также доступна история выполнения всех упражнений спортсмена.

Спортсмен может просматривать и оценивать, выложенные тренерами тренировочные программы или советы.

Также спортсмен может открывать и просматривать профили тренеров и других спортсменов.

* Тренер

Тренер может зарегистрироваться или войти в уже существующую учётную запись. После авторизации ему будет доступен профиль, где он сможет изменить уже имеющиеся или внести после регистрации данные о себе (Имя, возраст, пол, рост, вес). Ещё тренер может внести информацию об индивидуальных достижениях (например, мастер спорта, заслуженный тренер России)

Тренер, как и спортсмен может внести свои результаты выполнения упражнений. Для этого нужно выбрать упражнение из списка, затем ввести свой результат в зависимости от выбранного упражнения (например, вес и количество повторений или только количество повторений). Далее проводится подсчёт одноповторного максимума и результат сохраняется в профиле.

Тренер может выкладывать свои тренировочные программы или разные советы (например, по выполнению упражнений или советы по питанию).

Также тренер может открывать и просматривать профили тренеров и других спортсменов.

Тренер может просматривать и оценивать, выложенные другими тренерами тренировочные программы или советы.

**6 Диаграмма вариантов использования (use case)**

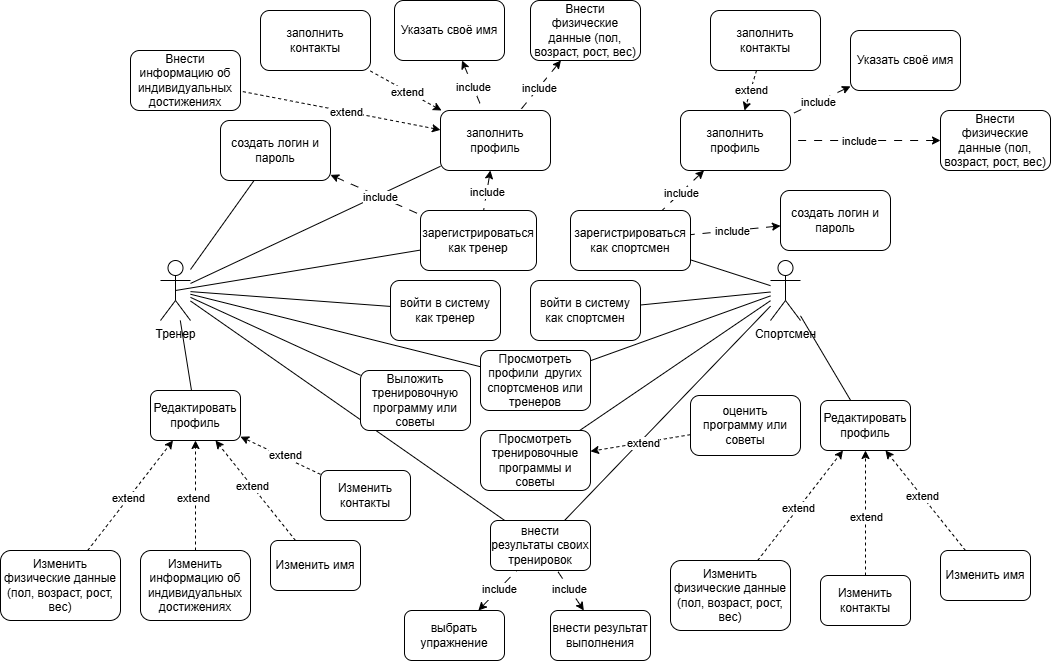


Рисунок 1 – диаграмма вариантов использования

**7 Предполагаемые технологии и платформа реализации**

* СУБД: PostgreSQL;
* • ОС: Windows;
* язык программирования: Python;
* фреймворк: Flask;
* тип приложения: веб-приложение.

**8 Срок представления курсовой работы**

27.12.2024

**Проектирование базы данных**

На этом этапе используется модель сущность-связь (рис.1). Схема «сущность-связь» (также ERD или ER-диаграмма) — это разновидность блок-схемы, где показано, как разные «сущности» (люди, объекты, концепции и так далее) связаны между собой внутри системы. ER-диаграммы чаще всего применяются для проектирования и отладки реляционных баз данных в сфере образования, исследования и разработки программного обеспечения и информационных систем для бизнеса. ER-диаграммы (или ER-модели) полагаются на стандартный набор символов, включая прямоугольники, ромбы, овалы и соединительные линии, для отображения сущностей, их атрибутов и связей. Связи между сущностями проводятся в том случае, если экземпляры сущностей связаны друг с другом семантически. Экземпляры связей также несут информацию и устанавливают ассоциации между конкретными экземплярами сущностей. Сущность может быть связана сама с собой, в случае если между собой ассоциированы ее отдельные экземпляры. Связи различаются по кратности (мощности): «один-к-одному», «один-ко-многим», «многие-ко-многим». Тип связи определяется по отдельности на обоих ее концах. Чтобы определить тип связи на одном из концов, необходимо зафиксировать одиночный экземпляр одной из связанных сущностей и оценить, сколько экземпляров второй сущности с ним может быть логически связано: только один или несколько. Затем аналогичные действия производятся для другого конца связи.

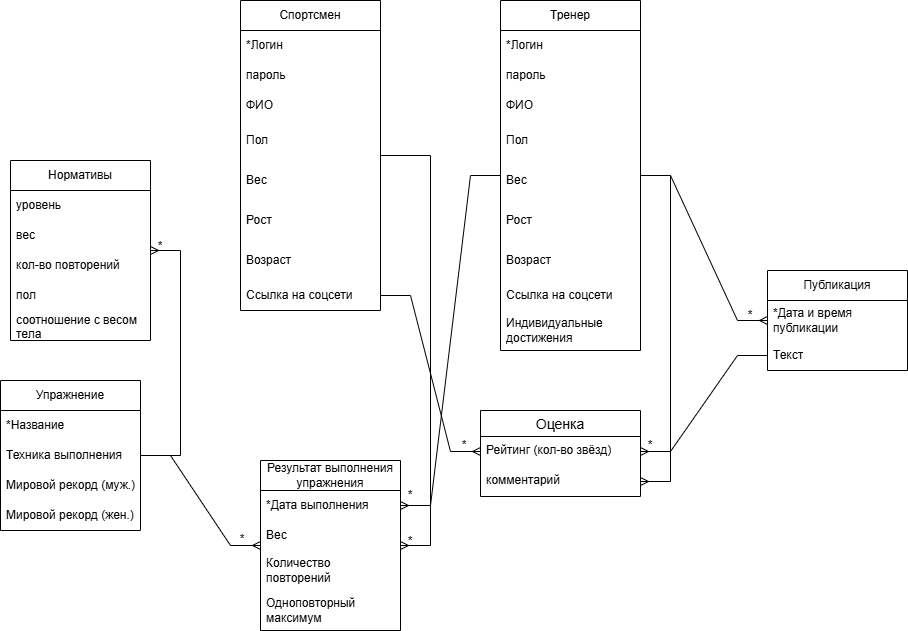


Рисунок 1 – схема сущность-связь для разработанной базы данных

После составления ER-модели необходимо, руководствуясь нижеприведенным алгоритмом, перейти к реляционной модели (рис. 2) той же самой предметной области. В дальнейшем в СУБД используется именно она. Реляционная модель является менее абстрактной и уточняет некоторые детали, такие как домены для атрибутов, специфические для СУБД названия отношений и атрибутов, ограничения целостности и др. Алгоритм перехода от ER-модели к реляционной модели:

1. Каждой сущности ставится в соответствие отношение. Атрибуты сущности становятся атрибутами отношения. Если атрибут был ключевым, он становится частью потенциального ключа отношения.

2. Для каждой связи «один-ко-многим» ключевые атрибуты со стороны «один» копируются на сторону «многие». Если связь была ключевой (и только в этом случае), скопированные атрибуты становятся частью ключа.

3. Для каждой связи «многие-ко-многим» создается промежуточное отношение, в которое копируются ключевые атрибуты связанных сущностей. Все они образуют составной ключ. Полученное отношение (т.н. «отношение пересечения») связывается с исходными связями «многие-к-одному».

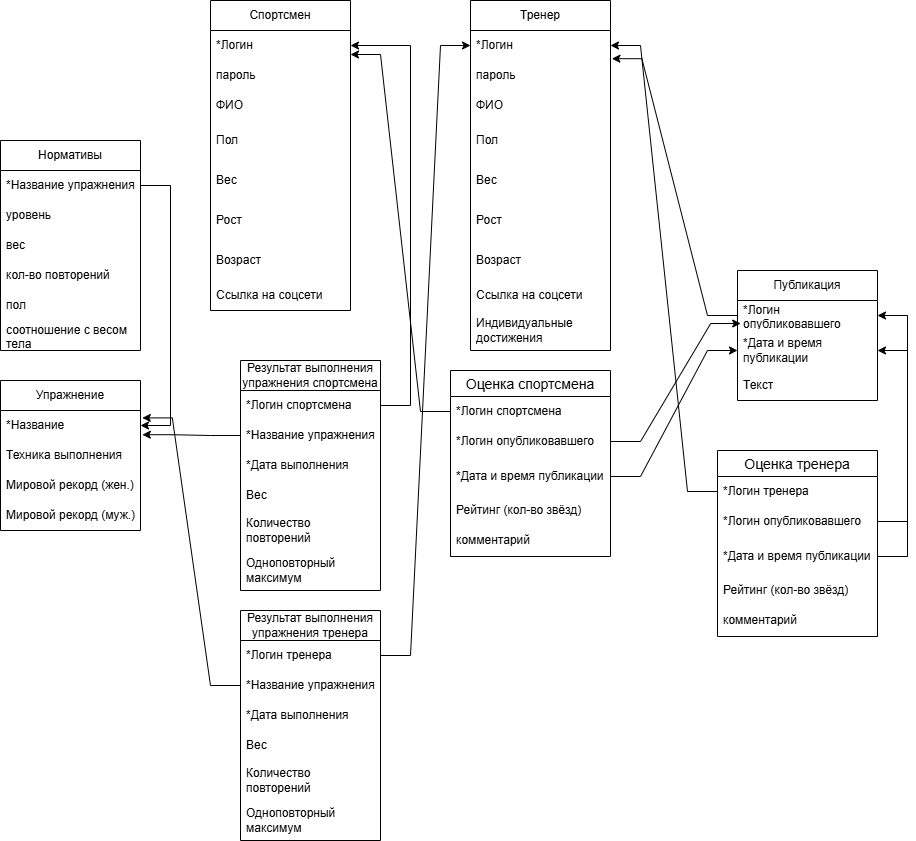


Рисунок 2 – реляционная схема базы данных

После создания реляционной схемы базы данных можно приступать к созданию базы данных.

Скрипт генерации БД (файл generate.sql):

DROP TABLE IF EXISTS COACH\_RATING;

DROP TABLE IF EXISTS ATHLETE\_RATING;

DROP TABLE IF EXISTS PUBLICATION;

DROP TABLE IF EXISTS COACH\_RESULT;

DROP TABLE IF EXISTS ATHLETE\_RESULT;

DROP TABLE IF EXISTS COACH;

DROP TABLE IF EXISTS ATHLETE;

DROP TABLE IF EXISTS STANDARDS;

DROP TABLE IF EXISTS EXERCISE;

CREATE TABLE IF NOT EXISTS EXERCISE

(

    NAZV VARCHAR(100) PRIMARY KEY,

    TECHNIQE VARCHAR,

    WORLD\_RECORD\_MEN VARCHAR,

    WORLD\_RECORD\_WOMEN VARCHAR

);

CREATE TABLE IF NOT EXISTS STANDARDS

(

    EXERCISE\_NAME VARCHAR(100),

    STRENGTH\_LEVEL VARCHAR,

    WEIGHT\_KG FLOAT,

    REPS\_COUNT INT,

    GENDER VARCHAR(10),

    BODYWEIGHT\_RATIO FLOAT,

    FOREIGN KEY(EXERCISE\_NAME) REFERENCES EXERCISE(NAZV) ON UPDATE CASCADE

    PRIMARY KEY (EXERCISE\_NAME, STRENGTH\_LEVEL,GENDER)

);

CREATE TABLE IF NOT EXISTS ATHLETE

(

    USERNAME VARCHAR(150) PRIMARY KEY,

    PASSWD VARCHAR (200),

    FIO VARCHAR (200),

    GENDER VARCHAR(10),

    BODYWEIGHT FLOAT,

    HEIGHT FLOAT,

    AGE INT,

    SOCIAL\_MEDIA VARCHAR

    ROL INT DEFAULT 1

);

CREATE TABLE IF NOT EXISTS COACH

(

    USERNAME VARCHAR(150) PRIMARY KEY,

    PASSWD VARCHAR (200),

    FIO VARCHAR (200),

    GENDER VARCHAR(10),

    BODYWEIGHT FLOAT,

    HEIGHT FLOAT,

    AGE INT,

    SOCIAL\_MEDIA VARCHAR,

    ACHIEVEMENTS VARCHAR

    ROL INT DEFAULT 2

);

CREATE TABLE IF NOT EXISTS ATHLETE\_RESULT

(

    USERNAME VARCHAR(150),

    EXERCISE\_NAME VARCHAR(100),

    EXECUTION\_DATE TIMESTAMP,

    WEIGHT\_KG FLOAT,

    REPS\_COUNT INT,

    ONE\_RPM FLOAT,

    FOREIGN KEY (USERNAME) REFERENCES ATHLETE(USERNAME) ON UPDATE CASCADE,

    FOREIGN KEY (EXERCISE\_NAME) REFERENCES EXERCISE(NAZV) ON UPDATE CASCADE,

    PRIMARY KEY (USERNAME, EXERCISE\_NAME, EXECUTION\_DATE)

);

CREATE TABLE IF NOT EXISTS COACH\_RESULT

(

    USERNAME VARCHAR(150),

    EXERCISE\_NAME VARCHAR(100),

    EXECUTION\_DATE TIMESTAMP,

    WEIGHT\_KG FLOAT,

    REPS\_COUNT INT,

    ONE\_RPM FLOAT,

    FOREIGN KEY (USERNAME) REFERENCES COACH(USERNAME) ON UPDATE CASCADE,

    FOREIGN KEY (EXERCISE\_NAME) REFERENCES EXERCISE(NAZV) ON UPDATE CASCADE,

    PRIMARY KEY (USERNAME, EXERCISE\_NAME, EXECUTION\_DATE)

);

CREATE TABLE IF NOT EXISTS PUBLICATION

(

    AUTHOR\_USERNAME VARCHAR(150),

    PUBLICATION\_DATETIME TIMESTAMP,

    TEXT VARCHAR,

    FOREIGN KEY (AUTHOR\_USERNAME) REFERENCES COACH(USERNAME) ON UPDATE CASCADE,

    PRIMARY KEY (AUTHOR\_USERNAME, PUBLICATION\_DATETIME)

);

CREATE TABLE IF NOT EXISTS ATHLETE\_RATING

(

    VIEWER\_USERNAME VARCHAR(150),

    AUTHOR\_USERNAME VARCHAR(150),

    PUBLICATION\_DATETIME TIMESTAMP,

    STARS INT,

    COMMENT VARCHAR,

    FOREIGN KEY (VIEWER\_USERNAME) REFERENCES ATHLETE(USERNAME) ON UPDATE CASCADE,

    FOREIGN KEY (AUTHOR\_USERNAME, PUBLICATION\_DATETIME) REFERENCES PUBLICATION(AUTHOR\_USERNAME, PUBLICATION\_DATETIME) ON UPDATE CASCADE,

    PRIMARY KEY(VIEWER\_USERNAME, AUTHOR\_USERNAME,PUBLICATION\_DATETIME)

);

CREATE TABLE IF NOT EXISTS COACH\_RATING

(

    VIEWER\_USERNAME VARCHAR(150),

    AUTHOR\_USERNAME VARCHAR(150),

    PUBLICATION\_DATETIME TIMESTAMP,

    STARS INT,

    COMMENT VARCHAR,

    FOREIGN KEY (VIEWER\_USERNAME) REFERENCES COACH(USERNAME) ON UPDATE CASCADE,

    FOREIGN KEY (AUTHOR\_USERNAME, PUBLICATION\_DATETIME) REFERENCES PUBLICATION(AUTHOR\_USERNAME, PUBLICATION\_DATETIME) ON UPDATE CASCADE,

    PRIMARY KEY(VIEWER\_USERNAME, AUTHOR\_USERNAME,PUBLICATION\_DATETIME)

);

**Описание выбранных технологий реализации, развертывания и методов взаимодействия с базой данных**

После создания базы данных можно приступать к разработке веб-приложения. Для разработки был выбран язык программирования Python, фреймворк Flask, СУБД PostgreSQL.

Python — это язык программирования, который широко используется в интернет-приложениях, разработке программного обеспечения, науке о данных и машинном обучении (ML). Разработчики используют Python, потому что он эффективен, прост в изучении и работает на разных платформах.

Flask — это легковесный веб-фреймворк для языка Python, который предоставляет минимальный набор инструментов для создания веб-приложений. На нём можно сделать многостраничный сайт с множетсвом плагинов и сервисов.

PostgreSQL — это объектно-реляционная система управления базами данных (ORDBMS), наиболее развитая из открытых СУБД в мире. Имеет открытый исходный код и является альтернативой коммерческим базам данных.

Взаимодействие с базами данных может происходить с использованием различных методов, в зависимости от типа базы данных и требуемых операций. Например, через SQL:

SELECT — извлечение данных.

INSERT — вставка новых данных.

UPDATE — обновление существующих данных.

DELETE — удаление данных.

CREATE — создание новых таблиц, индексов, представлений и т.д.

ALTER — изменение структуры базы данных.

DROP — удаление объектов базы данных.

SQL-запросы обычно отправляются через клиентские программы или API:

GET — получение информации о данных или списка объектов;

DELETE — удаление данных;

POST — добавление или замена данных;

PUT — регулярное обновление данных.

Развертывание приложения проводилось на сервисе Render, работающем по модели PaaS(Platform-as-a-Service, платформа как услуга). В данной модели поставщик облачных услуг предоставляет готовую инфраструктуру с настроенным программным обеспечением, а пользователь только деплоит код без необходимости администрирования. Такой подход менее гибок, но хорошо подходит для небольших приложений. Render предлагает бесплатный хостинг веб-приложений на Python и других платформах. Бесплатный экземпляр веб-cервиса обладает следующими характеристиками: 0.1 CPU, 512 MB RAM. Для начала работы с сервисом Render была создана учетная запись. Регистрация проводилась с помощью учетной записи GitHub для подключения репозитория с проектом для быстрого и легкого развертывания. При регистрации было указано, что сервис будет использован для личных целей и развёртывания нового проекта. После завершения процесса регистрации был начат процесс развёртывания. Так как при регистрации была использована учётная запись GitHub, то можно сразу выбрать оттуда свой репозиторий, относящийся к разрабатываемому приложению, после подключения репозитория необходимо было указать: Имя проекта, язык программирования (в данном случае Python3), ветку репозитория, из которой будет считываться код проекта, (по умолчанию используется master или main), корневой каталог (по умолчанию используется каталог, выбранного репозитория), команды, отвечающие за запуск и сборку проекта (gunicorn -w 1 -b 0.0.0.0:$PORT myproject:app и python3 -m pip install -r requirements-render.txt соответсвенно).

Развертывание базы данных также проводилось в облаке, на сервисе Tembo (платформа, специализирующаяся на предоставлении управляемых экземпляров PostgreSQL с возможностью подключения большого количества расширений.)  Для начала необходимо зарегистрироваться, после регистрации нужно выбрать стек (специализацию экземпляра базы данных).  Для целей небольшого веб-приложения подойдет стек Standard, предназначенный для рабочих нагрузок общего типа. После выбора стека необходимо заполнить форму с деталями создаваемого экземпляра: задать ему имя, выбрать версию PostgreSQL (существенной роли не играет), выбрать облачного провайдера (необходимо выбрать AWS, так как для него доступно большее количество регионов). В качестве региона, как и в случае с Aiven, необходимо выбрать тот, который географически максимально близок к размещению приложения, в данном случае это Europe (Frankfurt). После этого необходимо выбрать один из типов создаваемого экземпляра. Тип влияет на аппаратные характеристики оборудования и определяет стоимость, интерес представляет бесплатный тип (Hobby). После выбора бесплатного типа дальнейшие настройки экземпляра становятся недоступны, и можно завершать заполнение формы. Процесс развертывания занимает несколько минут, и после изменения статуса экземпляра с "Provisioning" на "Good" можно начинать с ним работу.

**Листинги модулей**

Листинг модуля \_\_init\_\_.py:

from flask import Flask

from app.config import Config

from flask\_bootstrap import Bootstrap5

from flask\_login import LoginManager

app = Flask(\_\_name\_\_)

app.config.from\_object(Config)

bootstrap = Bootstrap5(app)

login\_manager = LoginManager()

login\_manager.init\_app(app)

login\_manager.login\_view = 'login'

from app import routes

Листинг модуля config.py:

import os

from dotenv import load\_dotenv

load\_dotenv()

class Config(object):

    SECRET\_KEY = os.environ.get('SECRET\_KEY')

    DB\_SERVER = os.environ.get('DB\_SERVER')

    DB\_USER = os.environ.get('DB\_USER')

    DB\_PASSWORD = os.environ.get('DB\_PASSWORD')

    DB\_NAME = os.environ.get('DB\_NAME')

Листинг модуля forms.py:

from flask\_wtf import FlaskForm

from wtforms import BooleanField, StringField, DateField, IntegerField, FloatField, PasswordField, SubmitField, SelectField, validators

#форма регистрации спортсмена

class ATH\_RegistrationForm(FlaskForm):

    username = StringField('Имя пользователя', [validators.Length(min=4, max=25)])

    password = PasswordField('Пароль', [validators.InputRequired(),

                                        validators.Length(min=6, max=100),

                                        validators.EqualTo('confirm', message='Пароли должны совпадать')])

    confirm  = PasswordField('Повторите пароль')

    age = IntegerField('Возраст', [validators.InputRequired()])

    FIO = StringField('ФИО', [validators.Length(min=4, max=100),validators.Optional()])

    height = FloatField('Рост в см', [validators.InputRequired()])

    weight = FloatField('Масса в кг', [validators.InputRequired()])

    media = StringField('Ссылка на соцсети', [validators.Length(min=4, max=100),validators.Optional()])

    gender = SelectField('Пол (муж или жен)', choices=[('муж'),('жен')])

    submit = SubmitField('Зарегистрироваться')

#форма регистрации тренера

class CO\_RegistrationForm(FlaskForm):

    username = StringField('Имя пользователя', [validators.Length(min=4, max=25)])

    password = PasswordField('Пароль', [validators.InputRequired(),

                                        validators.Length(min=6, max=100),

                                        validators.EqualTo('confirm', message='Пароли должны совпадать')])

    confirm  = PasswordField('Повторите пароль')

    age = IntegerField('Возраст', [validators.InputRequired()])

    FIO = StringField('ФИО', [validators.Length(min=4, max=100),validators.Optional()])

    height = FloatField('Рост в см', [validators.InputRequired()])

    weight = FloatField('Масса в кг', [validators.InputRequired()])

    media = StringField('Ссылка на соцсети', [validators.Length(min=4, max=100),validators.Optional()])

    achv = StringField('Индивидуальные достижения', [validators.Length(min=4, max=500),validators.Optional()])

    gender = SelectField('Пол (муж или жен)', choices=[('муж'),('жен')])

    submit = SubmitField('Зарегистрироваться')

#форма редактирования профиля спортсмена

class EditProfileForm(FlaskForm):

    age = IntegerField('Возраст')

    FIO = StringField('ФИО')

    gender = SelectField('Пол (муж или жен)', choices=[('муж'),('жен')])

    height = FloatField('Рост в см')

    weight = FloatField('Масса в кг')

    media = StringField('Ссылка на соцсети')

    submit = SubmitField('Сохранить')

#форма редактирования профиля тренера

class CO\_EditProfileForm(FlaskForm):

    age = IntegerField('Возраст')

    FIO = StringField('ФИО')

    gender = SelectField('Пол (муж или жен)', choices=[('муж'),('жен')])

    height = FloatField('Рост в см')

    weight = FloatField('Масса в кг')

    media = StringField('Ссылка на соцсети')

    achv = StringField('Индивидуальные достижения')

    submit = SubmitField('Сохранить')

#форма авторизации

class LoginForm(FlaskForm):

    username = StringField('Логин', [validators.InputRequired()])

    password = PasswordField('Пароль', [validators.InputRequired()])

    remember\_me = BooleanField('Запомнить меня')

    submit = SubmitField('Войти')

#форма добавления результата

class AddResult(FlaskForm):

    excercise = SelectField('Упражнение', choices=[('жим лёжа'),('приседания со штангой'),

                                                   ('становая тяга'),('подъём гантелей на бицепс'),

                                                   ('жим гантелей над головой'),('подтягивания'),

                                                   ('подтягивания с дополнительным отягощением'),('тяга вертикального блока'),

                                                   ('тяга горизонтального блока'),('ягодичный мостик')],validate\_choice=True)

    weight = FloatField('Вес в кг', [validators.Optional()])

    reps = IntegerField('Количество повторений')

    submit = SubmitField('Добавить')

#форма добавления публикации

class AddPost(FlaskForm):

    nazv = StringField('Введите название публикации')

    Text = StringField('Введите текст вашей публикации')

    submit = SubmitField('Запостить!')

#форма добавления отзыва

class RateForm (FlaskForm):

    rating = SelectField('Выберите оценку (от 1 до 5 звёзд)', choices=[(1),(2),(3),(4),(5)],validate\_choice=True)

    comment = StringField('Ваши комментарии',[validators.Optional()])

    submit = SubmitField('Оценить')

Листинг модуля routes.py:

from flask import render\_template

from app import app

import psycopg

@app.route('/') # главная страница

def index():

    with psycopg.connect(host=app.config['DB\_SERVER'], # подключение к БД

                         user=app.config['DB\_USER'],

                         password=app.config['DB\_PASSWORD'],

                         dbname=app.config['DB\_NAME']) as con:

        cur = con.cursor() # вывод спортсменов и тренеров

        coaches = cur.execute('SELECT username, fio, gender, bodyweight, height, age, social\_media, achievements FROM coach').fetchall()

        athletes = cur.execute('SELECT username, fio, gender, bodyweight, height, age, social\_media FROM athlete').fetchall()

    return render\_template('index.html',coaches = coaches, athletes = athletes)

@app.route('/testdb') # проверка подключения к БД

def test\_connection():

    con = None

    message = ""

    try:

        con = psycopg.connect(host=app.config['DB\_SERVER'], # подключение к БД

                              user=app.config['DB\_USER'],

                              password=app.config['DB\_PASSWORD'],

                              dbname=app.config['DB\_NAME'])

    except Exception as e: # вывод результата проверки

        message = f"Ошибка подключения: {e}"

    else:

        message = "Подключение успешно"

    finally:

        if con:

            con.close()

        return message

@app.route('/excercises') # вывод всех упражнений и техники выполнения

def get\_excercises():

    with psycopg.connect(host=app.config['DB\_SERVER'],# подключение к БД

                         user=app.config['DB\_USER'],

                         password=app.config['DB\_PASSWORD'],

                         dbname=app.config['DB\_NAME']) as con:

        cur = con.cursor()

        excercises = cur.execute(f'SELECT nazv, techniqe FROM exercise').fetchall()

    return render\_template('excercises.html',excercises=excercises)

from flask import request

from app.forms import ATH\_RegistrationForm

from flask import render\_template

from werkzeug.security import generate\_password\_hash

from app import app

from flask import redirect, render\_template, flash

from app.forms import ATH\_RegistrationForm

@app.route('/AT\_register', methods=['GET', 'POST']) # регистрация спортсмена

def register():

    if current\_user.is\_authenticated:

        return redirect(url\_for('index'))

    reg\_form = ATH\_RegistrationForm()

    if reg\_form.validate\_on\_submit():

        login = reg\_form.username.data # получение данных из формы регистрации

        password\_hash = generate\_password\_hash(reg\_form.password.data)

        age = reg\_form.age.data

        fio = reg\_form.FIO.data

        height = reg\_form.height.data

        weight = reg\_form.weight.data

        media = reg\_form.media.data

        gender = reg\_form.gender.data

        with psycopg.connect(host=app.config['DB\_SERVER'],# подключение к БД

                         user=app.config['DB\_USER'],

                         password=app.config['DB\_PASSWORD'],

                         dbname=app.config['DB\_NAME']) as con:

            cur = con.cursor()

            cur.execute('INSERT INTO athlete (username, passwd,fio,gender,bodyweight,height,age,social\_media)' # добавление нового пользователя в базу данных

                        'VALUES (%s, %s, %s, %s, %s, %s, %s, %s)',

                        (login,password\_hash,fio,gender,weight,height,age,media))

        # создание записи в таблице используя password\_hash

        flash(f'Регистрация {reg\_form.username.data} успешна', 'success')

        return redirect(url\_for('login'))

    return render\_template('registration.html', title='Регистрация спортсмена', form=reg\_form) # вывод страницы регистрации

from app.forms import CO\_RegistrationForm

@app.route('/CO\_register', methods=['GET', 'POST'])# регистрация тренера

def co\_register():

    if current\_user.is\_authenticated:

        return redirect(url\_for('index'))

    reg\_form = CO\_RegistrationForm()

    if reg\_form.validate\_on\_submit():

        login = reg\_form.username.data # получение данных из формы регистрации

        password\_hash = generate\_password\_hash(reg\_form.password.data)

        age = reg\_form.age.data

        fio = reg\_form.FIO.data

        height = reg\_form.height.data

        weight = reg\_form.weight.data

        media = reg\_form.media.data

        achv = reg\_form.achv.data

        gender = reg\_form.gender.data

        with psycopg.connect(host=app.config['DB\_SERVER'],

                         user=app.config['DB\_USER'],

                         password=app.config['DB\_PASSWORD'],

                         dbname=app.config['DB\_NAME']) as con:

            cur = con.cursor()

            cur.execute('INSERT INTO coach (username, passwd,fio,gender,bodyweight,height,age,social\_media,achievements)' # добавление нового пользователя в базу данных

                        'VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s)',

                        (login,password\_hash,fio,gender,weight,height,age,media,achv))

        # создать запись в таблице используя password\_hash

        flash(f'Регистрация {reg\_form.username.data} успешна', 'success')

        return redirect(url\_for('CO\_login'))

    return render\_template('CO\_registration.html', title='Регистрация тренера', form=reg\_form) # вывод страницы регистрации

from werkzeug.security import check\_password\_hash

from app.forms import LoginForm

from app.user import User

from flask import render\_template, redirect, flash, url\_for

from flask\_login import login\_user, current\_user

from urllib.parse import urlsplit

@app.route('/login', methods=['GET', 'POST']) # авторизация спортсмена

def login():

    login\_form = LoginForm()

    if login\_form.validate\_on\_submit():

        with psycopg.connect(host=app.config['DB\_SERVER'], # подключение к бд

                         user=app.config['DB\_USER'],

                         password=app.config['DB\_PASSWORD'],

                         dbname=app.config['DB\_NAME']) as con:

            cur = con.cursor()

            res = cur.execute('SELECT username, passwd '

                              'FROM "athlete" '

                              'WHERE username = %s', (login\_form.username.data,)).fetchone() # выбор пользователя с соответстсвующими данными

        if res is None or not check\_password\_hash(res[1], login\_form.password.data): # проверка пароля

            flash('Попытка входа неудачна', 'danger')

            return redirect(url\_for('login'))

        role = 1

        id = res[0]

        username, password = res

        user = User(id, username, password, role)

        login\_user(user, remember=login\_form.remember\_me.data)

        next = request.args.get('next')

        if not next or urlsplit (next).netloc != '':

            next = url\_for('index')

        flash(f'Вы успешно вошли в систему, {current\_user.username}', 'danger')

        return redirect(next)

    return render\_template('login.html', title='Авторизация спортсмена', form=login\_form) # вывод страницы авторизации

@app.route('/CO\_login', methods=['GET', 'POST']) # авторизация тренера

def CO\_login():

    login\_form = LoginForm()

    if login\_form.validate\_on\_submit():

        with psycopg.connect(host=app.config['DB\_SERVER'], # подключение к бд

                         user=app.config['DB\_USER'],

                         password=app.config['DB\_PASSWORD'],

                         dbname=app.config['DB\_NAME']) as con:

            cur = con.cursor()

            res = cur.execute('SELECT username, passwd '

                              'FROM "coach" '

                              'WHERE username = %s', (login\_form.username.data,)).fetchone() # выбор пользователя с соответстсвующими данными

        if res is None or not check\_password\_hash(res[1], login\_form.password.data): # проверка пароля

            flash('Попытка входа неудачна', 'danger')

            return redirect(url\_for('CO\_login'))

        role = 2

        id ='\_CO\_'+res[0] # указатель, что это id тренера

        username, password = res

        user = User(id, username, password, role)

        login\_user(user, remember=login\_form.remember\_me.data)

        next = request.args.get('next')

        if not next or urlsplit (next).netloc != '':

            next = url\_for('index')

        flash(f'Вы успешно вошли в систему, {current\_user.username}', 'danger')

        return redirect(next)

    return render\_template('login.html', title='Авторизация тренера', form=login\_form) # вывод страницы авторизации

from app import app

from flask\_login import logout\_user

from flask import redirect, url\_for

@app.route('/logout') # выход из учетной записи

def logout():

    logout\_user()

    return redirect(url\_for('index'))

@app.route ('/records')

def show\_records():

    with psycopg.connect(host=app.config['DB\_SERVER'], # подключение к бд

                         user=app.config['DB\_USER'],

                         password=app.config['DB\_PASSWORD'],

                         dbname=app.config['DB\_NAME']) as con:

        cur = con.cursor()

        records = cur.execute(f'SELECT nazv, world\_record\_men, world\_record\_women FROM exercise').fetchall() # выбор мировых рекордов из таблицы с упражнениями

    return render\_template('records.html',recs=records) # вывод страницы с рекордами

from flask import render\_template, abort

from flask\_login import current\_user, login\_required

@app.route('/profile/<username>') # профиль спортсмена

@login\_required

def profile(username):

    with psycopg.connect(host=app.config['DB\_SERVER'], # подключение к бд

                         user=app.config['DB\_USER'],

                         password=app.config['DB\_PASSWORD'],

                         dbname=app.config['DB\_NAME']) as con:

        cur = con.cursor()

        res = cur.execute('SELECT fio, gender, bodyweight, height, age, social\_media '

                    'FROM "athlete" '

                    'WHERE username = %s', (username,)).fetchone()  # вывод данных спортсмена

        exc = cur.execute('SELECT username, exercise\_name, weight\_kg, reps\_count, one\_rpm, execution\_date '

                    'FROM "athlete\_result" '

                    'WHERE username = %s ORDER BY exercise\_name ASC, execution\_date DESC', (username,)).fetchall() # вывод результатов спортсмена

    fio,gender,weight,height,age,socialmedia = res

    results = exc

    return render\_template('profile.html',fio = fio,gender = gender,weight = weight,height = height,

                           age = age ,socialmedia = socialmedia, results = results, profile\_username = username) # вывод страницы профиля

def format\_datetime\_for\_url(post\_datetime):

    # Преобразуем строку даты и времени в формат, безопасный для URL

    # Например, заменим все знаки на нижние подчеркивания

    b = str(post\_datetime)

    b = b.replace(" ","\_")

    b = b.replace(":","\_")

    return b

def datetime\_from\_url(b):

    l1 = list(b)

    l1[10] = ' '

    l1[13] = ':'

    l1[16] = ':'

    b = ''.join(l1)

    return b

@app.route('/CO\_profile/<username>') # профиль тренера

@login\_required

def CO\_profile(username):

    with psycopg.connect(host=app.config['DB\_SERVER'], # подключение к бд

                         user=app.config['DB\_USER'],

                         password=app.config['DB\_PASSWORD'],

                         dbname=app.config['DB\_NAME']) as con:

        cur = con.cursor()

        res = cur.execute('SELECT fio, gender, bodyweight, height, age, social\_media, achievements '

                    'FROM "coach" '

                    'WHERE username = %s', (username,)).fetchone() # вывод данных тренера

        exc = cur.execute('SELECT username, exercise\_name, weight\_kg, reps\_count, one\_rpm, execution\_date '

                    'FROM "coach\_result" '

                    'WHERE username = %s', (username,)).fetchall() # вывод результатов тренера

        prg = cur.execute('SELECT nazv, publication\_datetime, text '

                          'FROM "publication" '

                          'WHERE author\_username = %s', (username,)).fetchall() # вывод публикаций

        ath\_ratings = cur.execute ('SELECT \* '

                    'FROM "athlete\_rating" '

                    'WHERE author\_username = %s', (username,)).fetchall() # вывод отзывов спортсменов

        coach\_ratings = cur.execute ('SELECT \* '

                    'FROM "coach\_rating" '

                    'WHERE author\_username = %s', (username,)).fetchall() # вывод отзывов тренеров

    fio,gender,weight,height,age,socialmedia,achv = res

    results = exc

    posts = prg

    return render\_template('CO\_profile.html',fio = fio,gender = gender,weight = weight,height = height,

                           age = age ,socialmedia = socialmedia, achv = achv, results = results, posts = posts,

                           profile\_username = username, format\_datetime\_for\_url = format\_datetime\_for\_url, ath\_ratings = ath\_ratings, coach\_ratings = coach\_ratings) # страница профиля тренера

from app.forms import EditProfileForm,CO\_EditProfileForm

from flask import redirect, flash, url\_for

@app.route('/edit', methods=['GET', 'POST']) # редактирование профиля

@login\_required

def edit\_user():

    with psycopg.connect(host=app.config['DB\_SERVER'], # подключение к бд

                         user=app.config['DB\_USER'],

                         password=app.config['DB\_PASSWORD'],

                         dbname=app.config['DB\_NAME']) as con:

        cur = con.cursor()

        if current\_user.role == 1: # если это спортсмен, то достаём данные профиля из таблицы с спортсменами

            res = cur.execute('SELECT fio, gender, bodyweight, height, age, social\_media '

                        'FROM "athlete" '

                        'WHERE username = %s', (current\_user.username,)).fetchone()

            fio,gender,weight,height,age,socialmedia = res

            form = EditProfileForm(FIO=fio, weight = weight, height = height, age = age, media = socialmedia, gender = gender)

        elif current\_user.role == 2: # если это тренер, то достаём данные профиля из таблицы с тренерами

            res = cur.execute('SELECT fio, gender, bodyweight, height, age, social\_media, achievements '

                        'FROM "coach" '

                        'WHERE username = %s', (current\_user.username,)).fetchone()

            fio, gender, weight,height,age,socialmedia,achievements = res

            form = CO\_EditProfileForm(FIO=fio, weight = weight, height = height, age = age, media = socialmedia, gender = gender,achv = achievements)

    if form.validate\_on\_submit():

        fio = form.FIO.data # получение данных из формы

        gender = form.gender.data

        weight = form.weight.data

        height = form.height.data

        age = form.age.data

        socialmedia = form.media.data

        if current\_user.role == 2: # если тренер, то можно ещё изменить данные об индивидуальных достижениях

            achievements = form.achv.data

        print(fio)

        with psycopg.connect(host=app.config['DB\_SERVER'], # подключение к бд

                         user=app.config['DB\_USER'],

                         password=app.config['DB\_PASSWORD'],

                         dbname=app.config['DB\_NAME']) as con:

            cur = con.cursor()

            if current\_user.role == 1:# если спортсмен, то редактируем таблицу спорстмена

                cur.execute('UPDATE athlete SET fio =%s, gender = %s, bodyweight =%s, height = %s, age = %s, social\_media = %s WHERE username =%s',

                            (fio,gender,weight,height,age,socialmedia,current\_user.username))

            elif current\_user.role == 2:# если тренер, то редактируем таблицу тренера

                cur.execute('UPDATE coach SET fio =%s, gender= %s, bodyweight =%s, height = %s, age = %s, social\_media = %s, achievements = %s WHERE username =%s',

                            (fio,gender,weight,height,age,socialmedia,achievements,current\_user.username))

        print(fio)

        # сохранить новые данные пользователя в базе данных

        flash('Изменения сохранены')

        if current\_user.role == 1:# если спортсмен, то переход в профиль спортсмена

            return redirect(url\_for('profile', username = current\_user.username ))

        elif current\_user.role == 2: # если тренер, то переход в профиль тренера

            return redirect(url\_for('CO\_profile', username = current\_user.username))

    else:

        print('fail')

    return render\_template('edit.html', title='Редактирование пользователя', form=form)

from app.forms import AddResult

import datetime

@app.route('/add\_result', methods=['GET', 'POST']) # добавить результат упражения

@login\_required

def add\_result():

    add\_form = AddResult()

    if add\_form.validate\_on\_submit():

        now = datetime.datetime.now()

        username = current\_user.username # получение данных из формы

        execution\_date = now.strftime("%Y-%m-%d %H:%M:%S") # преобразование даты в нужный формат

        excercise\_name = add\_form.excercise.data

        weight = add\_form.weight.data

        reps = add\_form.reps.data

        print(execution\_date)

        if weight == None: # рассчёт одноповторного максимума

            ONE\_REP\_MAX = 0

        elif (reps == 1):

            ONE\_REP\_MAX = weight

        else:

            ONE\_REP\_MAX = weight\*(1+0.0333\*reps)

        with psycopg.connect(host=app.config['DB\_SERVER'], # подключение к бд

                            user=app.config['DB\_USER'],

                            password=app.config['DB\_PASSWORD'],

                            dbname=app.config['DB\_NAME']) as con:

            cur = con.cursor()

            if current\_user.role == 1: # если спортсмен, то добавляем в таблицу для спортсмена

                cur.execute('INSERT INTO athlete\_result '

                            'VALUES (%s, %s, %s, %s, %s, %s)',

                            (username,excercise\_name,execution\_date,weight,reps, ONE\_REP\_MAX))

                return redirect(url\_for('profile',username = username))

            elif current\_user.role == 2:# если тренер, то добавляем в таблицу для тренера

                cur.execute('INSERT INTO coach\_result '

                            'VALUES (%s, %s, %s, %s, %s, %s)',

                            (username,excercise\_name,execution\_date,weight,reps, ONE\_REP\_MAX))

                return redirect(url\_for('CO\_profile',username = username))

    return render\_template('add\_result.html',form = add\_form)

from app.forms import AddPost

@app.route('/upload', methods=['GET', 'POST']) # загрузить публикацию

@login\_required

def add\_post():

    add\_form = AddPost()

    if current\_user.role != 2: # проверка роли, если не тренер, то добавить публикацию нельзя

        abort(403)

    if add\_form.validate\_on\_submit():

        username = current\_user.username # получение данных из формы

        nazv = add\_form.nazv.data

        text = add\_form.Text.data

        now = datetime.datetime.now()

        publication\_date = now.strftime("%Y-%m-%d %H:%M:%S") # преобразование даты в нужный формат

        with psycopg.connect(host=app.config['DB\_SERVER'], # подключение к бд

                            user=app.config['DB\_USER'],

                            password=app.config['DB\_PASSWORD'],

                            dbname=app.config['DB\_NAME']) as con:

            cur = con.cursor()

            cur.execute('INSERT INTO publication '

                        'VALUES (%s, %s,%s,%s)',

                        (username,publication\_date,text,nazv)) # добавление публикации в бд

        return redirect(url\_for('CO\_profile',username = username))

    return render\_template('upload.html',form = add\_form)

@app.route('/standards', methods=['GET', 'POST']) # силовые стандарты

def get\_standards():

    with psycopg.connect(host=app.config['DB\_SERVER'], # подключение к бд

                            user=app.config['DB\_USER'],

                            password=app.config['DB\_PASSWORD'],

                            dbname=app.config['DB\_NAME']) as con:

            cur = con.cursor()

            stnd = cur.execute('SELECT \* '

                    'FROM "standards" ORDER BY exercise\_name ASC, gender DESC, bodyweight\_ratio DESC').fetchall() #

    return render\_template('standards.html',standards = stnd)

from app.forms import RateForm

@app.route('/CO\_profile/<username>/add\_rating/<dat>', methods=['GET', 'POST']) # добавление отзыва на публикацию в профиле тренера

@login\_required

def add\_rating(username,dat):

    print(f"username: {username}, dat: {dat}")

    form = RateForm()

    b = datetime\_from\_url(dat)

    if form.validate\_on\_submit():

        b = datetime\_from\_url(dat)

        print ('yes')

        rating = form.rating.data

        comment = form.comment.data

        with psycopg.connect(host=app.config['DB\_SERVER'], # подключение к бд

                                user=app.config['DB\_USER'],

                                password=app.config['DB\_PASSWORD'],

                                dbname=app.config['DB\_NAME']) as con:

                cur = con.cursor()

                rev = current\_user.username

                author = username

                publication\_date = cur.execute('SELECT publication\_datetime '

                            'FROM "publication" '

                            'WHERE author\_username = %s AND publication\_datetime = %s ' , (username, b)).fetchall() #

                if current\_user.role == 1: # если спортсмен, то добавляем отзыв в отзывы спортсменов

                    cur.execute('INSERT INTO athlete\_rating '

                                'VALUES (%s,%s,%s,%s,%s)',

                                (rev,author,publication\_date,rating,comment))

                elif current\_user.role == 2: # если тренер, то добавляем отзыв в отзывы тренеров

                    cur.execute('INSERT INTO coach\_rating '

                                'VALUES (%s,%s,%s,%s,%s)',

                                (rev,author,publication\_date,rating,comment))

        return redirect(url\_for('CO\_profile',username = username))

    return render\_template('add\_review.html',public\_date = b , form = form)

Листинг модуля user.py:

import psycopg

from app import app

from app import login\_manager

from flask\_login import UserMixin

class User(UserMixin):

    def \_\_init\_\_(self, id, username, password, role):

        self.id = id

        self.username = username

        self.password = password

        self.role = role

@login\_manager.user\_loader

def load\_user(id): # авторизация пользователя

    with psycopg.connect(host=app.config['DB\_SERVER'], # подключение к бд

                         user=app.config['DB\_USER'],

                         password=app.config['DB\_PASSWORD'],

                         dbname=app.config['DB\_NAME']) as con:

        cur = con.cursor()

        if id[0:4] == '\_CO\_': # ID тренера начинается с \_CO\_

            username, password, role = cur.execute('SELECT username, passwd, rol '

                                            'FROM "coach" '

                                            'WHERE username = %s', (id[4:],)).fetchone()

        else:

            username, password, role = cur.execute('SELECT username, passwd, rol '

                                            'FROM "athlete" '

                                            'WHERE username = %s', (id,)).fetchone()

    return User(id, username, password, role)

Листинг модуля myproject.py:

from app import app

Листинг модуля CO\_profile.html:

{% extends "base.html" %}

{% block content %}

<title>Профиль спортсмена</title>

</head>

<body>

<div class="container mt-5">

    <h2 class="text-center">Профиль тренера</h2>

    <!-- Отображение данных профиля -->

    <div class="card text-white bg-dark mb-3">

        <div class="card-header">

            <h4>Основные данные</h4>

        </div>

        <div class="card-body">

            <p><strong>Имя:</strong> {{ fio }}</p>

            <p><strong>Пол:</strong> {{ gender }}</p>

            <p><strong>Возраст:</strong> {{ age }}</p>

            <p><strong>Рост:</strong> {{ height }} см</p>

            <p><strong>Вес:</strong> {{ weight }} кг</p>

            <p><strong>Ссылка на соцсети:</strong> {{ socialmedia }} </p>

            <p><strong>Индивидуальные достижения:</strong> {{ achv }}</p>

        </div>

    </div>

    {% if current\_user.username == profile\_username and current\_user.role == 2 %}

    <!-- Кнопочка для добавления результата -->

    <div class="mt-4">

        <a href="/add\_result" class="btn btn-success">Добавить результат</a>

    </div>

    <!-- Кнопочка для редактирования профиля -->

    <div class="mt-4">

        <a href="/edit" class="btn btn-info">Редактировать профиль</a>

    </div>

    <!-- Кнопочка для добавления публикации -->

    <div class="mt-4">

        <a href="/upload" class="btn btn-primary">Добавить публикацию</a>

    </div>

    {% endif %}

</div>

<!-- Результаты спортсмена -->

<div class="card text-white bg-dark mt-4">

    <div class="card-header">

        <h4>результаты</h4>

    </div>

    <div class="card-body">

        <table class="table table-bordered table-dark">

            <thead>

                <tr>

                    <th>Упражнение</th>

                    <th>Вес (кг)</th>

                    <th>Повторения</th>

                    <th>Одноповторный максимум (1ПМ)</th>

                    <th>Дата выполнения</th>

                </tr>

            </thead>

            <tbody>

                {% if results %}

                    {% for result in results %}

                    <tr>

                        <td>{{ result[1] }}</td>

                        <td>{{ result[2] }}</td>

                        <td>{{ result[3] }}</td>

                        <td>{{ result[4] }}</td>

                        <td>{{ result[5] }}</td>

                    </tr>

                    {% endfor %}

                {% else %}

                    <tr><td colspan="5">Нет результатов для отображения.</td></tr>

                {% endif %}

            </tbody>

        </table>

    </div>

</div>

<!-- Публикации -->

<div class="card text-white bg-dark mt-4">

    <div class="card-header">

        <h4>Публикации</h4>

    </div>

    <div class="card-body">

        <table class="table table-bordered table-dark">

            <thead>

                <tr>

                    <th>Название</th>

                    <th>Дата и время публикации</th>

                    <th>Текст</th>

                    <th>Оставить отзыв</th>

                </tr>

            </thead>

            <tbody>

                {% if posts %}

                    {% for post in posts %}

                    <tr>

                        <td>{{ post[0] }}</td>

                        <td>{{ post[1] }}</td>

                        <td>{{ post[2] }}</td>

                        <td>

                            <!-- Кнопочка добавления оценки публикации -->

                            <a href="/CO\_profile/{{profile\_username}}/add\_rating/{{ format\_datetime\_for\_url(post[1]) }}" target="\_blank" class="btn btn-outline-primary">{{ 'Оставить отзыв' }}</a></td>

                            <!-- Заменяем пробелы и двоеточия на подчеркивание в дате для корректной работы ссылки -->

                        </td>

                    </tr>

                    {% endfor %}

                {% else %}

                    <tr><td colspan="4">Нет результатов для отображения.</td></tr>

                {% endif %}

            </tbody>

        </table>

    </div>

</div>

<!-- Отзывы спортсменов -->

<div class="card text-white bg-dark mt-4">

    <div class="card-header">

        <h4>Отзывы спортсменов</h4>

    </div>

    <div class="card-body">

        <table class="table table-bordered table-dark">

            <thead>

                <tr>

                    <th>Название публикации</th>

                    <th>Логин оценившего</th>

                    <th>Логин автора</th>

                    <th>Дата публикации</th>

                    <th>Количество звёзд (от 1 до 5)</th>

                    <th>Комментарии оценившего</th>

                </tr>

            </thead>

            <tbody>

                {% if ath\_ratings %}

                    {% for ath\_rating in ath\_ratings %}

                        <tr>

                            {% for post in posts %}

                                {% if ath\_rating[2] == post [1] %}

                                    <td>{{ post[0] }}</td>

                                {% endif %}

                            {%endfor%}

                            <td>{{ ath\_rating[0] }} <a href="/profile/{{ath\_rating[0]}}" target="\_blank" class="btn btn-outline-info">{{ 'перейти в профиль' }}</a></td>

                            <td>{{ ath\_rating[1] }} </td>

                            <td>{{ ath\_rating[2] }}</td>

                            <td>{{ ath\_rating[3] }}</td>

                            <td>{{ ath\_rating[4] }}</td>

                        </tr>

                    {% endfor %}

                {% else %}

                    <tr><td colspan="6">Нет результатов для отображения.</td></tr>

                {% endif %}

            </tbody>

        </table>

    </div>

</div>

<!-- Отзывы тренеров -->

<div class="card text-white bg-dark mt-4">

    <div class="card-header">

        <h4>Отзывы тренеров</h4>

    </div>

    <div class="card-body">

        <table class="table table-bordered table-dark">

            <thead>

                <tr>

                    <th>Название публикации</th>

                    <th>Логин оценившего</th>

                    <th>Логин автора</th>

                    <th>Дата публикации</th>

                    <th>Количество звёзд (от 1 до 5)</th>

                    <th>Комментарии оценившего</th>

                </tr>

            </thead>

            <tbody>

                {% if coach\_ratings %}

                    {% for coach\_rating in coach\_ratings %}

                    <tr>

                        {% for post in posts %}

                            {% if coach\_rating[2] == post [1] %}

                                <td>{{ post[0] }}</td>

                            {% endif %}

                        {%endfor%}

                        <td>{{ coach\_rating[0] }}<a href="/CO\_profile/{{coach\_rating[0]}}" target="\_blank" class="btn btn-outline-info">{{ 'перейти в профиль' }}</a> </td>

                        <td>{{ coach\_rating[1] }}</td>

                        <td>{{ coach\_rating[2] }}</td>

                        <td>{{ coach\_rating[3] }}</td>

                        <td>{{ coach\_rating[4] }}</td>

                    </tr>

                    {% endfor %}

                {% else %}

                    <tr><td colspan="6">Нет результатов для отображения.</td></tr>

                {% endif %}

            </tbody>

        </table>

    </div>

</div>

</body>

{% endblock %}

Листинг модуля CO\_registration.html:

{% extends "base.html" %}

{% block content %}

    <h2>{{ title }}</h2>

    <form method="POST" action="">

        {{ form.hidden\_tag() }}

        {% if form.username.errors %}

            {% for error in form.username.errors %}

                <div class="invalid-feedback">{{ error }}</div>

            {% endfor %}

        {% endif %}

        <div>{{ form.username.label }} {{ form.username(class="form-control bg-dark text-light border-secondary", style="width: 300px;") }}</div>

        {% if form.password.errors %}

            {% for error in form.password.errors %}

                <div class="invalid-feedback">{{ error }}</div>

            {% endfor %}

        {% endif %}

        <div>{{ form.password.label }} {{ form.password(class="form-control bg-dark text-light border-secondary", style="width: 300px;") }}</div>

        {% if form.confirm.errors %}

            {% for error in form.confirm.errors %}

                <div class="invalid-feedback">{{ error }}</div>

            {% endfor %}

        {% endif %}

        <div>{{ form.confirm.label }} {{ form.confirm(class="form-control bg-dark text-light border-secondary", style="width: 300px;") }}</div>

        {% if form.age.errors %}

            {% for error in form.age.errors %}

                <div class="invalid-feedback">{{ error }}</div>

            {% endfor %}

        {% endif %}

        <div>{{ form.age.label }} {{ form.age(class="form-control bg-dark text-light border-secondary", style="width: 300px;") }}</div>

        {% if form.FIO.errors %}

            {% for error in form.FIO.errors %}

                <div class="invalid-feedback">{{ error }}</div>

            {% endfor %}

        {% endif %}

        <div>{{ form.FIO.label }} {{ form.FIO(class="form-control bg-dark text-light border-secondary", style="width: 300px;") }}</div>

        {% if form.height.errors %}

            {% for error in form.height.errors %}

                <div class="invalid-feedback">{{ error }}</div>

            {% endfor %}

        {% endif %}

        <div>{{ form.height.label }} {{ form.height(class="form-control bg-dark text-light border-secondary", style="width: 300px;") }}</div>

        {% if form.weight.errors %}

            {% for error in form.height.errors %}

                <div class="invalid-feedback">{{ error }}</div>

            {% endfor %}

        {% endif %}

        <div>{{ form.weight.label }} {{ form.weight(class="form-control bg-dark text-light border-secondary", style="width: 300px;") }}</div>

        {% if form.media.errors %}

            {% for error in form.media.errors %}

                <div class="invalid-feedback">{{ error }}</div>

            {% endfor %}

        {% endif %}

        <div>{{ form.media.label }} {{ form.media(class="form-control bg-dark text-light border-secondary", style="width: 300px;") }}</div>

        {% if form.achv.errors %}

            {% for error in form.media.errors %}

                <div class="invalid-feedback">{{ error }}</div>

            {% endfor %}

        {% endif %}

        <div>{{ form.achv.label }} {{ form.achv(class="form-control bg-dark text-light border-secondary", style="width: 300px;") }}</div>

        {% if form.gender.errors %}

            {% for error in form.gender.errors %}

                <div class="invalid-feedback">{{ error }}</div>

            {% endfor %}

        {% endif %}

        <div>{{ form.gender.label }} {{ form.gender(class="form-control bg-dark text-light border-secondary", style="width: 100px;") }}</div>

        {{ form.submit(class="form-control bg-dark text-light border-secondary", style="width: 300px;") }}

    </form>

{% endblock %}

Листинг модуля edit.html:

{% extends "base.html" %}

{% block content %}

    <h2>Редактировать профиль</h2>

    <form method="POST" action="">

        {{ form.hidden\_tag() }}

        {% if form.age.errors %}

            {% for error in form.age.errors %}

                <div class="invalid-feedback">{{ error }}</div>

            {% endfor %}

        {% endif %}

        <div>{{ form.age.label }} {{ form.age(class="form-control bg-dark text-light border-secondary", style="width: 300px;") }}</div>

        {% if form.FIO.errors %}

            {% for error in form.FIO.errors %}

                <div class="invalid-feedback">{{ error }}</div>

            {% endfor %}

        {% endif %}

        <div>{{ form.FIO.label }} {{ form.FIO(class="form-control bg-dark text-light border-secondary", style="width: 300px;") }}</div>

        {% if form.height.errors %}

            {% for error in form.height.errors %}

                <div class="invalid-feedback">{{ error }}</div>

            {% endfor %}

        {% endif %}

        <div>{{ form.height.label }} {{ form.height(class="form-control bg-dark text-light border-secondary", style="width: 300px;") }}</div>

        {% if form.weight.errors %}

            {% for error in form.weight.errors %}

                <div class="invalid-feedback">{{ error }}</div>

            {% endfor %}

        {% endif %}

        <div>{{ form.weight.label }} {{ form.weight(class="form-control bg-dark text-light border-secondary", style="width: 300px;") }}</div>

        {% if form.media.errors %}

            {% for error in form.media.errors %}

                <div class="invalid-feedback">{{ error }}</div>

            {% endfor %}

        {% endif %}

        <div>{{ form.media.label }} {{ form.media(class="form-control bg-dark text-light border-secondary", style="width: 300px;") }}</div>

        {% if form.gender.errors %}

            {% for error in form.gender.errors %}

                <div class="invalid-feedback">{{ error }}</div>

            {% endfor %}

        {% endif %}

        <div>{{ form.gender.label }} {{ form.gender(class="form-control bg-dark text-light border-secondary", style="width: 100px;") }}</div>

        {% if current\_user.role == 2 %}

            {% if form.achv.errors %}

                {% for error in form.achv.errors %}

                    <div class="invalid-feedback">{{ error }}</div>

                {% endfor %}

            {% endif %}

            <div>{{ form.achv.label }} {{ form.achv(class="form-control bg-dark text-light border-secondary", style="width: 300px;") }}</div>

        {% endif %}

        {{ form.submit(class="form-control bg-dark text-light border-secondary", style="width: 300px;") }}

    </form>

{% endblock %}

Листинг модуля excercises.html:

{% extends "base.html" %}

{% block content %}

    <h2>Редактировать профиль</h2>

    <form method="POST" action="">

        {{ form.hidden\_tag() }}

        {% if form.age.errors %}

            {% for error in form.age.errors %}

                <div class="invalid-feedback">{{ error }}</div>

            {% endfor %}

        {% endif %}

        <div>{{ form.age.label }} {{ form.age(class="form-control bg-dark text-light border-secondary", style="width: 300px;") }}</div>

        {% if form.FIO.errors %}

            {% for error in form.FIO.errors %}

                <div class="invalid-feedback">{{ error }}</div>

            {% endfor %}

        {% endif %}

        <div>{{ form.FIO.label }} {{ form.FIO(class="form-control bg-dark text-light border-secondary", style="width: 300px;") }}</div>

        {% if form.height.errors %}

            {% for error in form.height.errors %}

                <div class="invalid-feedback">{{ error }}</div>

            {% endfor %}

        {% endif %}

        <div>{{ form.height.label }} {{ form.height(class="form-control bg-dark text-light border-secondary", style="width: 300px;") }}</div>

        {% if form.weight.errors %}

            {% for error in form.weight.errors %}

                <div class="invalid-feedback">{{ error }}</div>

            {% endfor %}

        {% endif %}

        <div>{{ form.weight.label }} {{ form.weight(class="form-control bg-dark text-light border-secondary", style="width: 300px;") }}</div>

        {% if form.media.errors %}

            {% for error in form.media.errors %}

                <div class="invalid-feedback">{{ error }}</div>

            {% endfor %}

        {% endif %}

        <div>{{ form.media.label }} {{ form.media(class="form-control bg-dark text-light border-secondary", style="width: 300px;") }}</div>

        {% if form.gender.errors %}

            {% for error in form.gender.errors %}

                <div class="invalid-feedback">{{ error }}</div>

            {% endfor %}

        {% endif %}

        <div>{{ form.gender.label }} {{ form.gender(class="form-control bg-dark text-light border-secondary", style="width: 100px;") }}</div>

        {% if current\_user.role == 2 %}

            {% if form.achv.errors %}

                {% for error in form.achv.errors %}

                    <div class="invalid-feedback">{{ error }}</div>

                {% endfor %}

            {% endif %}

            <div>{{ form.achv.label }} {{ form.achv(class="form-control bg-dark text-light border-secondary", style="width: 300px;") }}</div>

        {% endif %}

        {{ form.submit(class="form-control bg-dark text-light border-secondary", style="width: 300px;") }}

    </form>

{% endblock %}

Листинг модуля index.html:

{%extends 'base.html'%}

{% block content %}

<h1>Главная</h1>

<h1>Наши тренеры:</h1>

<table class="table table-bordered table-dark">

    <thead>

        <tr>

            <th>Имя пользователя</th>

            <th>ФИО</th>

            <th>Пол</th>

            <th>Вес</th>

            <th>Рост</th>

            <th>Возраст</th>

            <th>Ссылка на соцсети</th>

            <th>достижения</th>

        </tr>

    </thead>

    <tbody>

    {% for coach in coaches %}

        <tr>

            <td>{{ coach[0] }} <a href="CO\_profile/{{coach[0]}}" target="\_blank" class="btn btn-outline-info">{{ 'перейти в профиль' }}</a></td>

            <td>{{ coach[1] }}</td>

            <td>{{ coach[2] }}</td>

            <td>{{ coach[3] }}</td>

            <td>{{ coach[4] }}</td>

            <td>{{ coach[5] }}</td>

            <td>{{ coach[6] }}</td>

            <td>{{ coach[7] }}</td>

        </tr>

    {% endfor %}

    </tbody>

</table>

<h1>Наши спортсмены:</h1>

<table class="table table-bordered table-dark">

    <thead>

        <tr>

            <th>Имя пользователя</th>

            <th>ФИО</th>

            <th>Пол</th>

            <th>Вес</th>

            <th>Рост</th>

            <th>Возраст</th>

            <th>Ссылка на соцсети</th>

        </tr>

    </thead>

    <tbody>

    {% for athlete in athletes %}

        <tr>

            <td>{{ athlete[0] }} <a href="profile/{{athlete[0]}}" target="\_blank" class="btn btn-outline-info">{{ 'перейти в профиль' }}</a></td>

            <td>{{ athlete[1] }}</td>

            <td>{{ athlete[2] }}</td>

            <td>{{ athlete[3] }}</td>

            <td>{{ athlete[4] }}</td>

            <td>{{ athlete[5] }}</td>

            <td>{{ athlete[6] }}</td>

        </tr>

    {% endfor %}

    </tbody>

</table>

{% endblock %}

Листинг модуля login.html:

{% extends "base.html" %}

{% block content %}

    <h2>{{ title }}</h2>

    <form method="POST" action="">

        {{ form.hidden\_tag() }}

        {% if form.username.errors %}

            {% for error in form.username.errors %}

                <div class="invalid-feedback">{{ error }}</div>

            {% endfor %}

        {% endif %}

        <div>{{ form.username.label }} {{ form.username(class="form-control bg-dark text-light border-secondary", style="width: 300px;") }}</div>

        {% if form.password.errors %}

            {% for error in form.password.errors %}

                <div class="invalid-feedback">{{ error }}</div>

            {% endfor %}

        {% endif %}

        <div>{{ form.password.label }} {{ form.password(class="form-control bg-dark text-light border-secondary", style="width: 300px;") }}</div>

        {% if form.remember\_me.errors %}

            {% for error in form.remember\_me.errors %}

                <div class="invalid-feedback">{{ error }}</div>

            {% endfor %}

        {% endif %}

        <div>{{ form.remember\_me.label }} {{ form.remember\_me() }}</div>

        {{ form.submit(class="form-control bg-dark text-light border-secondary", style="width: 300px;") }}

    </form>

{% endblock %}

Листинг модуля profile:

{% extends "base.html" %}

{% block content %}

<title>Профиль спортсмена</title>

</head>

<body>

<div class="container mt-5">

    <h2 class="text-center">Профиль спортсмена</h2>

    <!-- Отображение данных профиля -->

    <div class="card text-white bg-dark mb-3">

        <div class="card-header">

            <h4>Основные данные</h4>

        </div>

        <div class="card-body">

            <p><strong>Имя:</strong> {{ fio }}</p>

            <p><strong>Пол:</strong> {{ gender }}</p>

            <p><strong>Возраст:</strong> {{ age }}</p>

            <p><strong>Рост:</strong> {{ height }} см</p>

            <p><strong>Вес:</strong> {{ weight }} кг</p>

            <p><strong>Ссылка на соцсети:</strong>{{ socialmedia }}</p>

        </div>

    </div>

    {% if current\_user.username == profile\_username %}

    <!-- Кнопочка для добавления результата -->

    <div class="mt-4">

        <a href="/add\_result" class="btn btn-success">Добавить результат</a>

    </div>

    <!-- Кнопочка для редактирования профиля -->

    <div class="mt-4">

        <a href="/edit" class="btn btn-info">Редактировать профиль</a>

    </div>

    {% endif %}

</div>

<!-- Результаты спортсмена -->

<div class="card text-white bg-dark mt-4">

    <div class="card-header">

        <h4>результаты</h4>

    </div>

    <div class="card-body">

        <table class="table table-bordered table-dark">

            <thead>

                <tr>

                    <th>Упражнение</th>

                    <th>Вес (кг)</th>

                    <th>Повторения</th>

                    <th>Одноповторный максимум (1ПМ)</th>

                    <th>Дата выполнения</th>

                </tr>

            </thead>

            <tbody>

                {% if results %}

                    {% for result in results %}

                    <tr>

                        <td>{{ result[1] }}</td>

                        <td>{{ result[2] }}</td>

                        <td>{{ result[3] }}</td>

                        <td>{{ result[4] }}</td>

                        <td>{{ result[5] }}</td>

                    </tr>

                    {% endfor %}

                {% else %}

                    <tr><td colspan="5">Нет результатов для отображения.</td></tr>

                {% endif %}

            </tbody>

        </table>

    </div>

</div>

</body>

{% endblock %}

Листинг модуля records.html:

{% extends "base.html" %}

{% block content %}

<h1>Рекорды</h1>

<table class="table table-bordered table-dark">

    <thead>

        <tr>

            <th>упражнение</th>

            <th>Рекорд(муж):</th>

            <th>Рекорд(жен):</th>

        </tr>

    </thead>

    <tbody>

    {% for record in recs %}

        <tr>

            <td>{{ record[0]}}</td>

            <td>{{ record[1] }}</td>

            <td>{{ record[2] }}</td>

        </tr>

    {% endfor %}

    </tbody>

</table>

{% endblock %}

Листинг модуля registration.html:

{% extends "base.html" %}

{% block content %}

    <h2>{{ title }}</h2>

    <form method="POST" action="">

        {{ form.hidden\_tag() }}

        {% if form.username.errors %}

            {% for error in form.username.errors %}

                <div class="invalid-feedback">{{ error }}</div>

            {% endfor %}

        {% endif %}

        <div>{{ form.username.label }} {{ form.username(class="form-control bg-dark text-light border-secondary", style="width: 300px;") }}</div>

        {% if form.password.errors %}

            {% for error in form.password.errors %}

                <div class="invalid-feedback">{{ error }}</div>

            {% endfor %}

        {% endif %}

        <div>{{ form.password.label }} {{ form.password(class="form-control bg-dark text-light border-secondary", style="width: 300px;") }}</div>

        {% if form.confirm.errors %}

            {% for error in form.confirm.errors %}

                <div class="invalid-feedback">{{ error }}</div>

            {% endfor %}

        {% endif %}

        <div>{{ form.confirm.label }} {{ form.confirm(class="form-control bg-dark text-light border-secondary", style="width: 300px;") }}</div>

        {% if form.age.errors %}

            {% for error in form.age.errors %}

                <div class="invalid-feedback">{{ error }}</div>

            {% endfor %}

        {% endif %}

        <div>{{ form.age.label }} {{ form.age(class="form-control bg-dark text-light border-secondary", style="width: 300px;") }}</div>

        {% if form.FIO.errors %}

            {% for error in form.FIO.errors %}

                <div class="invalid-feedback">{{ error }}</div>

            {% endfor %}

        {% endif %}

        <div>{{ form.FIO.label }} {{ form.FIO(class="form-control bg-dark text-light border-secondary", style="width: 300px;") }}</div>

        {% if form.height.errors %}

            {% for error in form.height.errors %}

                <div class="invalid-feedback">{{ error }}</div>

            {% endfor %}

        {% endif %}

        <div>{{ form.height.label }} {{ form.height(class="form-control bg-dark text-light border-secondary", style="width: 300px;") }}</div>

        {% if form.weight.errors %}

            {% for error in form.height.errors %}

                <div class="invalid-feedback">{{ error }}</div>

            {% endfor %}

        {% endif %}

        <div>{{ form.weight.label }} {{ form.weight(class="form-control bg-dark text-light border-secondary", style="width: 300px;") }}</div>

        {% if form.media.errors %}

            {% for error in form.media.errors %}

                <div class="invalid-feedback">{{ error }}</div>

            {% endfor %}

        {% endif %}

        <div>{{ form.media.label }} {{ form.media(class="form-control bg-dark text-light border-secondary", style="width: 300px;") }}</div>

        {% if form.gender.errors %}

            {% for error in form.gender.errors %}

                <div class="invalid-feedback">{{ error }}</div>

            {% endfor %}

        {% endif %}

        <div>{{ form.gender.label }} {{ form.gender(class="form-control bg-dark text-light border-secondary", style="width: 100px;") }}</div>

        {{ form.submit(class="form-control bg-dark text-light border-secondary", style="width: 300px;") }}

    </form>

{% endblock %}

Листинг модуля standards:

{% extends "base.html" %}

{% block content %}

<h1>Силовые стандарты</h1>

<table class="table table-bordered table-dark">

    <thead>

        <tr>

            <th>упражнение</th>

            <th>Уровень:</th>

            <th>Вес</th>

            <th>Количество повторений</th>

            <th>Пол</th>

            <th>Соотношение с весом тела</th>

        </tr>

    </thead>

    <tbody>

    {% for standard in standards %}

        <tr>

            <td>{{ standard[0] }}</td>

            <td>{{ standard[1] }}</td>

            <td>{{ standard[2] }}</td>

            <td>{{ standard[3] }}</td>

            <td>{{ standard[4] }}</td>

            <td>{{ standard[5] }}</td>

        </tr>

    {% endfor %}

    </tbody>

</table>

{% endblock %}

Листинг модуля upload.html:

{% extends "base.html" %}

{% block content %}

<h2>Сделать публикацию</h2>

    <form method="POST" action="">

        {{ form.hidden\_tag() }}

        <div class="mb-3">

            {% if form.nazv.errors %}

                {% for error in form.nazv.errors %}

                    <div class="invalid-feedback">{{ error }}</div>

                {% endfor %}

            {% endif %}

            {{ form.nazv.label }} {{ form.nazv(class="form-control bg-dark text-light border-secondary",style="width: 400px;") }}

        </div>

        <div class="mb-3">

        {% if form.Text.errors %}

            {% for error in form.Text.errors %}

                <div class="invalid-feedback">{{ error }}</div>

            {% endfor %}

        {% endif %}

        {{ form.Text(class="form-control form-control-lg bg-dark text-light border-secondary", style="width: 100%; height: 200px; resize: vertical;") }}

        </div>

        {{ form.submit(class="form-control bg-dark text-light border-secondary", style="width: 100px;") }}

    </form>

{% endblock %}

Листинг модуля add\_result.html:

{% extends "base.html" %}

{% block content %}

    <h2>Добавить результат</h2>

    <form method="POST" action="">

        {{ form.hidden\_tag() }}

        {% if form.excercise.errors %}

            {% for error in form.username.errors %}

                <div class="invalid-feedback">{{ error }}</div>

            {% endfor %}

        {% endif %}

        <div>{{ form.excercise.label }} {{ form.excercise(class="form-control bg-dark text-light border-secondary", style="width: 300px;") }}</div>

        {% if form.weight.errors %}

            {% for error in form.weight.errors %}

                <div class="invalid-feedback">{{ error }}</div>

            {% endfor %}

        {% endif %}

        <div>{{ form.weight.label }} {{ form.weight(class="form-control bg-dark text-light border-secondary", style="width: 300px;") }}</div>

        {% if form.reps.errors %}

            {% for error in form.reps.errors %}

                <div class="invalid-feedback">{{ error }}</div>

            {% endfor %}

        {% endif %}

        <div>{{ form.reps.label }} {{ form.reps(class="form-control bg-dark text-light border-secondary", style="width: 300px;") }}</div>

        {{ form.submit(class="form-control bg-dark text-light border-secondary", style="width: 300px;") }}

    </form>

{% endblock %}

Листинг модуля add\_review:

{% extends "base.html" %}

{% block content %}

<h2>Оставить отзыв для публикации "{{ public\_date }}"</h2>

<form method="POST" action="">

    {{ form.hidden\_tag() }}

    <div class="mb-3">

        {% if form.comment.errors %}

            {% for error in form.comment.errors %}

                <div class="invalid-feedback d-block">{{ error }}</div>

            {% endfor %}

        {% endif %}

        {{ form.comment(class="form-control form-control-lg bg-dark text-light border-secondary") }}

    </div>

    </div>

     <div class="mb-3">

        {% if form.rating.errors %}

            {% for error in form.rating.errors %}

                <div class="invalid-feedback d-block">{{ error }}</div>

            {% endfor %}

        {% endif %}

        {{ form.rating(class="form-control bg-dark text-light border-secondary", style="width: 300px;") }}

    </div>

    <div class="mb-3">

        {{ form.submit(class="btn btn-primary") }}

    </div>

</form>

{% endblock %}

Листинг модуля base.html:

{% from 'bootstrap5/nav.html' import render\_nav\_item %}

<!DOCTYPE html>

<html lang="ru">

    <head>

        <meta charset="utf-8">

        <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

        {% block styles %}

            {{ bootstrap.load\_css() }}

            <style>

                body

                {

                    background-color: #191d20;

                    color: white;

                }

                .navbar {

                    background-color: #343a40;

                }

                .navbar-nav .nav-link {

                    color: #f8f9fa;

                }

                .navbar-nav .nav-link:hover {

                    color: #007bff;

                }

            </style>

        {% endblock %}

        {% if title %}

            <title>{{ title }}</title>

        {% else %}

            <title>Добро пожаловать</title>

        {% endif %}

    </head>

    <body>

        <nav class="navbar navbar-expand-lg navbar-light" style="background-color: #4ba7ea;">

            <div class="container-fluid">

                <div class="navbar-nav">

                    {{ render\_nav\_item('index', 'Главная') }}

                    {{ render\_nav\_item('test\_connection', 'Проверить подключение') }}

                    {% if current\_user.is\_authenticated %}

                        {% if current\_user.role == 1 %}

                            <span>Добро пожаловать, {{current\_user.username}}, вы авторизовались как спортсмен </span>

                            {{ render\_nav\_item('profile', 'Профиль', username=current\_user.username) }}

                        {% elif current\_user.role == 2 %}

                            <span>Добро пожаловать, {{current\_user.username}}, вы авторизовались как тренер </span>

                            {{ render\_nav\_item('CO\_profile', 'Профиль', username=current\_user.username) }}

                        {% endif %}

                        {{ render\_nav\_item('get\_excercises', 'Все упражнения') }}

                        {{ render\_nav\_item('show\_records', 'Мировые рекорды') }}

                        {{ render\_nav\_item('get\_standards', 'Силовые стандарты') }}

                        {{ render\_nav\_item('logout', 'Выйти') }}

                    {% else %}

                        {{ render\_nav\_item('register', 'Зарегистрироваться как спортсмен') }}

                        {{ render\_nav\_item('co\_register', 'Зарегистрироваться как тренер') }}

                        {{ render\_nav\_item('login', 'войти как спортсмен') }}

                        {{ render\_nav\_item('CO\_login', 'войти как тренер') }}

                    {% endif %}

                </div>

            </div>

        </nav>

        {% block content %}{% endblock %}

        {% block scripts %}

            {{ bootstrap.load\_js() }}

        {% endblock %}

    </body>

</html>

**Тестирование функционала**

Зарегистрироваться как спортсмен можно по кнопке «Зарегистрироваться как спортсмен»

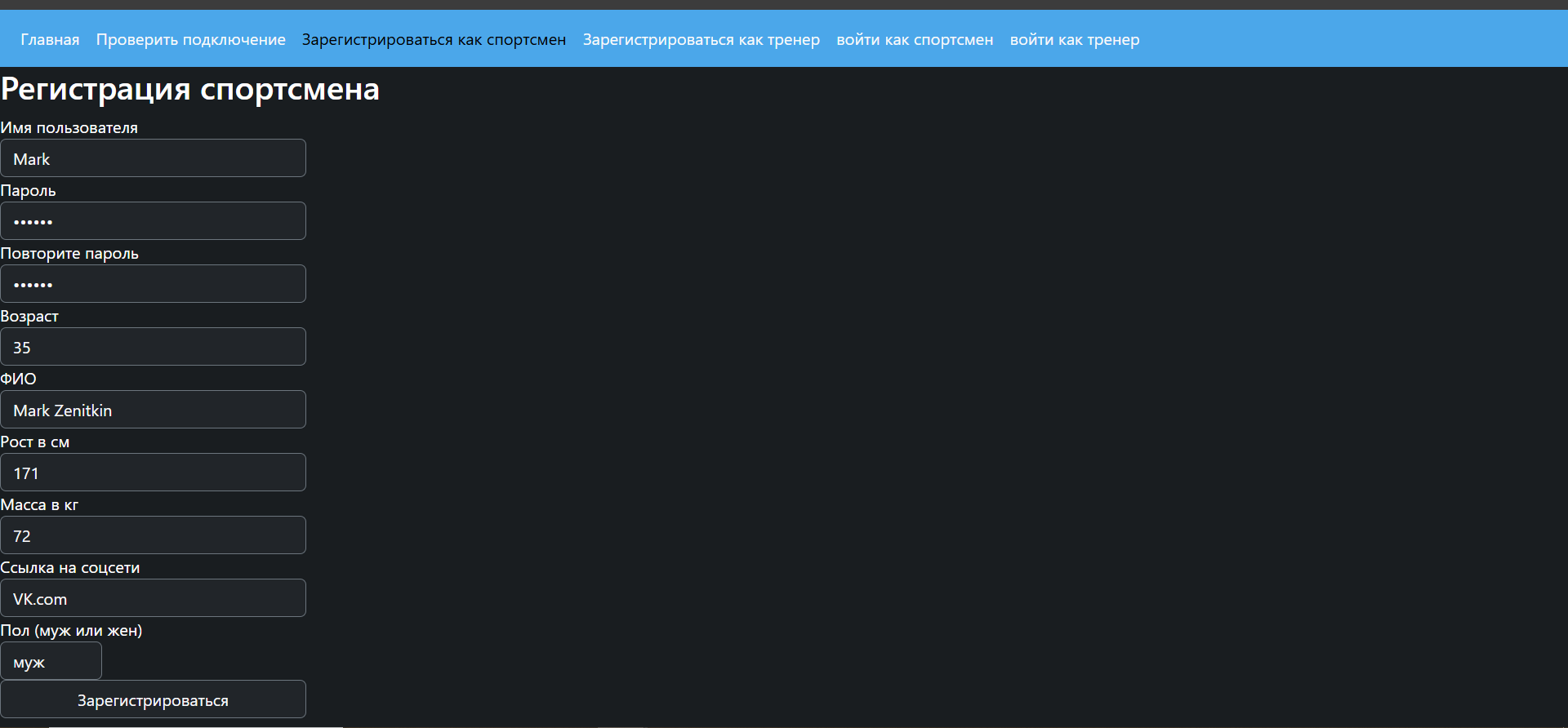


Рисунок 3 – страница регистрации спортсмена

Зарегистрироваться как тренер можно по кнопке «Зарегистрироваться как тренер»

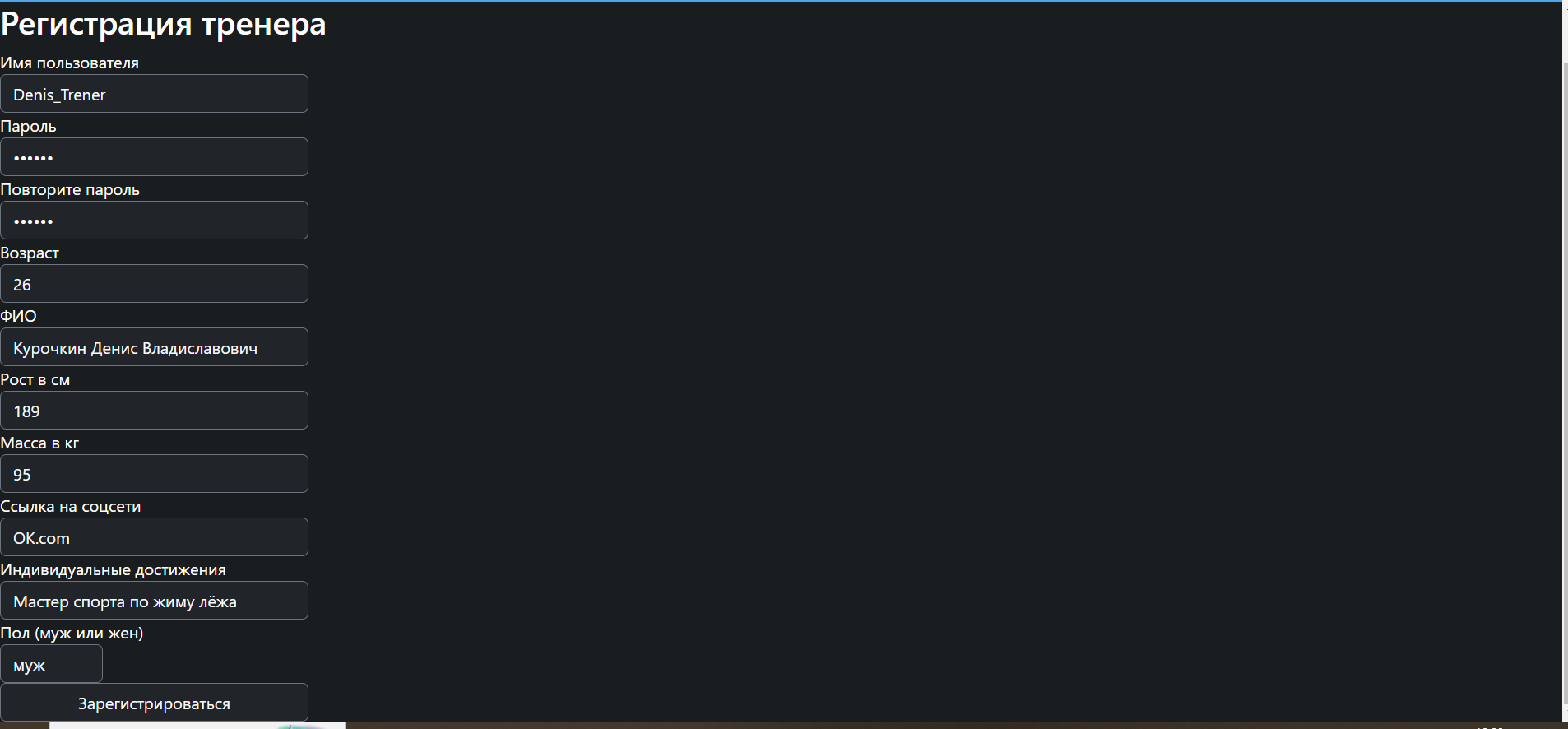


Рисунок 4 – страница регистрации тренера

Авторизоваться можно как спортсмен по кнопке «войти как спортсмен или войти как тренер» Страница авторизации для тренера выглядит аналогично.

После авторизации пользователь может перейти в свой профиль по кнопке «Профиль», где он сможет изменить уже имеющиеся или внести после регистрации данные о себе (Имя, пол, возраст, рост, вес, ссылка на соцсети) по кнопке «редактировать профиль».

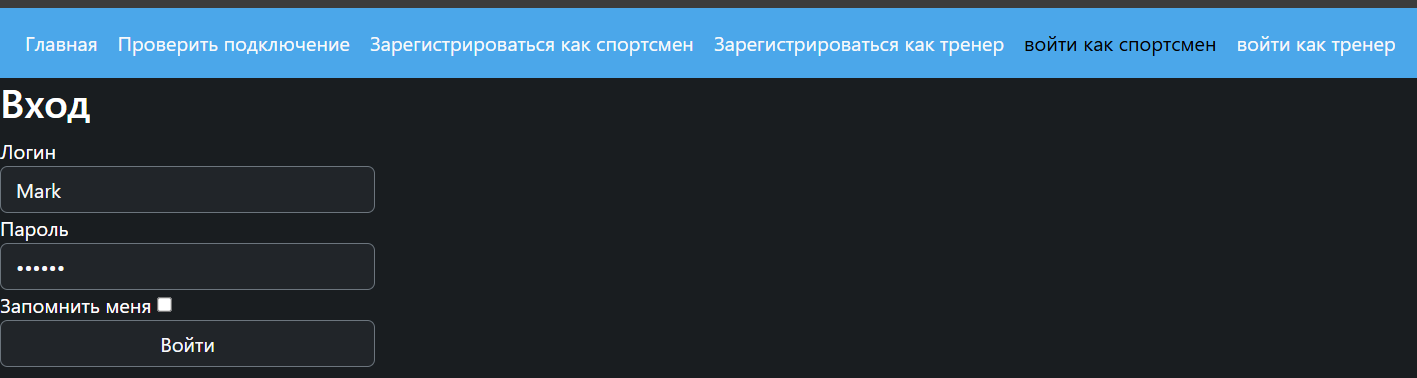


Рисунок 5 – страница авторизации

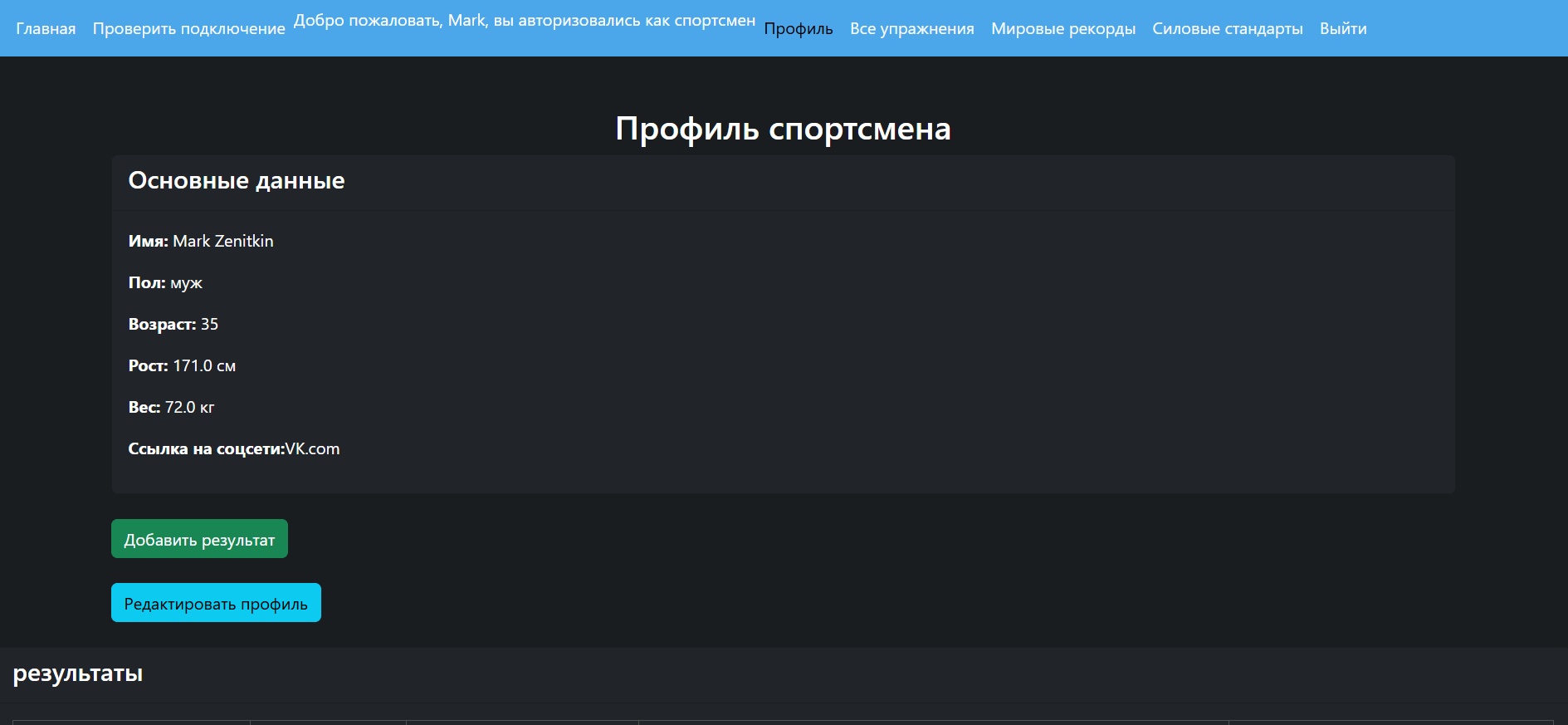


Рисунок 6 – страница профиля спортсмена

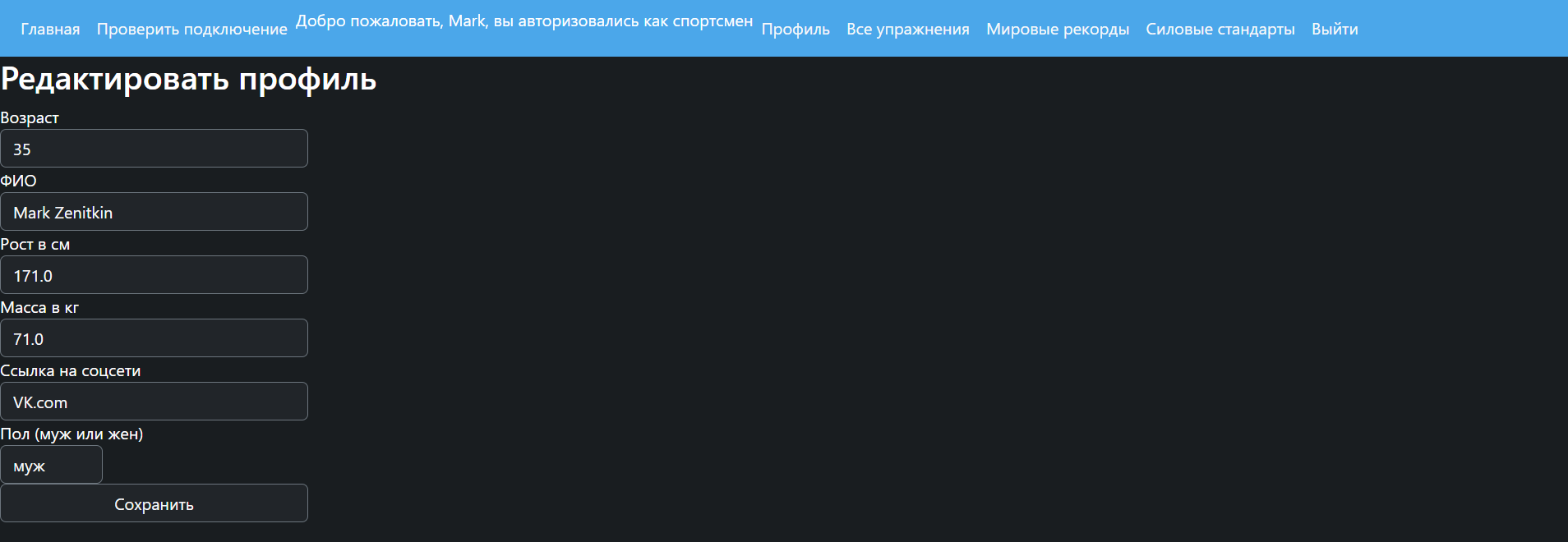


Рисунок 7 – страница редактирования профиля спортсмена

После завершения редактирования профиля пользователь перенаправляется обратно к себе в профиль.

В своём профиле спортсмен может добавить результаты выполнения упражнений по кнопке «Добавить результат».

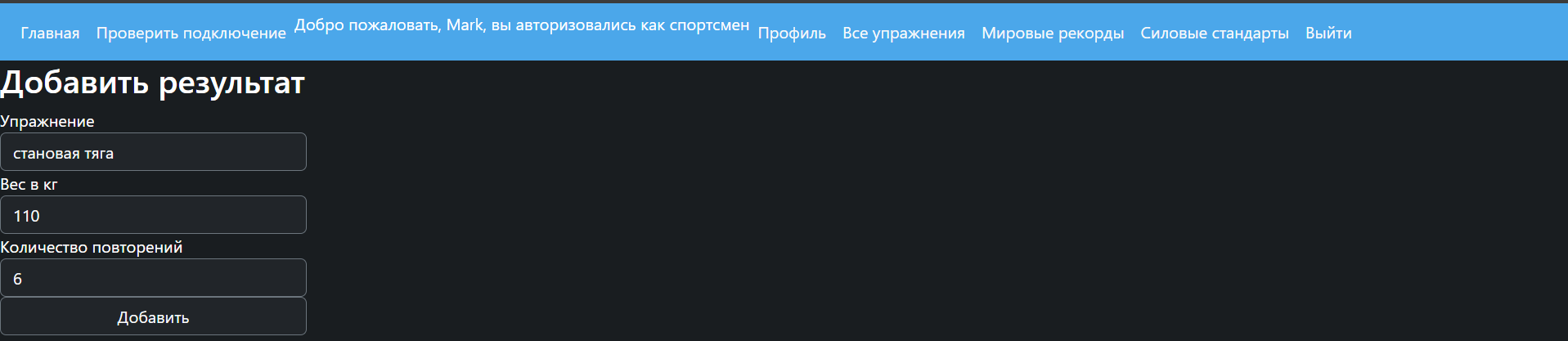


Рисунок 8 – страница добавления результата выполнения упражнения

После добавления результата пользователь перенаправляется обратно к себе в профиль, где можно увидеть только что добавленный результат.

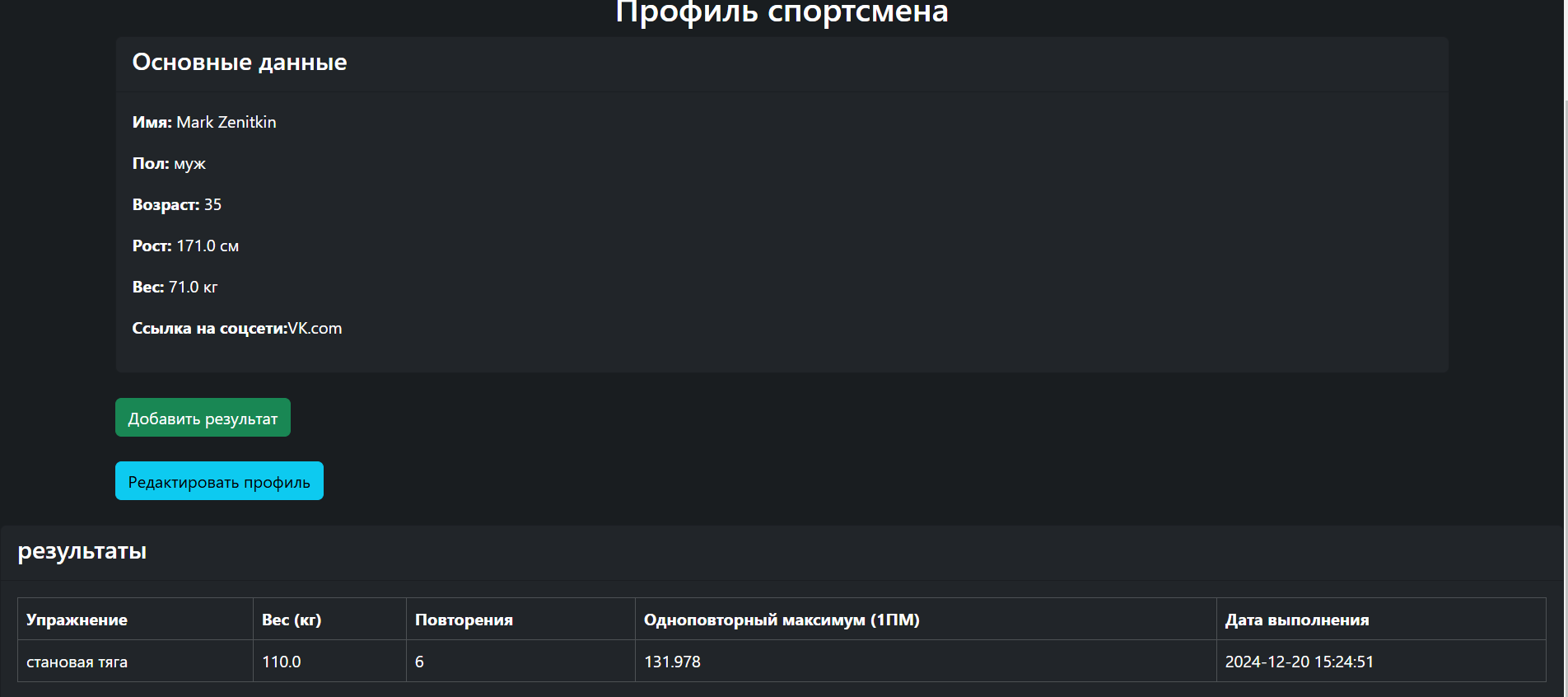


Рисунок 9 – отображение результатов выполнения упражнений в профиле

На главной странице отображается список тренеров и спортсменов, если пользователь авторизован, то он может переходить в профиль к другим пользователям и просматривать их данные (ФИО, пол и т.д.), результаты выполнения упражнений, публикации и отзывы (если зайти в профиль тренера), упражнения, мировые рекорды и силовые стандарты.

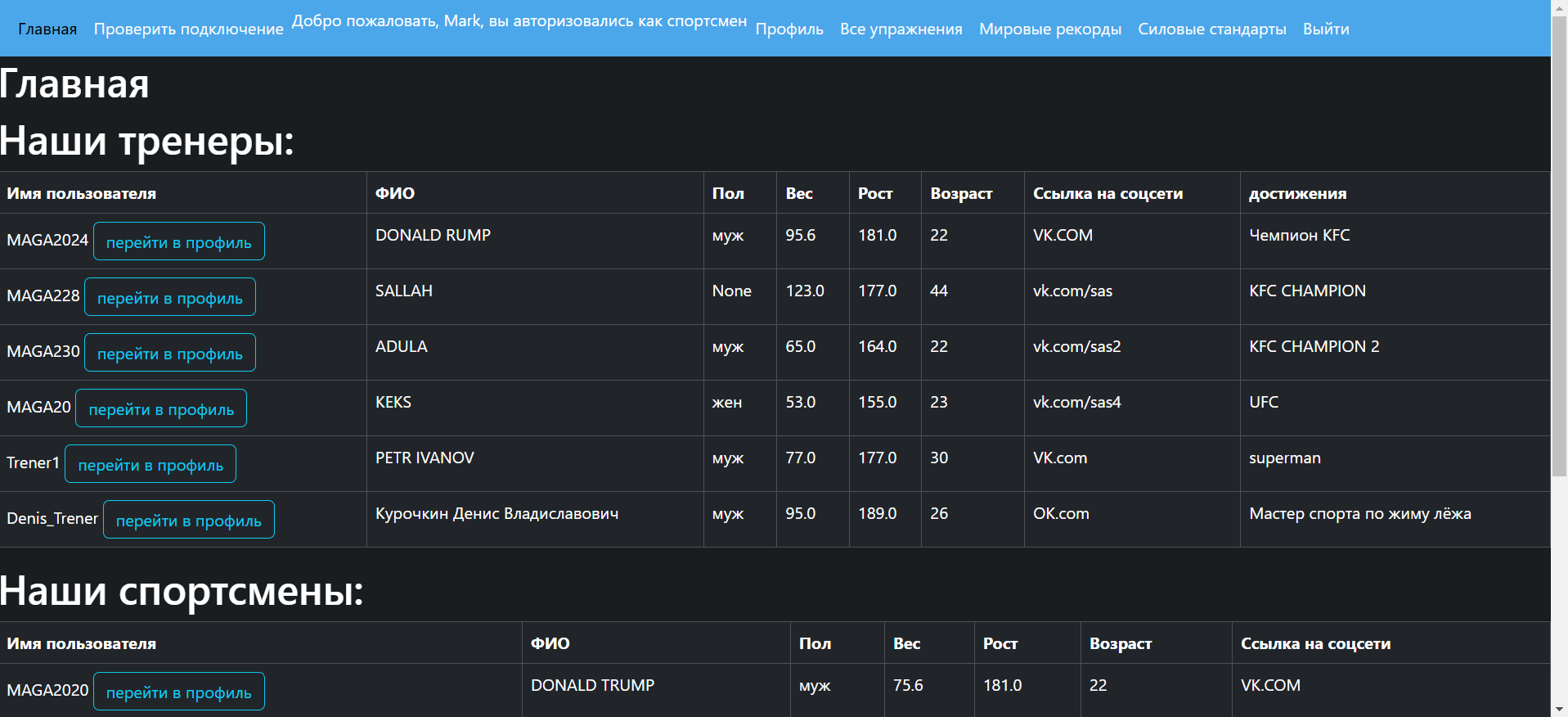


Рисунок 10 – главная страница сайта

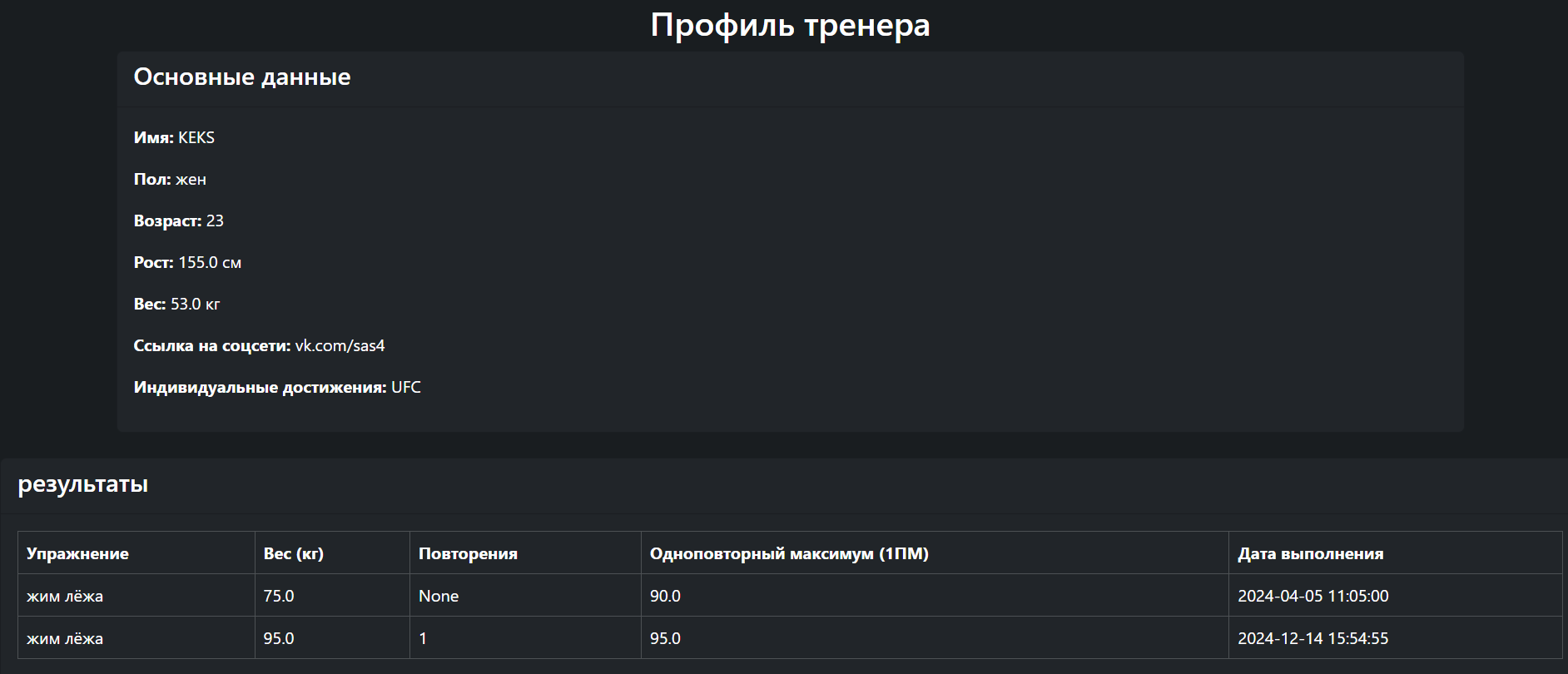


Рисунок 11 – профиль другого пользователя

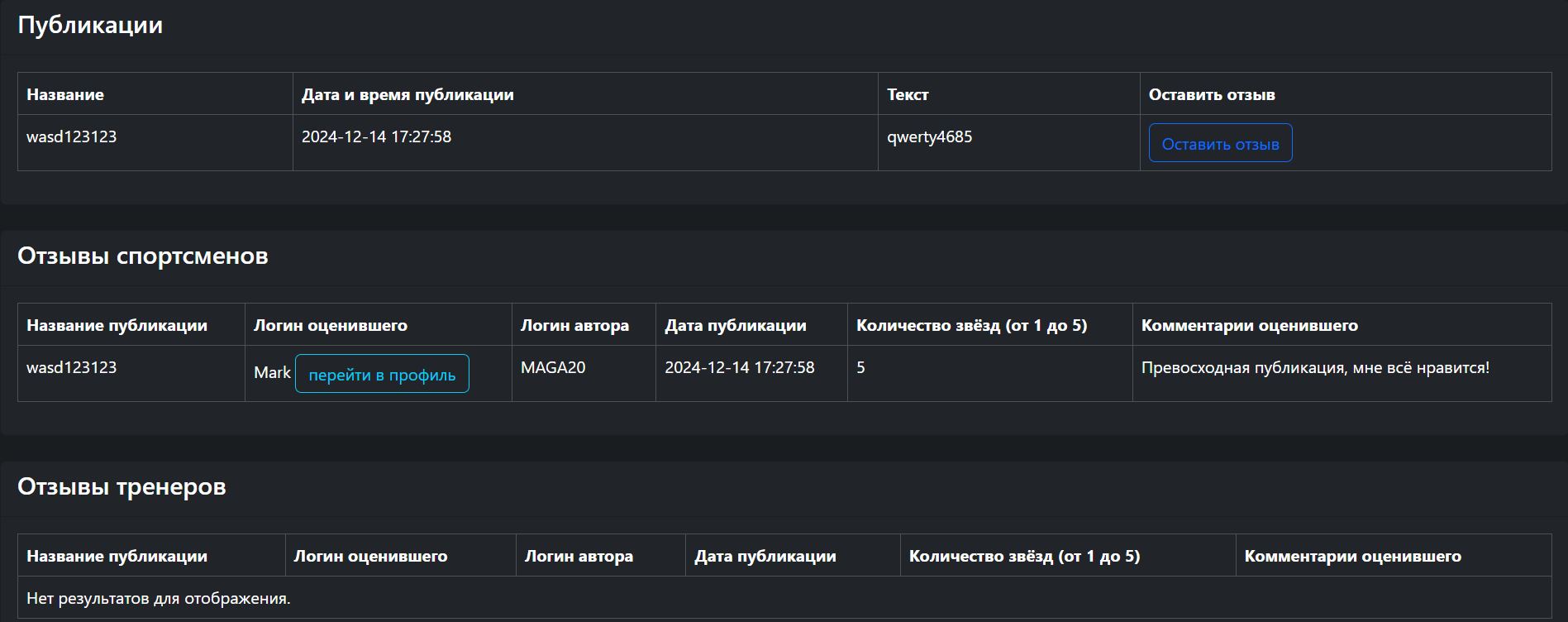


Рисунок 12 – публикации в профиле тренера и отзывы

Процесс регистрации, авторизации и добавления результата выполнения упражнения для тренера точно такой же, как и для спортсмена.

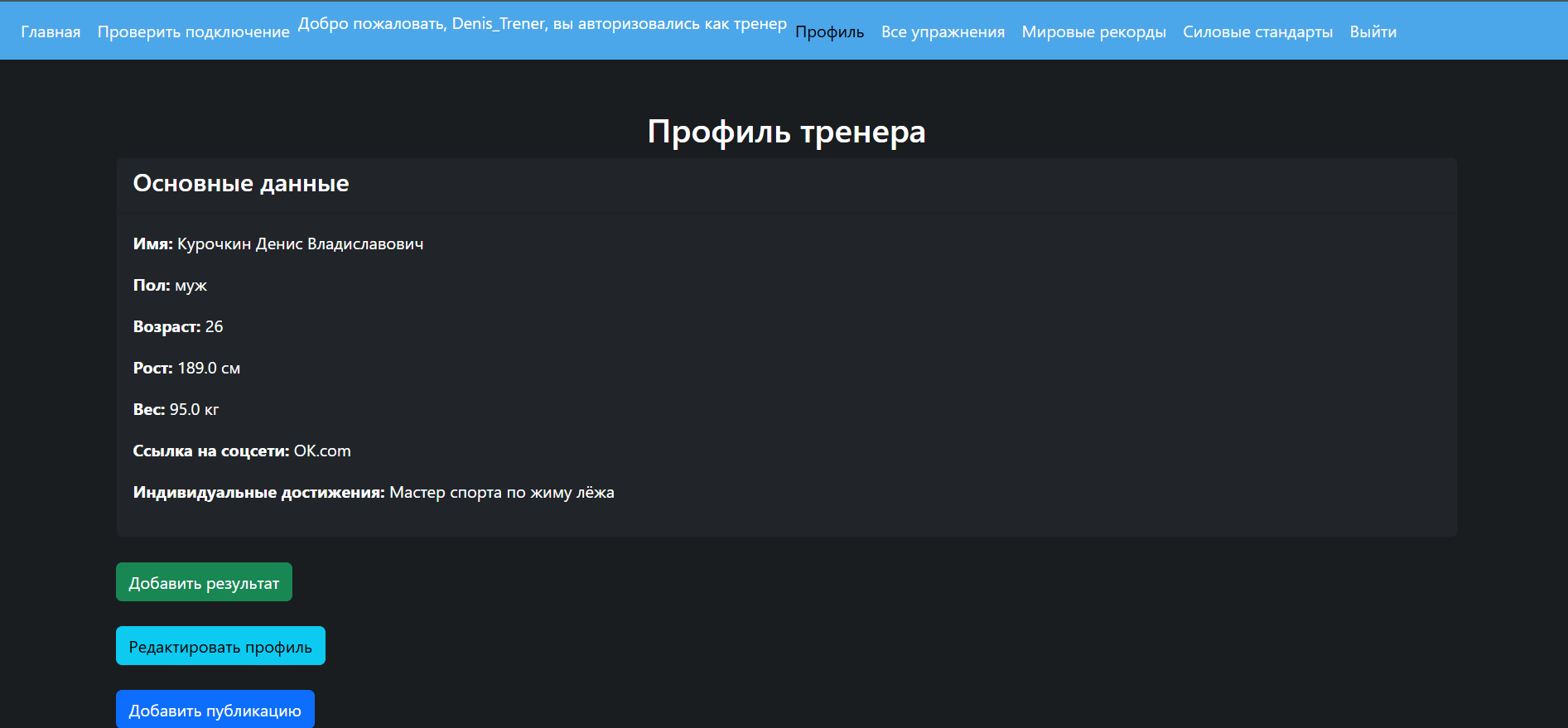


Рисунок 13 – профиль тренера

Добавление публикации можно начать по кнопке «Добавить публикацию».

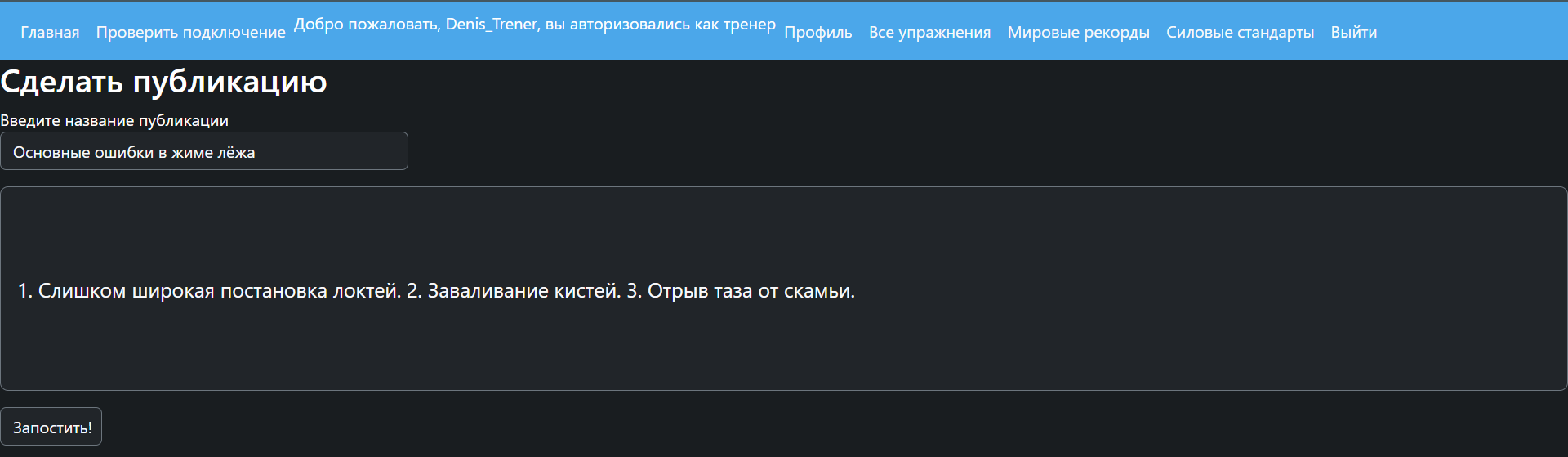


Рисунок 14 – страница добавления публикации

Пользователь может оставить отзыв можно по кнопке «Оставить отзыв» возле публикации.

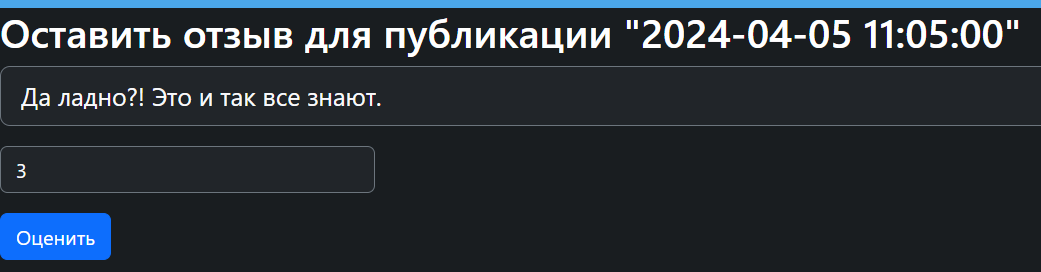


Рисунок 15 – страница добавления отзыва

Так же любому авторизованному пользователю доступны упражнения и техника выполнения, мировые рекорды и силовые стандарты.

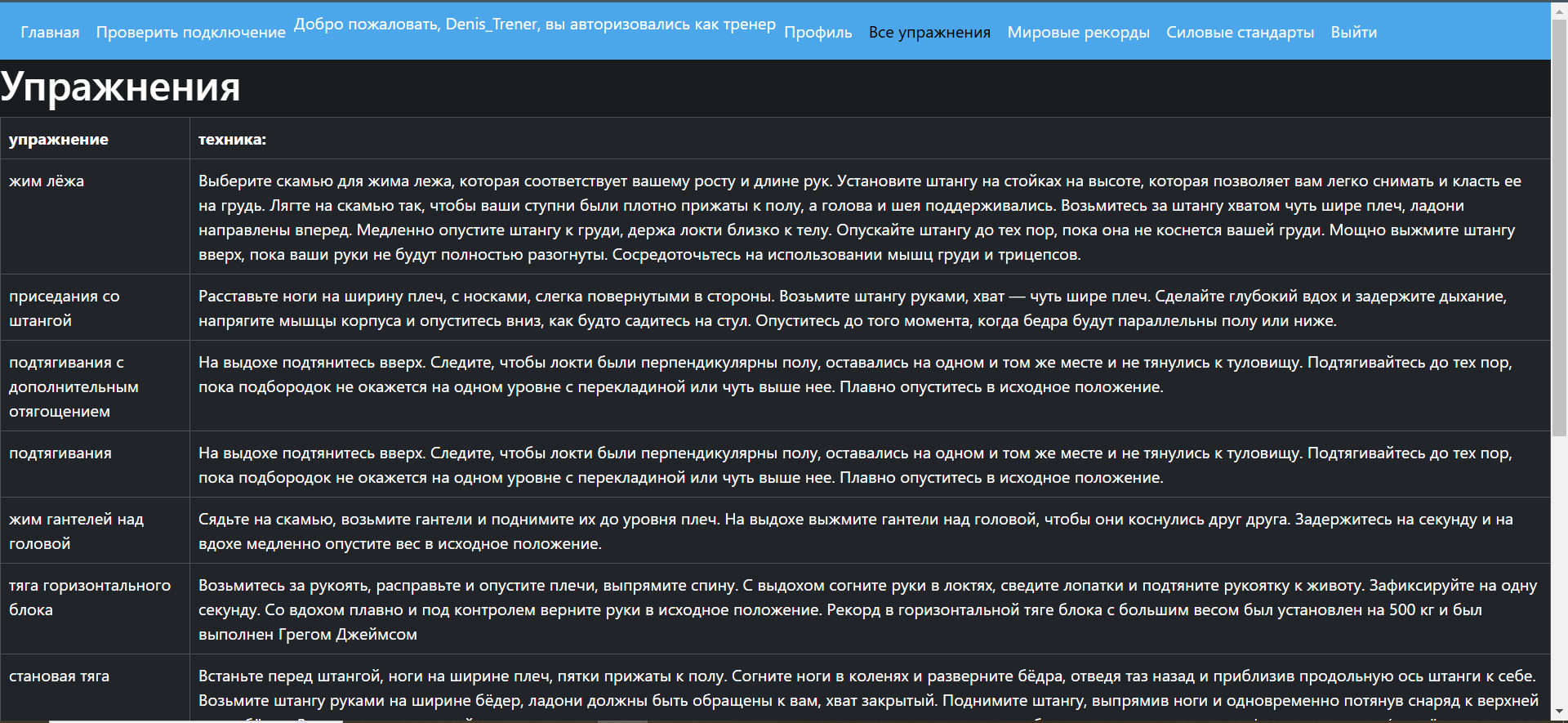


Рисунок 16 – страница со всеми упражнениями и техникой выполнения



Рисунок 17 – страница с мировыми рекордами в упражнениях



Рисунок 18 – страница с силовыми стандартами

**ЗАКЛЮЧЕНИЕ**

В ходе выполнения работы было разработано веб-приложение, которое позволяет пользователям отслеживать результаты своих тренировок, делиться тренировочными программами, советами по питанию и другой информацией, оставлять отзывы, просматривать информацию об упражнениях (мировые рекорды, техника выполнения, силовые стандарты), были приобретены базовые навыки веб-разработки, ознакомительно изучено облачное развертывание баз данных и веб-приложений, а также закреплены навыки проектирования реляционных баз данных на основе требований и ограничений предметной области.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Аксёнов А.В. Пособие «Разработка и развертывание веб-приложения на языке Python» (интернет-ресурс) //URL: <https://github.com/db-course/course-project-manual/blob/master/index.rst>