

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)

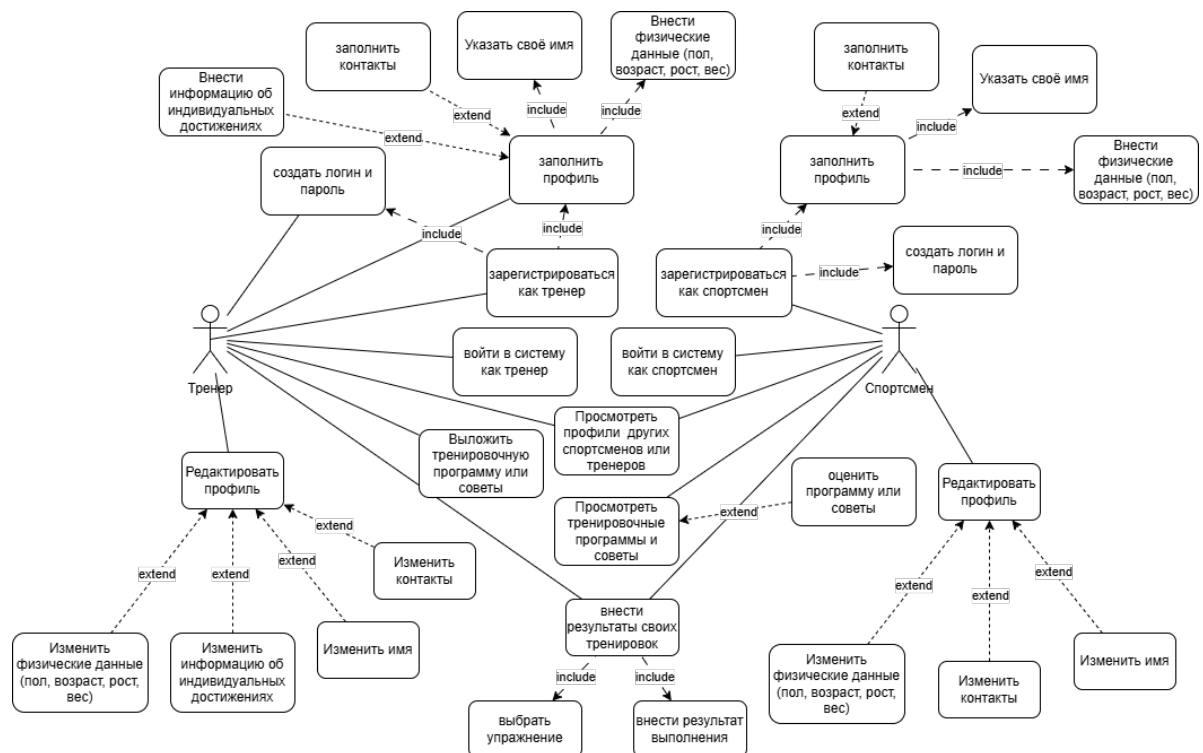


Рисунок 1 – диаграмма вариантов использования

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.

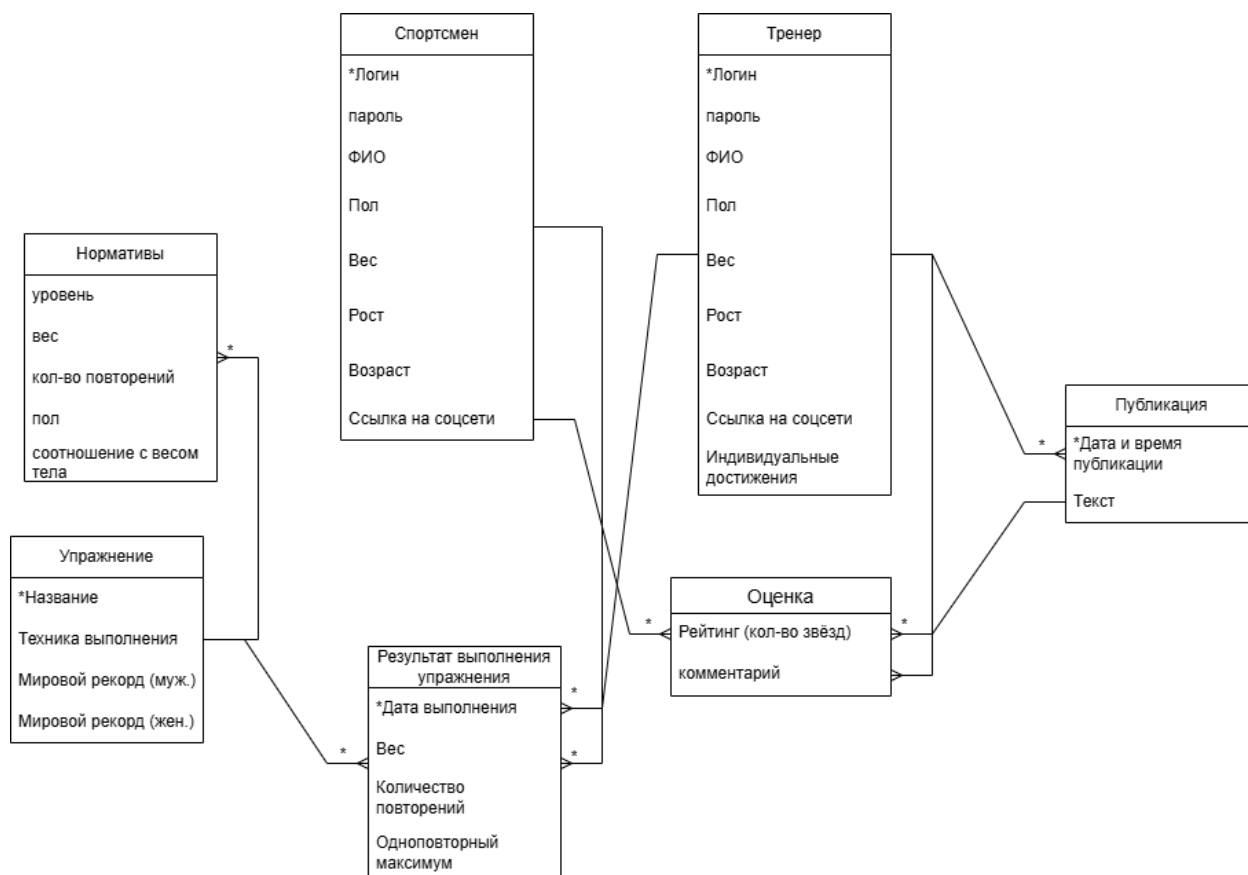


Рисунок 1 – схема сущность-связь для разработанной базы данных

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».

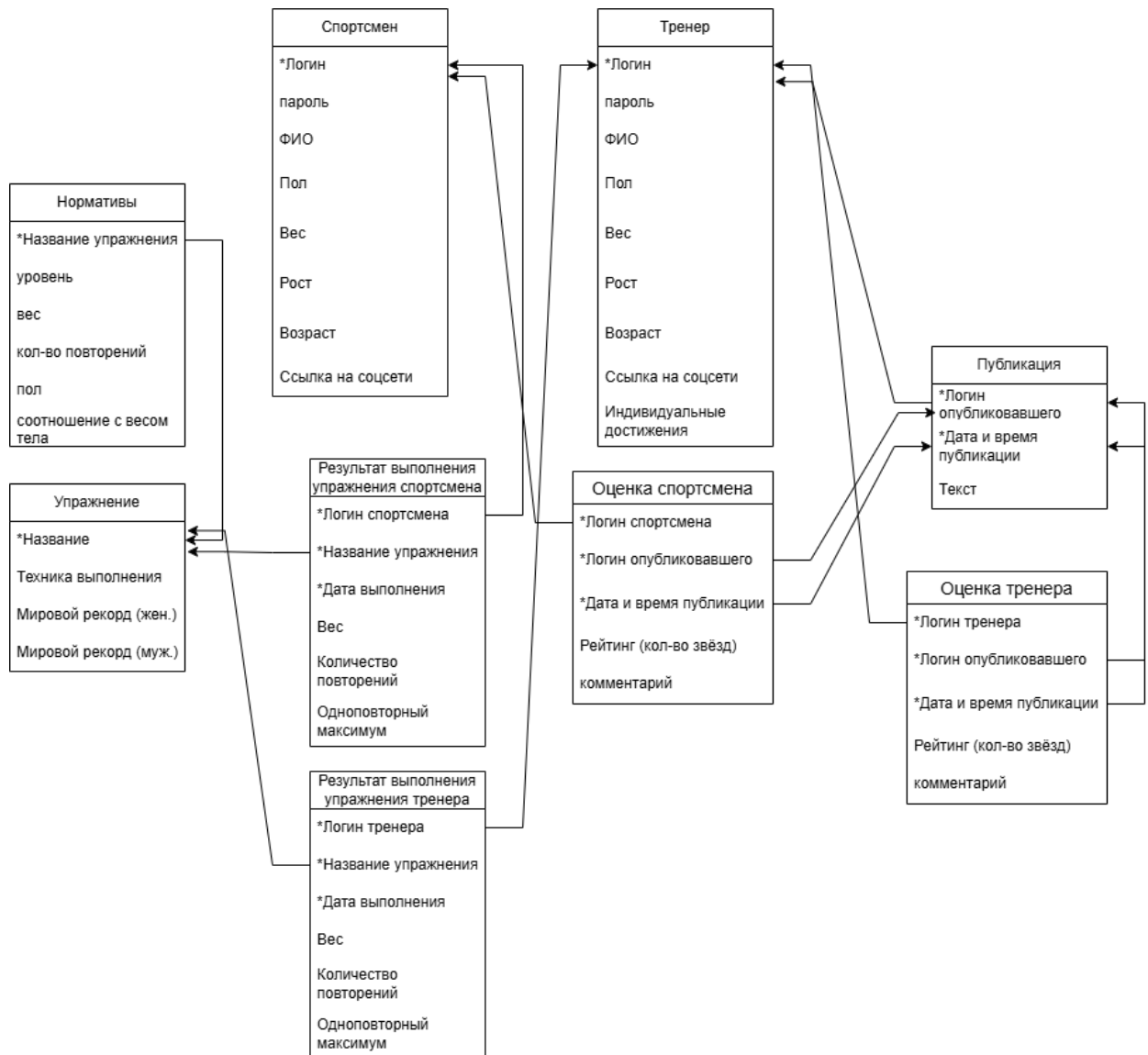


Рисунок 2 – реляционная схема базы данных

После создания реляционной схемы базы данных можно приступить 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 приступить 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 множеством 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-сервиса 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 соответственно).

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('/exercices') # вывод всех упражнений и техники выполнения
def get_exercices():
    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()
        exercices = cur.execute(f'SELECT nazv, technique FROM exercise').fetchall()
        return render_template('exercices.html',exercices=exercices)

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 соответствующими
            # 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 соответствующими
            # 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 psycopg2.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 psycopg2.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 psycopg2.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
спортсмeна
                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.us
ername))
            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,cu
rrent_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 упражнения
@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: # расчёт одноповторного максимума
            ONE_REP_MAX = 0
        elif (reps == 1):
            ONE_REP_MAX = weight
        else:
            ONE_REP_MAX = weight*(1+0.0333*reps)
        with psycopg2.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 psycopg2.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 psycopg2.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 psycopg2.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 pycorg
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 pycorg.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>
</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 %}
                        </tr>
                    {% endfor %}
                </tbody>
            </table>
        </div>
    </div>

```

```

        <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 %}
    </form>
{% endblock %}

```

```

        <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 %}
    </form>
{% endblock %}

```



```

        <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 %}
    </form>

```

```

    <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>
    {{ 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_exercises', '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 %}

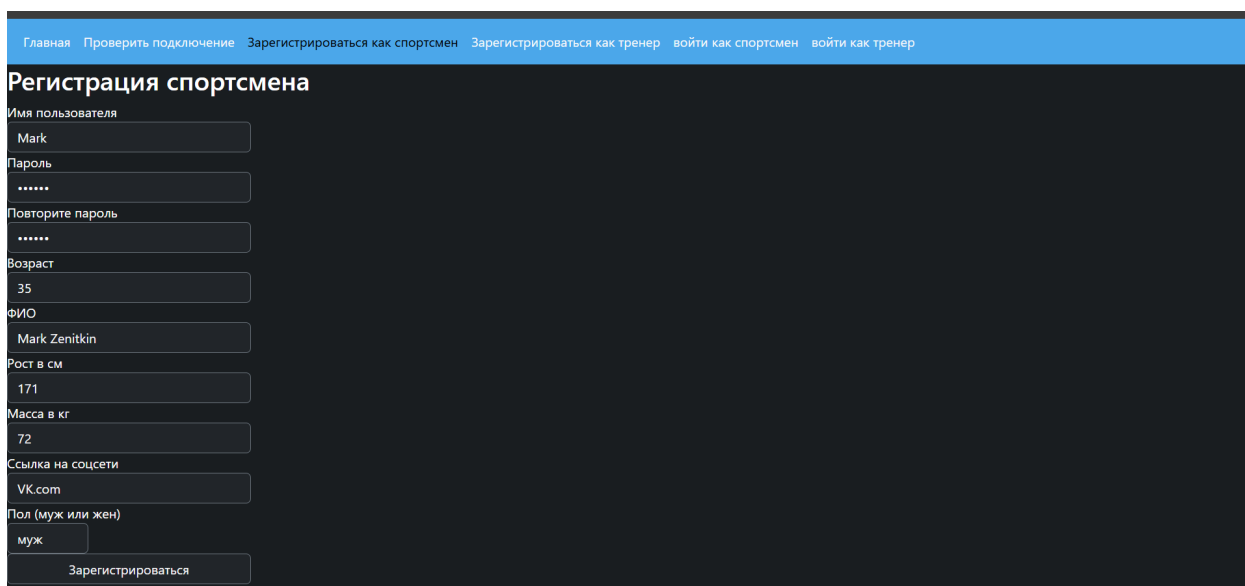
            </div>
        </div>
    </nav>
    {% block content %}{% endblock %}
    {% block scripts %}
        {{ bootstrap.load_js() }}
    {% endblock %}
</body>

```

</html>

Тестирование функционала

Зарегистрироваться как спортсмен можно по кнопке «Зарегистрироваться как спортсмен»



The screenshot shows a web form titled "Регистрация спортсмена" (Sportsman Registration). At the top, there is a navigation bar with links: Главная, Проверить подключение, Зарегистрироваться как спортсмен, Зарегистрироваться как тренер, войти как спортсмен, and войти как тренер. The form fields are as follows:

Field Label	Value
Имя пользователя	Mark
Пароль	*****
Повторите пароль	*****
Возраст	35
ФИО	Mark Zenitkin
Рост в см	171
Масса в кг	72
Ссылка на соцсети	VK.com
Пол (муж или жен)	муж
<input type="button" value="Зарегистрироваться"/>	

Рисунок 3 – страница регистрации спортсмена

Зарегистрироваться как тренер можно по кнопке «Зарегистрироваться как тренер»



The screenshot shows a web form titled "Регистрация тренера" (Coach Registration). The form fields are as follows:

Field Label	Value
Имя пользователя	Denis_Trener
Пароль	*****
Повторите пароль	*****
Возраст	26
ФИО	Курочкин Денис Владиславович
Рост в см	189
Масса в кг	95
Ссылка на соцсети	OK.com
Индивидуальные достижения	Мастер спорта по жиму лёжа
Пол (муж или жен)	муж
<input type="button" value="Зарегистрироваться"/>	

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».

The screenshot shows a login page with a blue header containing navigation links: Главная, Проверить подключение, Зарегистрироваться как спортсмен, Зарегистрироваться как тренер, войти как спортсмен, and войти как тренер. The main content area is dark gray and titled 'Вход'. It contains a 'Логин' field with the text 'Mark', a 'Пароль' field with six dots, a 'Запомнить меня' checkbox, and a 'Войти' button.

Рисунок 5 – страница авторизации

The screenshot shows an athlete's profile page. The blue header includes links: Главная, Проверить подключение, Добро пожаловать, Mark, вы авторизовались как спортсмен, Профиль, Все упражнения, Мировые рекорды, Силовые стандарты, and Выйти. The main content area is dark gray and titled 'Профиль спортсмена'. It features a 'Основные данные' section with the following information: Имя: Mark Zenitkin, Пол: муж, Возраст: 35, Рост: 171.0 см, Вес: 72.0 кг, and Ссылка на соцсети: VK.com. Below this section are two buttons: 'Добавить результат' (green) and 'Редактировать профиль' (blue). At the bottom left, the word 'результаты' is visible.

Рисунок 6 – страница профиля спортсмена

The screenshot shows the profile editing page. The blue header is identical to the previous page. The main content area is dark gray and titled 'Редактировать профиль'. It contains several input fields for editing personal data: 'Возраст' (35), 'ФИО' (Mark Zenitkin), 'Рост в см' (171.0), 'Масса в кг' (71.0), 'Ссылка на соцсети' (VK.com), and 'Пол (муж или жен)' (муж). A 'Сохранить' button is located at the bottom right.

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».

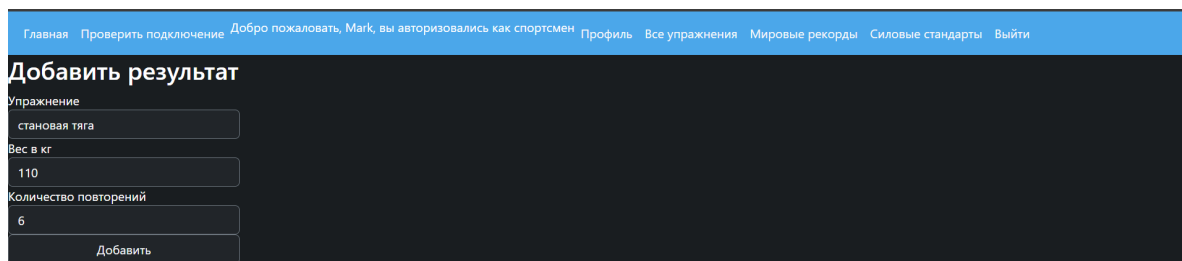
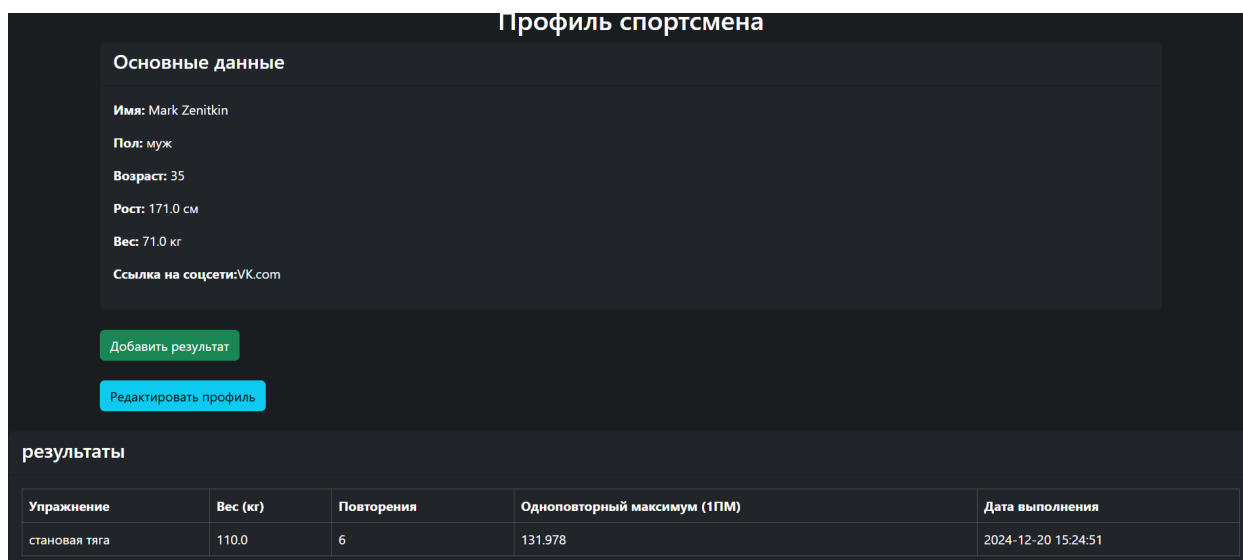


Рисунок 8 – страница добавления результата выполнения упражнения

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



Упражнение	Вес (кг)	Повторения	Одноповторный максимум (1ПМ)	Дата выполнения
становая тяга	110.0	6	131.978	2024-12-20 15:24:51

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.

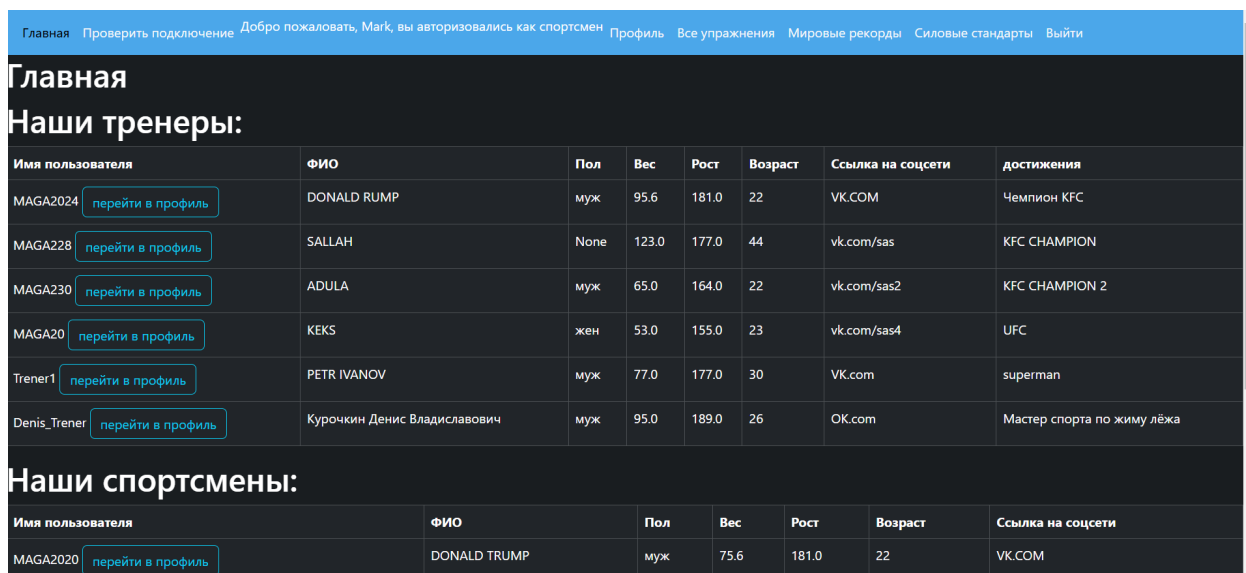


Рисунок 10 – главная страница сайта

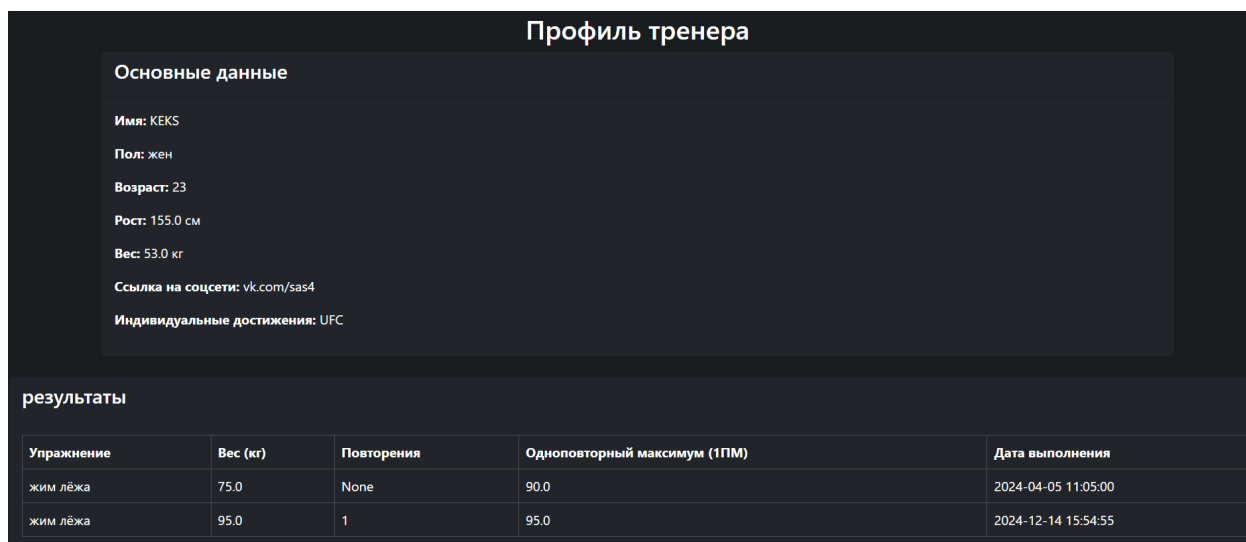


Рисунок 11 – профиль другого пользователя

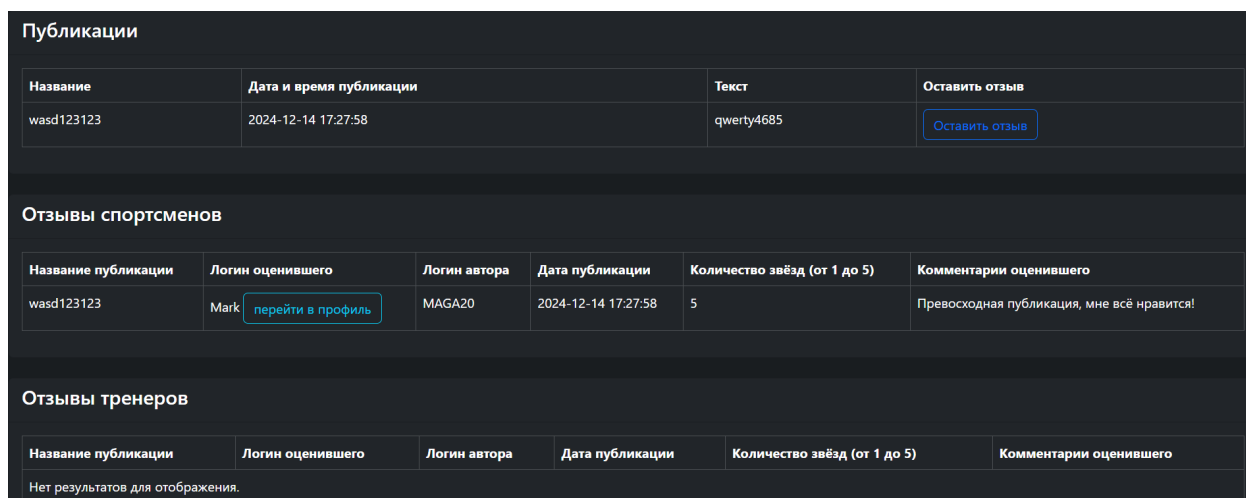


Рисунок 12 – публикации в профиле тренера и отзывы

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

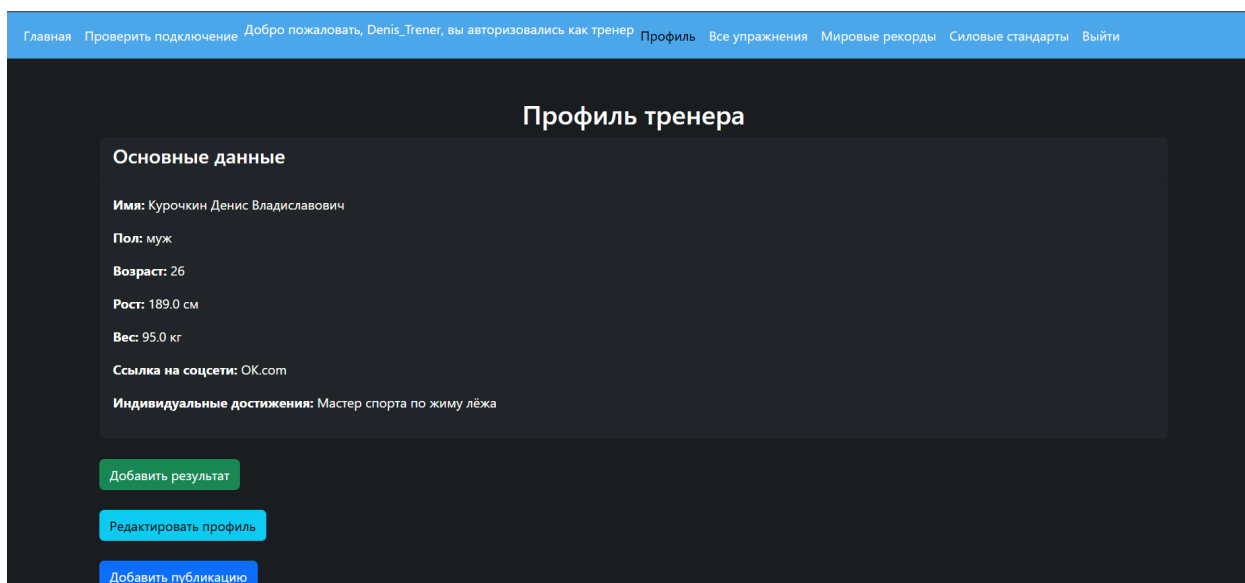


Рисунок 13 – профиль тренера

Добавление публикации можно начать по кнопке «Добавить публикацию».

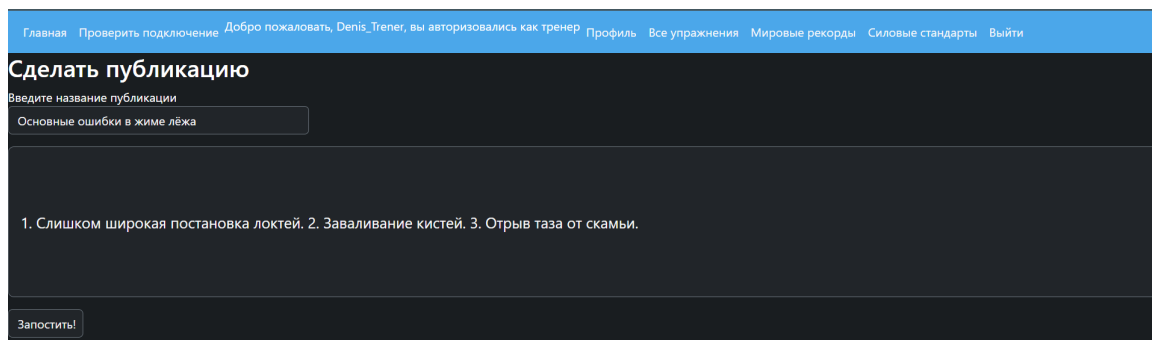


Рисунок 14 – страница добавления публикации

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

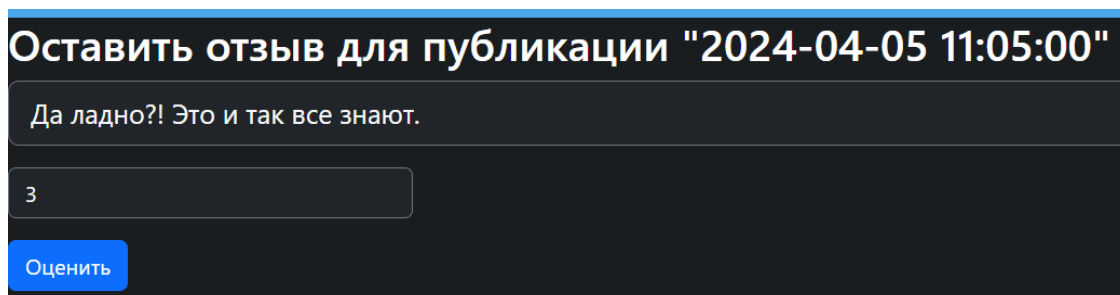


Рисунок 15 – страница добавления отзыва

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

Главная Проверить подключение Добро пожаловать, Denis_Trener, вы авторизовались как тренер Профиль Все упражнения Мировые рекорды Силовые стандарты Выйти	
Упражнения	
упражнение	техника:
жим лёжа	Выберите скамью для жима лежа, которая соответствует вашему росту и длине рук. Установите штангу на стойках на высоте, которая позволяет вам легко снимать и класть ее на грудь. Лягте на скамью так, чтобы ваши ступни были плотно прижаты к полу, а голова и шея поддерживались. Возьмитесь за штангу хватом чуть шире плеч, ладони направлены вперед. Медленно опустите штангу к груди, держа локти близко к телу. Опускайте штангу до тех пор, пока она не коснется вашей груди. Мощно выжмите штангу вверх, пока ваши руки не будут полностью разогнуты. Сосредоточьтесь на использовании мышц груди и трицепсов.
приседания со штангой	Расставьте ноги на ширину плеч, с носками, слегка повернутыми в стороны. Возьмите штангу руками, хват — чуть шире плеч. Сделайте глубокий вдох и задержите дыхание, напрягите мышцы корпуса и опуститесь вниз, как будто садитесь на стул. Опуститесь до того момента, когда бедра будут параллельны полу или ниже.
подтягивания с дополнительным отягощением	На выдохе подтянитесь вверх. Следите, чтобы локти были перпендикулярны полу, оставались на одном и том же месте и не тянулись к туловищу. Подтягивайтесь до тех пор, пока подбородок не окажется на одном уровне с перекладиной или чуть выше нее. Плавное опуститесь в исходное положение.
подтягивания	На выдохе подтянитесь вверх. Следите, чтобы локти были перпендикулярны полу, оставались на одном и том же месте и не тянулись к туловищу. Подтягивайтесь до тех пор, пока подбородок не окажется на одном уровне с перекладиной или чуть выше нее. Плавное опуститесь в исходное положение.
жим гантелей над головой	Сядьте на скамью, возьмите гантели и поднимите их до уровня плеч. На выдохе выжмите гантели над головой, чтобы они коснулись друг друга. Задержитесь на секунду и на вдохе медленно опустите вес в исходное положение.
тяга горизонтального блока	Возьмитесь за рукоятку, расправьте и опустите плечи, выпрямите спину. С выдохом согните руки в локтях, сведите лопатки и подтяните рукоятку к животу. Зафиксируйте на одну секунду. Со вдохом плавно и под контролем верните руки в исходное положение. Рекорд в горизонтальной тяге блока с большим весом был установлен на 500 кг и был выполнен Греггом Джеймсом
становая тяга	Встаньте перед штангой, ноги на ширине плеч, пятки прижаты к полу. Согните ноги в коленях и разверните бедра, отведя таз назад и приблизив продольную ось штанги к себе. Возьмите штангу руками на ширине бедер, ладони должны быть обращены к вам, хват закрытый. Поднимите штангу, выпрямив ноги и одновременно потянув снаряд к верхней

Рисунок 16 – страница со всеми упражнениями и техникой выполнения

Рекорды		
упражнение	Рекорд(муж):	Рекорд(жен):
жим лёжа	Абсолютный мировой рекорд по жиму лёжа без экипировки — 355 кг (21 февраля 2021), принадлежит Джулиусу Мэддоксу (Julius Maddox).	207,5кг Эйприл Мэтис 12.03.2016 SPF США
приседания со штангой	Абсолютный мировой рекорд в многослойной экипировке принадлежит Брайану Кэрролу (США) и составляет 592,3 килограмма	320кг Эйприл Мэтис 04.01.2017 APF США
подтягивания с дополнительным отягощением	Мировой рекорд по подтягиваниям с дополнительным весом был установлен Мартином Штайнмейером, который подтянулся с дополнительным весом 150 кг.	Рекорды среди женщин с дополнительным весом также варьируются, но рекорд по подтягиваниям с 70 кг был установлен Миа Миглиори
подтягивания	38-летний преподаватель Гарвардской школы права Адам Сэндел подтянулся 77 раз	когда Даше исполнилось 14 лет, она не просто показала лучший результат, а установила мировой рекорд на том же чемпионате. Девочка выполнила подтягивания 42 раза подряд
жим гантелей над головой	Рекорд по жиму гантелей над головой с максимальным весом составил 80 кг на одну руку. Этот рекорд был установлен известным бодибилдером Ронни Колеманом.	Рекорд среди женщин составляет около 40-45 кг на одну руку, хотя такие данные могут варьироваться в зависимости от соревнования.
тяга горизонтального блока	Рекорд в горизонтальной тяге блока с большим весом был установлен на 500 кг и был выполнен Греггом Джеймсом.	Рекорд среди женщин составляет около 200-250 кг.
становая тяга	Дэнни Григсби (Danny Grigsby). Ему удалось поднять 487,5 кг (1074 фунта) на WRPF American Pro 2022 года	Тамара Уолкотт (Tamara Walcott) – королева безэкипировочной становой тяги, так как у нее самая тяжелая тяга в истории женского пауэрлифтинга. Ее лучшая становая тяга пришлась на WRPF American Pro 2022 года, когда она смогла зафиксировать 290 кг (639 фунтов).
жим гантелей на	рекорд для жима гантелей на бицепс в одном повторении муж	Рекорды для женщин варьируются от 20 до 30 кг в зависимости от уровня подготовки

Рисунок 17 – страница с мировыми рекордами в упражнениях

Силовые стандарты					
упражнение	Уровень:	Вес	Количество повторений	Пол	Соотношение с весом тела
жим гантелей над головой	элита	60.0	1	муж	0.75
жим гантелей над головой	продвинутый	45.0	1	муж	0.6
жим гантелей над головой	тренированный	32.0	1	муж	0.4
жим гантелей над головой	начинающий	21.0	1	муж	0.25
жим гантелей над головой	нетренированный	13.0	1	муж	0.15
жим лёжа	элита	169.0	1	муж	2.0
жим лёжа	продвинутый	98.0	1	муж	1.75
жим лёжа	тренированный	132.0	1	муж	1.25
жим лёжа	начинающий	70.0	1	муж	0.75
жим лёжа	нетренированный	47.0	1	муж	0.5
подтягивания	элита	None	37	муж	5.0
подтягивания	продвинутый	None	25	муж	4.0
подтягивания	тренированный	None	14	муж	3.0

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>