МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
**«Южно-Уральский государственный университет
(национальный исследовательский университет)»**
Высшая школа электроники и компьютерных наук
Кафедра системного программирования

РАБОТА ПРОВЕРЕНА

Рецензент
Доцент кафедры ВМиИТ
ФГБОУ ВО «ЧелГУ», к.ф.-м.н.

_____ А.Ю. Маковецкий

«___»_____ 2024 г.

ДОПУСТИТЬ К ЗАЩИТЕ

Заведующий кафедрой, д.ф.-м.н., профессор

_____ Л.Б. Соколинский

«___»_____ 2024 г.

Разработка веб-сервиса для создания программ тренировок на основе индивидуальных показателей клиента фитнес-центра

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
ЮУрГУ – 09.04.04.2024.308-1470.ВКР

Научные руководители:
доцент кафедры СП, к.ф.-м.н.
_____ С.У. Турлакова,

ст. преподаватель кафедры СП
_____ Н.С. Силкина

Автор работы,
студент группы КЭ-228
_____ А.А. Валиулин

Ученый секретарь
(нормоконтролер)
_____ И.Д. Володченко
«___»_____ 2024 г.

Челябинск, 2024 г.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Высшая школа электроники и компьютерных наук
Кафедра системного программирования

УТВЕРЖДАЮ
Зав. кафедрой СП
_____ Л.Б. Соколинский
29.01.2024 г.

ЗАДАНИЕ
на выполнение выпускной квалификационной работы магистранта
студенту группы КЭ-228
Валиулину Александру Артуровичу,
обучающемуся по направлению
09.04.04 «Программная инженерия»
(магистерская программа «Искусственный интеллект и инженерия данных»)

1. **Тема работы** (утверждена приказом ректора от 22.04.2024 г. № 764-13/12)
Разработка веб-сервиса для создания программ тренировок на основе индивидуальных показателей клиента фитнес-центра.
2. **Срок сдачи студентом законченной работы:** 20.05.2024 г.
3. **Исходные данные к работе**

3.1. Старолетов С.М. Основы тестирования и верификации программного обеспечения: учебное пособие. – Санкт–Петербург: Лань, 2020. – 344 с.

3.2. Tesseract. [Электронный ресурс] URL: https://github.com/tesseract-ocr/tesseract (дата обращения: 21.04.2024 г.).

3.3. Pytesseract. 0.3.10 [Электронный ресурс] URL: https://pypi.org/project/pytesseract/ (дата обращения: 21.04.2024 г.).

4. **Перечень подлежащих разработке вопросов**

4.1. Провести обзор аналогичных приложений и сервисов.

4.2. Разработать приложение для сбора данных.

4.3. Провести тестирование приложения для сбора данных.

4.4. Спроектировать и реализовать веб-сервис.

4.5. Провести тестирование разработанного веб-сервиса.

5. Дата выдачи задания: 29.01.2024 г.

Научный руководитель,
доцент кафедры СП, к.ф.-м.н. С.У. Турлакова

ст. преподаватель кафедры СП Н.С. Силкина

Задание принял к исполнению А.А. Валиулин

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Актуальность

С каждым годом все больше людей осознают важность здоровья своего физического и психического благополучия. Фитнес занятия становятся неотъемлемой частью повседневной жизни для многих людей. Тем самым, спрос на персонализированные программы тренировок растет.

С развитием технологий мы видим возникновение новых инновационных подходов в сфере здоровья и фитнеса. Умные устройства и аналитика позволяют получать и отслеживать персонализированные физические параметры каждого человека.

Так как, каждый человек уникален, и то, что подходит для одного, не всегда подходит для другого. Создание персонализированных программ тренировок позволяет учитывать физиологические особенности, уровень подготовки, наличие травм и другие факторы, что повышает эффективность и безопасность занятий.

Разрабатываемое решение по сравнению с имеющимися аналогами предполагает использование искусственного интеллекта для создания программ тренировок. Имеющиеся на данный момент решения построения программ на основании данных клиента основываются на опыте тренеров и статистических таблиц, что является очень трудозатратой и не всегда эффективной операцией. Использование искусственного интеллекта позволит снизить трудоемкость работы и учесть информацию о не эффективно построенных программах.

Постановка задачи

Целью выпускной квалификационной работы является разработка веб-сервиса для создания программ тренировок на основе индивидуальных показателей клиента фитнес центра. Для достижения поставленной цели необходимо решить следующие задачи:

1) провести обзор аналогичных приложений и сервисов;
2) разработать приложение для сбора данных;

3) провести тестирование приложения для сбора данных;
4) спроектировать и реализовать веб-сервис;
5) провести тестирование разработанного веб-сервиса.

Структура и содержание работы

Работа состоит из введения, пяти глав, заключения и списка литературы. Объем работы составляет 51 страницу, объем списка литературы – 27 источников.

В первой главе описывается анализ предметной области. Рассматриваются существующие на данный момент аналоги, их преимущества и недостатки, а также формируются первоначальные требования к веб-сервису.

Вторая глава посвящена сбору данных и формированию обучающей выборки. Рассмотрению устройства, с которого будут поступать данные, а также разработке клиент серверного приложения, для сбора данных с устройства и отправки их для хранения на сервер. Также в данной главе рассказывается о сборе данных в виде анкетирования.

В третьей главе описывается обучение моделей машинного обучения.

В четвертой главе описывается проектирование веб-сервиса. Формируются функциональные и нефункциональные требования к веб-сервису, а также рассматривается проектирование базы данных и архитектура веб-приложения.

В пятой главе описана реализации веб-сервиса.

В шестой главе описывается тестирование разработанного веб-сервиса.

1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

Предметная область сервиса

Разрабатываемый в рамках данной работы сервис, будет представлять собой развернутое на удаленном сервере веб-приложение.

Основной задачей является создание индивидуальных программ фитнес тренировок на основании персональных характеристик каждого пользователя. Для этого сервис должен предоставить общедоступный интерфейс (API интерфейс), через который пользователь сможет указать какие программы тренировок ему требуются, какой результат он хочет получить, какими физическими характеристиками он обладает.

В свою очередь, сервис на основании данных пользователя должен создать индивидуальный план тренировок для пользователя и передать их пользователю по средствам интерфейса или результата запроса.

Обзор аналогов

По результатам проведенного анализа аналогов были выявлены несколько аналогов [1–3], которые предоставляют возможность производить автоматическое построение программ тренировок. Однако, полученные программы требуют корректировки с профессиональным тренером. Это означает, что хотя автоматическое построение программ тренировок предлагается, необходимо вмешательство тренера, чтобы адаптировать программы к индивидуальным потребностям и возможностям клиента.

Среди косвенных аналогов [4, 5] выделены приложения, которые занимаются мониторингом показателей и контролем выполнения конкретных упражнений. Однако, эти приложения не предоставляют комплексных упорядоченных наборов упражнений, необходимых для достижения желаемого результата. Приложения фокусируются лишь на отдельных аспектах тренировок, но не предлагают полноценные программы тренировок. Дополнительным недостатком аналогов является отсутствие возможности производить интеграцию с другими приложениями, занимающимися построением программ или занимающимися мониторингом фитнес трениро-

вок. Это может быть проблематично для пользователей, которые хотят использовать данный функционал в своих приложениях. Так как многие сервисы, работающие в данной области, используют различные устройства для мониторинга конкретных параметров и последующего построения на их основании различных программ занятий.

Вывод по первой главе

В целом, аналоги предоставляют схожий функционал, но имеют ряд недостатков. Отсутствует возможность независимо от специалиста строить индивидуальные программы тренировок. Это может привести к недостаточно эффективным тренировкам, которые не учитывают индивидуальные потребности и цели. Также отсутствует возможность интеграции в другие приложения. Аналогичные приложения обычно не интегрируются с другими приложениями и сервисами, что ограничивает их функциональность и удобство использования. Это может затруднить синхронизацию данных о тренировках с другими приложениями, такими как системы отслеживания питания или приложения мониторинга здоровья. Предлагаемое решение позволит реализовать автоматическое построение программ тренировок, индивидуальную адаптацию для каждого клиента, возможность использования в разных фитнес центрах и интеграцию с другими приложениями и сервисами. в разных фитнес центрах и интеграцию с другими сервисами.

2. СБОР ДАННЫХ

Для успешного создания модели искусственного интеллекта, которая сможет создавать программы тренировок, необходимо предварительно собрать множество обучающих данных, которые в себя включают:

- индивидуальные характеристики человека;
- цель программы тренировки;
- программу тренировки, составленная на основе цели программы тренировки и индивидуальных характеристик человека.

Для сбора индивидуальных характеристик человека будет использоваться оборудование, именуемое «Столб здоровья», который позволяет получить следующие характеристики: температура тела, рост, масса, индекс массы тела (ИМТ), артериальное давление, пульс, насыщение крови кислородом, сила сжатия кистей, процент содержания жира в организме.

Каждый участник, после сбора данных о его физических характеристиках, будет опрошен для выявления цели программы его тренировки.

Полученная информация будет передана специалисту для составления индивидуальной программы тренировки, что позволит создать обучающий набор данных, используемый для создания модели искусственного интеллекта.

2.1. Разработка приложения для сбора данных

2.1.1. Применение столба здоровья

Одним из источников данных, необходимых для выполнения поставленной задачи, служит устройство «Столб здоровья». Данное устройство представляет собой программно-аппаратный комплекс под управлением операционной системы Windows со следующим набором измерительных приборов: концентратор, ростометр, жироанализатор, термометр, пульсоксиметр, тонометр. На устройстве присутствует программное обеспечение, которое выполняет функцию управления и сбора данных с измерительных приборов. Фотография столба здоровья представлена на рисунке 1.

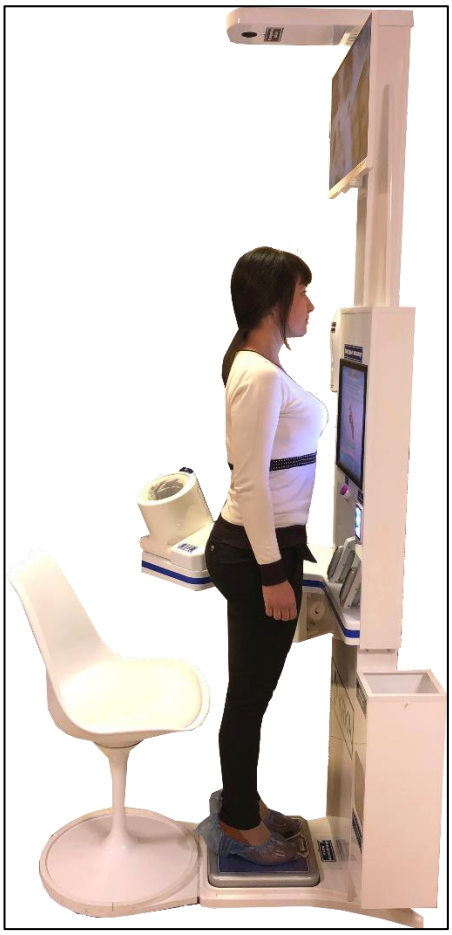


Рисунок 1 – Столб здоровья

Программное обеспечение столба здоровья представляет собой веб-сервис, у которого фронтэнд и бэкэнд расположены на одном устройстве. После того, как веб-сервер был запущен, пользователю открывается веб-интерфейс, в котором, он указывает свой пол и возраст, также пользователь может указать адрес электронной почты. Далее пользователь, следуя подсказкам, изображенным на экране, производит измерения показателей. По окончанию данной процедуры пользователю выводится результат в виде таблицы (рисунок 2), результат также отправляется по почте.

Рисунок 2 – Результаты измерений

В связи с тем, что программное обеспечение столба здоровья не предоставляет возможности сбора, хранения и передачи данных, а только выводит результаты работы на экран, было принято решение расширить функционал для получения этих возможностей. Однако, ввиду отсутствия исходных кодов встроить требуемый функционал не представляется возможным. Поэтому было принято решение разработать приложение, которое по средствам запросов к серверу будет получать необходимые данные. Однако в процессе тестирования было выяснено, что сервер сразу после отправки данных веб-интерфейсу удаляет их из памяти, и повторный запрос невозможен. В случае, когда разработанная программа перехватывала запрос, данные не отображались в веб-интерфейсе, что является неприемлемым для проверки корректности выдаваемых столбом здоровья результатов работы датчиков.

В связи с этим было принято решение разработать клиент-серверное приложение, которое будет создавать скриншоты экрана в момент отображения результатов работы на экран, распознавать данные на скриншотах и сохранять их на сервере. После распознавания данных все скриншоты удаляются для избегания дублирования данных. Серверная часть приложения представляет собой программу, которая будет собирать информацию от нескольких отправителей (столбов здоровья) и сохранять ее в csv файл. Также приложение будет предоставлять пользователю веб-интерфейс для просмотра, редактирования и скачивания имеющихся данных.

2.1.2. Клиентское приложение

Для реализации клиентского приложения использовался язык программирования высокого уровня Python [6], данный язык был выбран по причине распростароненности его в сфере веб-разработок и простоты написания программ, для распознавания текста использовался продукт компании Google Tesseract [7].

При разработке веб-сервиса применялись следующие библиотеки:

- pillow [8] – для операций над изображениями;
- pytesseract [9] – для распознавания текста;
- os [10] – для работы с файлами операционной системы;
- csv [11] – для сохранения данных в форате csv;
- threading [12] – для ускорения работы программы;
- time [13] – для создания временных меток.

Клиентское приложение поделено на три основных функции, которые последовательно выполняются друг за другом. Реализация программы основана на знании макета, на котором выводится таблица результатов.

Функция `threadingDef` выполняет предобработку изображения, представляющего собой скриншот сделанных программой и хранящий в названии уникальный идентификатор обследуемого. Она определяет цвет текста, который необходимо будет распознаваться, а также вырезает основные данные из изображения и передает их функции `image_recognition`. Программная реализация функции `threadingDef` представлена на рисунке 3.

Данная функция на вход получает список, в котором будет сохранятся результат работы функции, идентификатор изображения, список всех изображений, которые должны быть обработаны, а также список флагов, которые сигнализируют об успешной работе функции. Далее происходит загрузка изображения из внешней памяти и определение целевого цвета на изображении. В определенном месте изображения, по заранее заданным координатам происходит получение нужного цвета, который необходимо будет выделить в следующей функции. После этого происходит создание списка, в котором будут хранится вырезанные по координатам фрагменты исходного изображения, хранящие в себе информацию о конкретном измерении со столба здоровья, после чего происходит вызов функции `image_recognition`.

```
def threadingDef(list_rez, i, files_png,flag_list):
    im = Image.open(directory + "\\" + files_png[i])
    rgb = im.load()
    R = rgb[709, 292][0]
    G = rgb[709, 292][1]
    B = rgb[709, 292][2]

    list_im = []
    list_im.append(im.crop((250, 168, 340, 208))) # темп
    list_im.append(im.crop((364, 168, 452, 208))) # рост
    list_im.append(im.crop((458, 168, 554, 208))) # масса
    list_im.append(im.crop((561, 168, 659, 208))) # имп
    list_im.append(im.crop((665, 168, 760, 208))) # арт давление1
    list_im.append(im.crop((760, 168, 867, 208))) # арт давление2
    list_im.append(im.crop((872, 168, 969, 208))) # пульс
    list_im.append(im.crop((976, 168, 1074, 208))) # o2
    list_im.append(im.crop((1080, 168, 1126, 208))) # сила 1
    list_im.append(im.crop((1130, 168, 1180, 208))) # сила 2
    list_im.append(im.crop((1183, 168, 1269, 208))) # жир
    for j in list_im:
        list_rez[i].append(image_recognition(j, R, G, B))
    flag_list[i] = True
```

Рисунок 3 – Программная реализация функции `threadingDef`

Функция `image_recognition` закрашивает в белый цвет на полученном изображении всю лишнюю информацию, оставляя целевой текст, закрашивая его черным цветом. Данная операция необходима для повышения точности распознавания текста, так как по заявлениям разработчиков Tesseract более корректно распознает черный текст на белом фоне. После всех преобразований выполняется распознавание текста. Программная реализация функции представлена на рисунке 4.

```
def image_recognition(image, R, G, B):
    pixdata = image.load()
    for y in range(image.size[1]):
        for x in range(image.size[0]):
            if pixdata[x, y][0] == R and pixdata[x, y][1] == G and pixda-
ta[x, y][2] == B:
                pixdata[x, y] = (0, 0, 0, 255)
            else:
                pixdata[x, y] = (255, 255, 255, 255)
    return preprocessing((pytesseract.image_to_string(image, config='--
psm 10 --oem 1 -c tessedit_char_whitelist=0123456789.')))
```

Рисунок 4 – Программная реализация функции `image_recognition`

Данная функция на вход получает изображене, на котором необходимо распознать текст, а также три параметра, отвечающие за составляющие целевого цвета, который необходимо будет выделить на картинке. По-

сле того, как изображение было загружено, происходит операция по замене фона на белый цвет и замена целевого цвета на черный. Далее происходит вызов функции `preprocessing`.

Функция `preprocessing` преобразовывает текст, полученный в процессе распознавания, в числовой вид. Программная реализация третьей функции представлена на рисунке 5.

```
def preprocessing(data):
    rez = ""
    flag = True
    for i in data:
        if i.isdigit():
            rez += i
        elif i == '.' and flag:
            rez += i
            flag = False
        else:
            break
    if rez == '':
        rez = -1
    try:
        return float(rez)
    except ValueError:
        return -1.0
```

Рисунок 5 – Программная реализация функции `preprocessing`

Данная функция получает на вход результат работы Tesseract в виде строки символов, после чего создается переменная строкового типа, которая будет содержать прошедшее проверку число. Далее происходит проверка корректного распознавания числа, каждый символ распознаной строки проверятся на число, и в случаи если это число, то оно добавляется в rez, если это точка она добавляется к результату и флаг сигнализирующий о том, что точка найдена и меняет свое значение, что позволяет избежать повторного ее появления в результирующей строке. В случаи если символ не является числом цикл прекращается. После этого происходит проверка не пуста ли результирующая строка и попытка преобразования строки в число с плавающей точкой. Если переменная rez пуста, или попытка преобразовать число в строку заканчивается неудачей,

то возвращается число сигнализирующие о том, что распознование закончилось неудачно.

После выполнения трех функций полученный результат сохраняется в список `list_rez`, который в конце будет отправлен на сервер.

Связи с тем, что среднее время распознавания одного изображения равно три секунды, было принято решение распараллелить программу клиента посредствам нитей. Для каждого изображения, которое поступает на вход программе, создается отдельная нить, которая и занимается его обработкой и распознаванием.

2.1.3. Серверная часть приложения

Для реализации серверной части приложения использовались язык программирования Python, фреймворк Flask [14], язык разметки HTML и шаблонизатор Jinja [15].

Приложение представляет собой набор декораторов, которые регистрируют URL-адреса для следующих действий:

– `@app.route('/')` – для отображения главной страницы, на которой представлена таблица с собранными данными, также предоставляется возможность выбора строки таблицы для редактирования;

– `@app.route('/api', methods=['POST'])` – для сбора данных, поступающих с клиентских приложений;

– `@app.route('/editing/<uid>')` – для отображения страницы редактирования данных выбранной строки таблицы;

– `@app.route('/save', methods=['POST'])` – для сохранения результатов редактирования строки таблицы.

Реализация декораторов `@app.route('/')` и `@app.route('/api', methods=['POST'])` представлены на рисунках 6 и 7.

```
@app.route('/')
def index():
    global dict_rez
    with open('data.csv', 'r', newline='') as f:
        spamreader = csv.reader(f, delimiter=',', quotechar='|')
        for row in spamreader:
            dict_rez[row[0]] = list(map(float, row[1:]))
    return render_template('index.html', url_ip = url_ip, user_list =
list(dict_rez.items()), len_user = len(dict_rez))
```

Рисунок 6 – Реализация `@app.route('/')`

При вызове функции `index` происходит создание глобальной переменной, в которой будет хранится все данные загруженные из csv файла, далее в коде происходит выгрузка данных из файла и их добавление в переменную `dict_rez`, после чего происходит вызов функции рендера, которая выводит на экран пользователя HTML страницу со всеми собранными с клиента данными с возможностью перейти на отельную страницу редактирования данных.

```
@app.route('/api', methods=['POST'])
def api():
    global dict_rez
    data = request.get_json()['list_rez']
    for i in data:
        dict_rez[i[-1]] = i[0:-1]
        dict_rez[i[-1]].append(0.0)
        dict_rez[i[-1]].append(0.0)
        dict_rez[i[-1]].append(0.0)
        dict_rez[i[-1]].append(0.0)
    with open('data.csv', 'w', newline='') as f:
        writer = csv.writer(f)
        for i in dict_rez:
            writer.writerow([i]+dict_rez[i])
    return "ok"
```

Рисунок 7 – Реализация `@app.route('/api', methods=['POST'])`

При вызове функции `api` происходит инициализация глобальной переменной `dict_rez`, после чего происходит получение данных от клиента через POST запрос с последующим добавлением полученных данных в `dict_rez` и добавление дополнительных четырех полей, для возможного добавления информации. Далее происходит запись данных во внешний файл.

Разметка главной страницы (index.html) с применением шаблонизатора представлена на рисунке 8.

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8" />
  <title>Столб здоровья</title>
 </head>
 <body>
  <table border="1" cellpadding="10">
  <tr align="center"><td>Идентификатор</td>

  <td>Температура</td><td>Рост</td><td>Масса</td>
  <td>ИМТ</td><td>Арт. давление 1</td><td>Арт. давление 2</td>
  <td>Пульс</td><td>Насыщение крови</td><td>Сила кисти 1</td>
  <td>Сила кисти 2</td><td>Содерж. жир</td><td>Поле 1</td>
  <td>Поле 2</td><td>Поле 3</td><td>Поле 4</td></tr>

{% for i in range(len_user) %}
    <tr><td> <a href="{{url_ip}}/editing/{{ user_list[i][0] }}">{{ us-
er_list[i][0] }}</a> </td>

      {% for j in range(15) %}
      <td align="center"> {{ user_list[i][1][j] }}</td>
      {% endfor %}</tr>
{% endfor %}
</table>
 </body>
</html>
```

Рисунок 8 – Разметка главной страницы (index.html)

Главная страница представляет собой HTML страницу, в которой описана таблица для вывода пользователю информации о собранных данных. Также на данной странице применяется шаблонизатор Jinja, который в цикле создает необходимое количество строк в таблице, а также заполняет ее необходимыми данными. Таблица включает в себя уникальный идентификатор пользователя, оформленный в виде ссылки на страницу для редактирования данных, информацию о температуре, массе тела, индексе массы тела, артериальном давлении, пульсе, насыщенности крови кислородом, силе кистей и процентном содержании жира.

Фрагмент графического представления главной страницы представлен на рисунке 9.

Идент.	t°	Рост	Масса	ИМТ	Дав. 1	Дав. 2	Пульс	O2	Сы ки
ke2228vaa92	35.7	176.0	81.4	26.28	137.0	81.0	75.0	96.0	3
ke2222vaa92	35.7	176.0	81.4	26.28	137.0	81.0	75.0	96.0	3
ac2261ses33	36.4	174.0	70.5	23.29	120.0	78.0	73.0	99.0	4

Рисунок 9 – Фрагмент графического представления главной станицы

Тестирование клиент-серверного приложения

Функциональное тестирование – это вид тестирования, проверяющий функциональные возможности программы [16]. Набор тестов представлен в таблице 1.

Таблица 1 – Набор тестов

№	Название теста	Шаги	Ожидаемый результат
1	Распознавание текста	1. Запуск серверного приложения 2. Запуск клиентского приложения	Данные были успешно распознаны и отправлены на сервер для сохранения данных
2	Проверка на отсутствующие данные	1. Запуск серверного приложения 2. Запуск клиентского приложения	Данные были успешно распознаны и отправлены на сервер, данные, которые отсутствуют обозначены «-1»
3	Проверка на изображение	1. Запуск серверного приложения 2. Запуск клиентского приложения	Файлы, не являющиеся изображениями, не подвергались обработке
4	Изменение размеров экрана	1. Запуск серверного приложения 2. Запуск клиентского приложения	Данные были успешно распознаны и записаны в файл

Тесты 1–3 были пройдены успешно. Тест 4 не был пройден. Однако клиентское приложение было реализовано для столба здоровья. Монитор вмонтирован в данное устройство и использование клиентского приложения для других размеров экрана не предполагается.

2.2. Внешние данные

В связи с тем, что данных, полученных со столба здоровья недостаточно для получения обучающего набора данных, было принято решение собрать дополнительную информацию с помощью анкетирования:

- рост;
- масса тела;
- обхват запястья;
- измерение запястья;
- пол человека;
- возраст;
- уровень физической активности;
- периодичность употребления пищи.

Был создан электронный опрос, который позволил значительно дополнить набор данных, собранных ранее со столба здоровья. Данный опрос был размещен в электронном курсе «Физическая культура и спорт», и был доступен для прохождения большинству студентов проходящий занятия у преподавателей кафедры физического воспитания института спорта туризма и сервиса ЮУрГУ.

2.3. Обучающий набор данных

В результате проделанной работы по сбору обучающих данных, было получено два набора данных.

Первый набор содержит 87 строк, собранных со столба здоровья. Данный датасет был расширен до 120 строк синтетическим путем, при помощи CTGAN [17]. Данная модель представляет собой набор генераторов синтетических данных на основе глубокого обучения на входных данных, что позволяет достичь высокой достоверности результатов. Пример создания синтетических данных приведен на рисунке 10.

```
from ctgan import CTGAN
import pandas as pd
df = pd.read_csv('data.csv')
df.drop(df.columns [0], axis= 1 , inplace= True)
model = CTGAN()
model.fit(df)
synthetic_data = model.sample(150)
synthetic_data = synthetic_data.round(2)
synthetic_data.to_csv("data_synthetic.csv")
```

Рисунок 10 – Создание синтетических данных

Второй набор данных содержит 653 строки, собранных путем анкетирования, который также был расширен синтетическим путем до 900 строк, при помощи CTGAN.

Вывод по второй главе

В результате работы было разработано приложение для сбора данных со столба здоровья. Также был разработана анкета для сбора данных о характеристиках человеческого тела. По результатам работы по сбору данных было получено два обучающих набора данных, расширенных синтетическим путем до 120 и 900 строк соответственно. Дальнейшее увеличение обучающего набора данных синтетическим путем не представляется возможным, так как, если искусственных данных будет много больше реальных, то повышается риск того, что модель машинного обучения не сможет корректно обрабатывать поступающие ей данных, так как будет основываться на несуществующих зависимостях, созданных генератором данных.

3. ОБУЧЕНИЕ МОДЕЛЕЙ МАШИННОГО ОБУЧЕНИЯ

Для корректного создания программ тренировок для каждого пользователя, их необходимо классифицировать. В ходе изучения предметной области было выяснено, что люди делятся на три соматотипа [18–20] описывающие их физические характеристики: эктоморф, мезоморф и эндоморф. Произведем классификацию по этим типам.

Для реализации функции классификации пользователей по их индивидуальным параметрам, и выдачи подходящих им программ тренировок были выбраны следующие методы классификации, реализованные в библиотеке Scikit-learn: метод k ближайших соседей, метод дерева решений, метод случайного леса, метод опорных векторов, а также была разработана простая нейросетевая модель, основанная на библиотеке Keras использующая в своей реализации TensorFlow.

Метод k-ближайших соседей

Метод k-ближайших соседей [21] – это простой и эффективный алгоритм машинного обучения, который используется для классификации и регрессии. Он основан на принципе, что точки данных с похожими характеристиками имеют схожие целевые значения.

Алгоритм, имея набор данных, состоящий из объектов данных с известными значениями признаков и целевыми значениями. Вычисляет для каждой новой точки расстояние до всех остальных точек в тренировочном наборе данных, используя при этом меру расстояния (например, Евклидово расстояние). Далее из набора выбирается k точек данных с наименьшим расстоянием. Эти точки известны как k ближайших соседей. Далее для всех данных, которые не отнесены к какому-либо классу устанавливается наиболее распространенный класс среди k ближайших соседей.

Программный код обучения и результат проверки на корректность классификации приведен на рисунке 11. Точность классификации на данных, собранных путем анкетирования, составила 0,889. Точность на данных, собранных посредствам столба здоровья, составила 0,8.

```
[ ]  from sklearn.neighbors import KNeighborsClassifier
     n_neighbors = [2, 4, 8, 16, 24, 32, 64]
     score1 = []
     score2 = []
     for i in n_neighbors:
       KNeighbors = KNeighborsClassifier(n_neighbors=i)
       KNeighbors.fit(x1, y1)
       score1.append(KNeighbors.score(x2,y2))
     for i in n_neighbors:
       KNeighbors = KNeighborsClassifier(n_neighbors=i)
       KNeighbors.fit(x11, y11)
       score2.append(KNeighbors.score(x22,y22))

[ ]  print(max_depth[score1.index(max(score1))], max(score1))
     print(max_depth[score2.index(max(score2))], max(score2))

     4 0.88989898989899
     4 0.8
```

Рисунок 11 – Обучение метода k-ближайших соседей

Метод дерева решений

Метод дерева решений [22] – это алгоритм машинного обучения, который используется для решения задач классификации. Он представляет собой древовидную структуру, где каждый узел представляет собой проверку на признак, каждая ветвь представляет возможный результат проверки, а листья представляют классы классификации.

Алгоритм работает следующим образом. Начиная с корневого узла выбирается признак, который наилучшим образом разделяет данные на подмножества с разными целевыми значениями. Это делается с использованием метрики (например, коэффициент Джини). Данные разделяются на подмножества на основе выбранного признака. Каждое подмножество становится дочерним узлом корневого узла. Процесс выбора признака и разделения данных повторяется для каждого дочернего узла, пока не будут

достигнуты следующие условия остановки:

– все точки данных в узле принадлежат одному классу;

– нет больше признаков, которые можно использовать для разделения данных;

– достигнута максимальная глубина дерева.

Программный код обучения и результат проверки на корректность классификации приведен на рисунке 12. Точность классификации на данных, собранных путем анкетирования, составила 0,84. Точность на данных, собранных посредствам столба здоровья, составила 0,8.

```
[ ]  from sklearn import tree
     DecisionTree = tree.DecisionTreeClassifier()
     DecisionTree.fit(x1, y1)
     print(DecisionTree.score(x2,y2))
     DecisionTree = tree.DecisionTreeClassifier()
     DecisionTree.fit(x11, y11)
     print(DecisionTree.score(x22,y22))

     0.84
     0.8
```

Рисунок 12 – Обучение метода дерева решений

Метод случайного леса

Метод случайного леса [23] – это ансамблевый алгоритм машинного обучения, который объединяет множество деревьев решений для повышения точности и устойчивости.

Алгоритм работает следующим образом. Набор, данный случайным образом, делится на множества уникальных для каждого дерева решения под выборки. Далее дерево решений обучается на каждом из созданных наборов данных. При обучении используется случайное подмножество

признаков, что делает деревья более разнообразными и менее подверженными переобучению. После обучения всех деревьев предсказания объединяются для получения окончательного решения путем голосования.

Программный код обучения и результат проверки на корректность классификации приведен на рисунке 13. Точность классификации на данных, собранных путем анкетирования, составила 0,869. Точность на данных, собранных посредствам столба здоровья, составила 0,8.

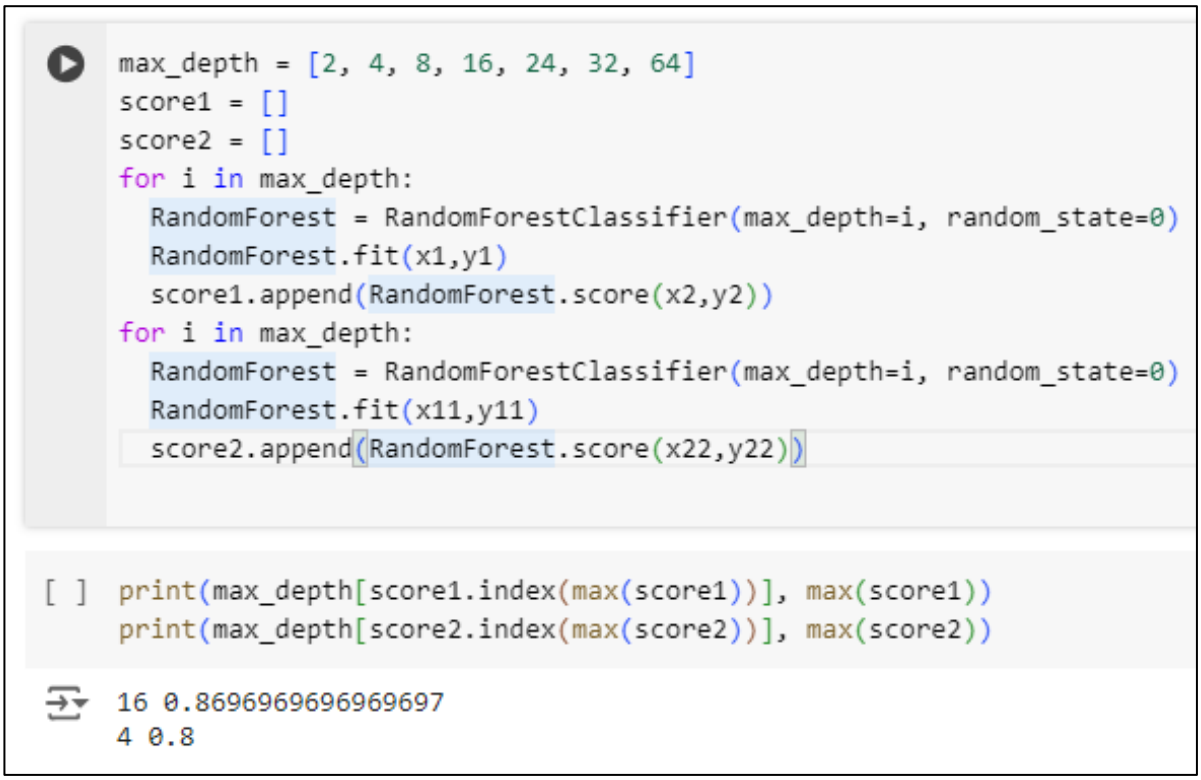


Рисунок 13 – Обучение метода случайного леса

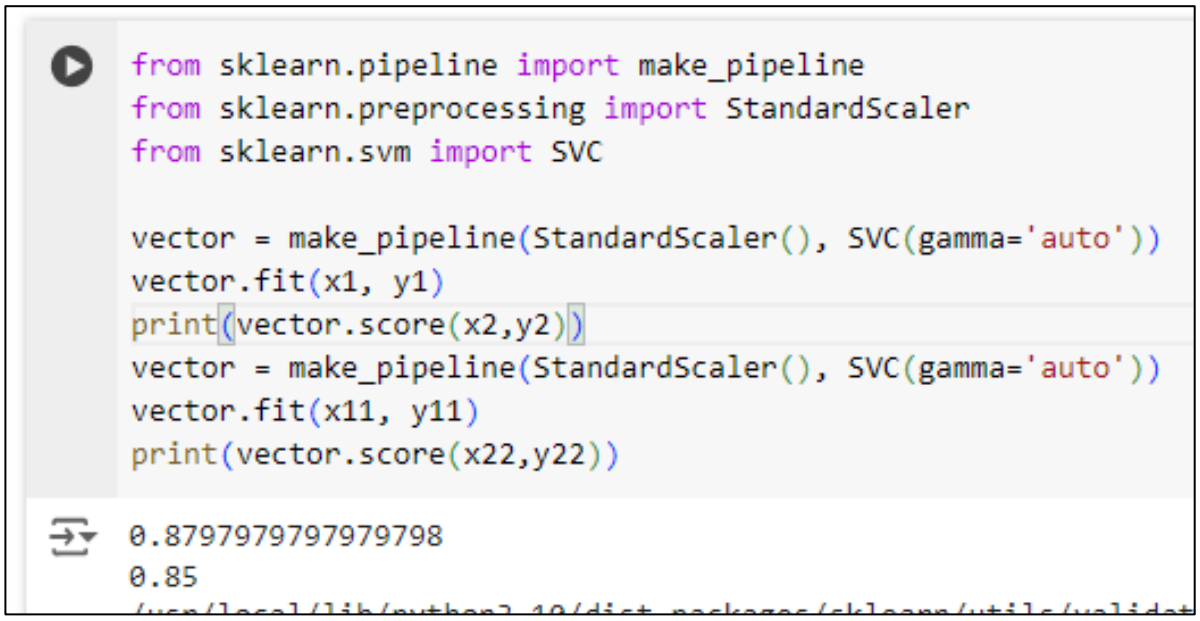
Метод опорных векторов

Метод опорных векторов [24] – это алгоритм машинного обучения, который стремится найти гиперплоскость, разделяющая точки данных разных классов с максимальным запасом.

Алгоритм работает следующим образом. Алгоритм ищет гиперплоскость, которая разделяет данные, максимизировав расстояние до ближайших точек данных, называемых опорными векторами. Далее применяется функция решения – это уравнение гиперплоскости, которое используется

для классификации новых данных. Точки данных с одной стороны гиперплоскости относятся к одному классу, а точки данных с другой стороны к другому классу.

Программный код обучения и результат проверки на корректность классификации приведен на рисунке 14. Точность классификации на данных, собранных путем анкетирования, составила 0,87. Точность на данных, собранных посредствам столба здоровья, составила 0,85.



```
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC

vector = make_pipeline(StandardScaler(), SVC(gamma='auto'))
vector.fit(x1, y1)
print(vector.score(x2,y2))
vector = make_pipeline(StandardScaler(), SVC(gamma='auto'))
vector.fit(x11, y11)
print(vector.score(x22,y22))
```

```
0.8797979797979798
0.85
```

Рисунок 14 – Обучения метода опорных векторов

Нейросетевая модель

Нейросетевая модель [25] – это алгоритм машинного обучения, вдохновленный работой человеческого мозга. Он состоят из слоев взаимосвязанных узлов, называемых нейронами. Нейроны обрабатывают входные данные и передают выходные данные на следующие слои, постепенно извлекая более абстрактные и сложные представления данных. Разработанные архитектуры нейросетевых моделей представлены в приложении.

Первой модели на вход поступает 7 параметров, собранных путем анкетирования, а второй модели 11 параметров, собранных с применением столба здоровья. В первой сети, не считая входного и выходного слоя, присутствует три полносвязных слоя на 16, 32 и 64 нейрона с функцией

активации relu. Во второй сети, не считая входного и выходного слоя, присутствует четыре полносвязных слоя на 16, 32, 64 и 128 нейронов с функцией активации relu. Выходные слои имеют 3 нейрона, которые отвечают за решение модели касательно класса телосложения человека. Функция активации выходных слоев softmax возвращает вероятность принадлежности входных данных к каждому классу. Оптимизаторами сетей был выбран adam, а метрика ошибки accuracy.

Обучение первой модели составило 30 эпох, точность предсказания классов на данных, собранных путем анкетирования, составила 0,89, обучение второй модели составило 30 эпох, точность предсказания на данных, собранных по средствам столба здоровья, составила 0,91.

Подбор параметров нейронной сети представлен в таблице 2.

Таблица 2 – Подбор параметров нейронной сети

Количество скрытых слоев	**Количество эпох обучения**	**Анкетирование**	**Столб здоровья**
2	30	0,8462	0,7800
3	10	0,8313	0,7700
3	20	0,8475	0,8000
3	30	0,8900	0,8500
4	10	0,8338	0,6600
4	20	0,8562	0,8500
4	30	0,8500	0,9100

Вывод по третьей главе

В результате работы по обучению моделей машинного обучения были получены пять моделей, которые могут с высоком процентом вероятности корректно классифицировать человека. Но связи с тем, что точность всех моделей в районе 80% принято решение использовать в классификации данных ансамбль всех реализованных моделей.

4. ПРОЕКТИРОВАНИЕ ВЕБ-СЕРВИСА

4.1. Диаграмма вариантов использования

Разрабатываемый веб-сервис должен удовлетворять следующим функциональным требованиям.

1. Веб-сервис должен предоставлять веб-интерфейс для прямого взаимодействия пользователя с системой.
2. Веб-сервис должен предоставлять открытый API интерфейс для возможной интеграции сервиса в другие приложения.
3. Веб-сервис должен предоставлять интерфейс для ввода минимально необходимых индивидуальных параметров человека.
4. Веб-сервис должен создавать программу тренировок, на основании индивидуальных параметров человека.
5. Веб-сервис должен через некоторое время запросить у пользователя информацию о результатах программы тренировок, его новые физические показатели, а также его отзыв о программе.

Разрабатываемый веб-сервис должен удовлетворять следующим нефункциональным требованиям.

1. Веб-сервис должен быть написан на языке программирования высокого уровня python с использованием фреймворка Flask.
2. Веб-сервис должен быть доступен из любой географически удаленной точки.
3. Веб-сервис должен использовать СУБД SQLite, для хранения параметров человеческого тела.
4. Интерфейс веб-сервиса должен быть доступен из браузера.

На основании приведенных выше требований была составлена диаграмма вариантов использования, представленная на рисунке 15.

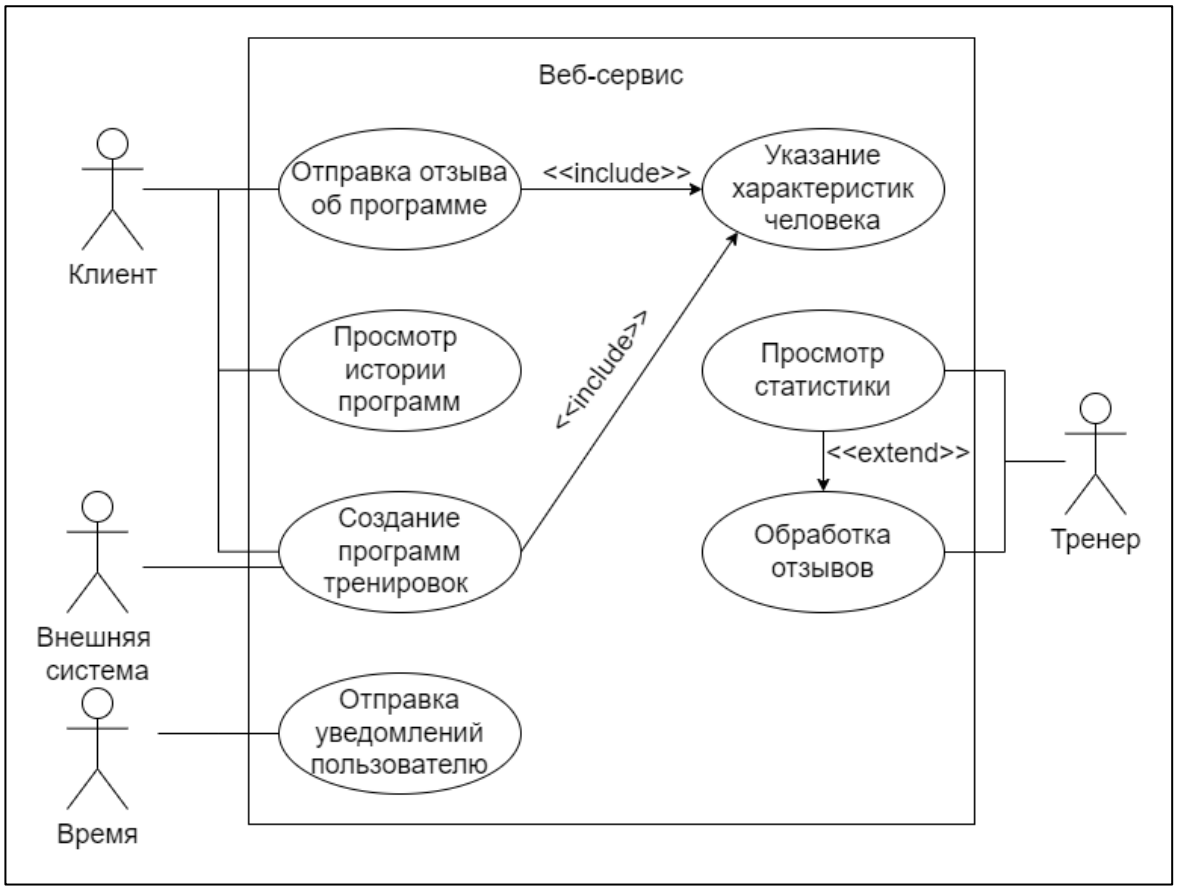


Рисунок 15 – Диаграмма вариантов использования

На диаграмме вариантов использования представлены 4 актеров.

1. Клиент – данный актер отражает сущность людей, которые будут пользоваться данным веб-сервисом.

2. Внешняя система – данный актер отражает сущность систем, в которые будет интегрирован данный веб-сервис по средствам открытого API.

3. Время – данный актер отражает время, прошедшее с момента создания программ тренировок для каждого пользователя.

4. Тренер – данный актер отражает специалиста, который может оценить качество созданных для пользователя программ тренировок.

Клиенту доступен следующий функционал.

1. Создание программ тренировок – создание программы тренировок пользователя.

2. Отправка отзыва о программе – создание и отправка отзыва о созданной ранее для пользователя программе тренировок.

3. Просмотр истории программ – просмотр истории созданных ранее для пользователя программ тренировок.

4. Указание характеристик человека – возможность пользователю указать запрашиваемые характеристики тела.

Внешней системе доступен следующий функционал:

1) создание программ тренировок – создание программы тренировок для внешней системы;

2) указание характеристик человека – возможность внешней системы указать данные пользователя, для которого будет создана программа.

Тренеру доступен следующий функционал:

1) обработка отзывов – тренер на основании отзывов пользователя о программе тренировок производит оценку корректности сформированной для него программы тренировок, а также при необходимости производит ее модификацию;

2) просмотр статистики – возможность тренера посмотреть статистику конкретного пользователя.

У актера время есть возможность отправки уведомлений пользователю, в котором сообщается о необходимости оставить отзыв на программу тренировка спустя некоторое время после ее создания, данное ограничение необходимо для повышения объективности отзывов по программам.

4.2. Проектирование базы данных

После произведенного анализа данных была спроектированная схема базы данных, представленная на рисунке 16. В данной схеме присутствует шесть таблиц: таблица для хранения данных пользователя (Users), таблица для хранения программ тренировок пользователя (Training_programs), таблица для хранения набора данных собираемого при помощи столба здоровья (Data_PH), таблица для хранения данных, собираемых из опроса

(Data_form), таблица для хранения отзывов (Reviews), таблица для хранения всех возможный программ тренировок программ (Programs).

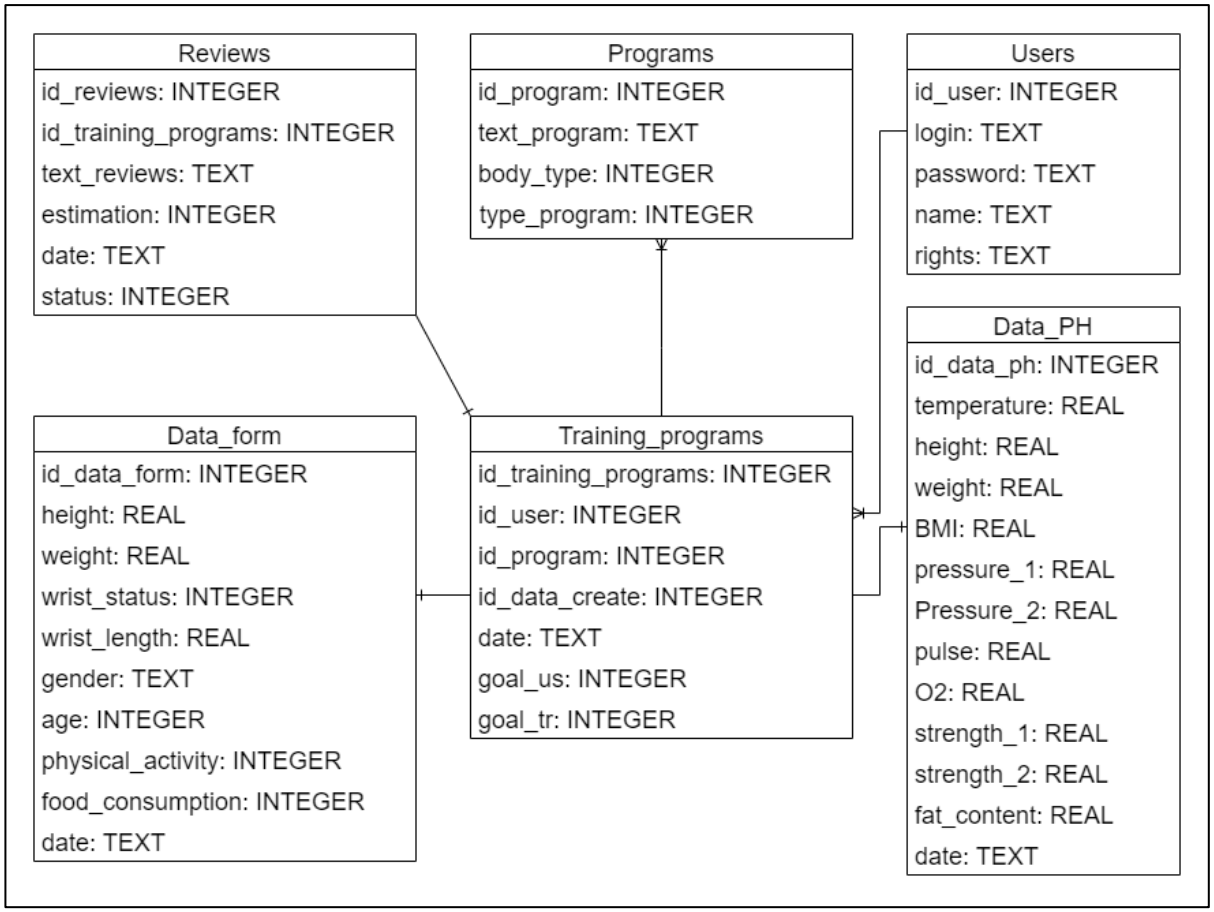


Рисунок 16 – Схема базы данных

Таблица Users содержит в себе следующие данные:

1) уникальный идентификатор пользователя (id_user), подставляющий собой число типа INTEGER;

2) логин (login) необходимый для авторизации в сервисе, представляющий собой строку типа TEXT и являющийся электронной почтой пользователя;

3) пароль (password) необходимый для авторизации в сервисе, представляющий собой строку типа TEXT хранящийся в виде хеша;

4) имя (name) пользователя, которое он задает при регистрации, представляющее собой строку типа TEXT;

5) право доступа (rights) к сервису, представляет собой строку типа TEXT и может хранить следующие права: admin, expert, user.

Таблица Training_programs содержит в себе следующие данные:

1) уникальный идентификатор программы тренировок (id_training_programs) пользователя, представляющий собой число типа INTEGER;

2) уникальный идентификатор пользователя (id_user), представляющий собой число типа INTEGER;

3) уникальный идентификатор программы (id_program), представляющий собой число типа INTEGER;

4) уникальный идентификатор данных физического состояния пользователя (id_data_create) при создании программы, представляющий собой число типа INTEGER;

5) уникальный идентификатор данных физического состояния пользователя (id_data_reviews), на момент создания отзыва, представляющий собой число типа INTEGER;

6) дата создания программы тренировок (date), представляющий собой строку типа TEXT;

7) цель программы тренировок (goal_us), установленной пользователем, представляющий собой число типа INTEGER;

8) цель программы тренировок (goal_tr), установленной тренером на основании отзыва пользователя, представляющий собой число типа INTEGER.

Таблица Data_PH содержит в себе следующие данные:

1) уникальный идентификатор данных пользователя (id_data_ph), собранных со столба здоровья, представляющий собой число типа INTEGER;

2) температура тела (temperature), представляет собой число типа REAL;

3) рост (height), представляет собой число типа REAL;

4) масса тела (weight), представляет собой число типа REAL;

5) индекс массы тела (BMI), представляет собой число типа REAL;

6) давление (1) (pressure_1), представляет собой число типа REAL;

7) давление (2) (Pressure_2), представляет собой число типа REAL;

8) пульс (pulse), представляет собой число типа REAL;

9) насыщение крови кислородом (O2), является числом типа REAL;

10) сила сжатия левой кисти рук (strength_1), представляет собой число типа REAL;

11) сила сжатия правой кисти рук (strength_2), представляет собой число типа REAL;

12) процент содержания жира (fat_content), представляет собой число типа REAL;

13) дата внесения данных (date), является строкой типа TEXT.

Таблица Data_form содержит в себе следующие данные:

1) уникальный идентификатор данных пользователя (id_data_form), собранных путем анкетирования, представляющий собой число типа INTEGER;

2) рост (height), представляющий собой число типа REAL;

3) масса тела (weight), представляющий собой число типа REAL;

4) обхват запястья (wrist_status), представляющий собой число типа INTEGER;

5) длина запястья (wrist_length), представляющий собой число типа REAL;

6) пол пользователя (gender), представляющий собой строку типа TEXT;

7) возраст (age), представляющий собой число типа INTEGER;

8) уровень физической активности (physical_activity), является числом типа INTEGER;

9) периодичность употребления пищи пользователем (food_consumption), представляющий собой число типа INTEGER;

10) дата внесения данных (date), является строкой типа TEXT.

Таблица Reviews содержит в себе следующие данные:

1) уникальный идентификатор отзыва (id_reviews), представляет собой число типа INTEGER;

2) уникальный идентификатор программы тренировок (id_training_programs), представляет собой число типа INTEGER;

3) текст отзыва (text_reviews), оставленного пользователем, представляет строку типа TEXT;

4) оценка (estimation), оставленная пользователем на программу тренировок, представляет собой число типа INTEGER от 1 до 10;

5) дата внесения данных (date), является строкой типа TEXT;

6) статус (status), означающий этап обработки отзыва, является числом типа INTEGER.

Таблица Programs содержит в себе следующие данные:

1) уникальный идентификатор программы (id_program), представляет собой число типа INTEGER;

2) текст программы (text_program), представляет собой строку типа TEXT;

3) тип телосложения пользователя (body_type), представляет собой число типа INTEGER;

4) тип программы тренировок (type_program), представляет собой число типа INTEGER.

4.3. Архитектура веб-сервиса

При проектировании приложения были выделены три основных компонента: устройство клиента, внешняя система и веб-сервер.

Устройство клиента представляет собой устройство, которое имеет стабильное подключение к сети Интернет, а также поддерживает актуальные версии популярных браузеров. Данные ограничения накладываются из необходимости клиента иметь постоянную связь с сервером для обмена сообщениями, а также для корректного отображения веб-интерфейса.

Внешняя система представляет собой какое-либо программное обеспечение, которое может по средствам открытого API работать с сервером.

Веб-сервер представляет собой географически удаленный от устройства клиента и внешней системы сервер. Сервер содержит почтовый сервис, отвечающий за отправку писем с просьбой составить отзыв. Веб-приложение представляющие собой центральный узел системы, он отвечает за работу веб-интерфейса, создания тренировок, а также за сбор отзывов. Также на сервере присутствует база данных, которая хранит в себе все необходимые сведения для работы сервиса.

Вывод по четвертой главе

В данной главе были описаны пять функциональных требований, описывающих возможности веб-сервиса. Также были описаны четыре нефункциональных требования, которые описывают необходимые средства для реализации веб-сервиса. Было проведено проектирование базы данных. По результатам проектирования было получено шесть таблиц, описывающие данных о пользователе, данные о программах тренировок, данные об индивидуальных характеристиках человека, а также данные об отзывах и текста программ тренировок. Также была описана архитектура веб-сервиса, включающая в себя клиента, внешнюю систему и сервер, на котором располагается веб-приложение, база данных и почтовый сервис.

5. РЕАЛИЗАЦИЯ ВЕБ-СЕРВИСА

5.1. Средства реализации

При реализации веб-сервиса использовался язык высокого уровня Python версии 3.12.0.

Для написания кода применялась IDE PyCharm [26], версии 2024.1.1, разработанная компанией JetBrains. Данная IDE была выбрана по причине предоставления большого функционала для работы с сервисами и программными компонентами.

Для реализации серверной части приложения используется микро фреймворк Flask версии 3.0.0 с расширением Flask-Login версии 0.6.3.

Для реализации базы данных применяется СУБД SQLite3 [27].

5.2. Реализация базы данных

Реализованная база данных придерживается модели, изображенной на рисунке 16.

Код создания таблицы Data_PH приведен на рисунке 17.

```
CREATE TABLE "Data_PH" (
	"id_data_ph"	INTEGER NOT NULL UNIQUE,
	"temperature"	REAL NOT NULL,
	"height"	REAL NOT NULL,
	"weight"	REAL NOT NULL,
	"BMI" REAL NOT NULL,
	"pressure_1"	REAL NOT NULL,
	"Pressure_2"	REAL NOT NULL,
	"pulse"	REAL NOT NULL,
	"O2"  REAL NOT NULL,
	"strength_1"	REAL NOT NULL,
	"strength_2"	REAL NOT NULL,
	"fat_content"	REAL NOT NULL,
	"date"	TEXT NOT NULL,
	PRIMARY KEY("id_data_ph" AUTOINCREMENT)
);
```

Рисунок 17 – Реализация таблицы Data_PH

Данная таблица содержит в себе 13 столбцов 11 из которых описывают данные, собранные со столба. Все столбцы не могут хранить пустое значение. Столбец `id_data_ph` обозначающий идентификатор строчки таблицы является ключевым, уникальным и автоинкрементным.

Код создания таблицы Data_form приведен на рисунке 18.

```
CREATE TABLE "Data_form" (
	"id_data_form"	INTEGER NOT NULL UNIQUE,
	"height"	REAL NOT NULL,
	"weight"	REAL NOT NULL,
	"wrist_status"	INTEGER NOT NULL,
	"wrist_length"	REAL NOT NULL,
	"gender"	TEXT NOT NULL,
	"age" INTEGER NOT NULL,
	"physical_activity"	INTEGER NOT NULL,
	"food_consumption"	INTEGER NOT NULL,
	"date"	TEXT NOT NULL,
	PRIMARY KEY("id_data_form" AUTOINCREMENT)
);
```

Рисунок 18 – Реализация таблицы Data_form

Данная таблица содержит в себе 10 столбцов 8 из которых описывают данные, собранные путем анкетирования. Все столбцы не могут хранить пустое значение. Столбец `id_data_form` обозначающий идентификатор значения таблицы является ключевым, уникальным и автоинкрементным.

Код создания таблицы Programs приведен на рисунке 19.

```
CREATE TABLE "Programs" (
	"id_program"	INTEGER NOT NULL UNIQUE,
	"text_program"	TEXT NOT NULL,
	"body_type" INTEGER NOT NULL,
	"type_program"	INTEGER NOT NULL,
	PRIMARY KEY("id_program" AUTOINCREMENT)
);
```

Рисунок 19 – Реализация таблицы Programs

Данная таблица содержит в себе 4 столбца, каждый из них не может хранить пустое значение. Столбец `id_program` обозначающий идентификатор значения таблицы является ключевым, уникальным и автоинкрементным.

Код создания таблицы Reviews приведен на рисунке 20.

```
CREATE TABLE "Reviews" (
	"id_reviews"	INTEGER NOT NULL UNIQUE,
	"id_training_programs"	INTEGER NOT NULL,
	"text_reviews"	TEXT NOT NULL,
	"estimation"	INTEGER NOT NULL,
	"date"	TEXT NOT NULL,
	"status"	INTEGER NOT NULL,
	PRIMARY KEY("id_reviews" AUTOINCREMENT)
);
```

Рисунок 20 – Реализация таблицы Reviews

Данная таблица содержит в себе 6 столбцов, 2 из которых описывают данные отзыва пользователя. Все столбцы не могут хранить пустое значение. Столбец `id_reviews` обозначающий идентификатор значения таблицы является ключевым, уникальным и автоинкрементным.

Код создания таблицы Training_programs приведен на рисунке 21.

```
CREATE TABLE "Training_programs" (
	"id_training_programs"	INTEGER NOT NULL UNIQUE,
	"id_user"	INTEGER NOT NULL,
	"id_program"	INTEGER NOT NULL,
	"id_data_create"	INTEGER NOT NULL,
	"id_data_reviews"	INTEGER NOT NULL,
	"date"	TEXT NOT NULL,
	"goal_us"	INTEGER NOT NULL,
	"goal_tr"	INTEGER NOT NULL,
	PRIMARY KEY("id_training_programs" AUTOINCREMENT)
);
```

Рисунок 21 – Реализация таблицы Training_programs

Данная таблица содержит в себе 8 столбцов, 3 из которых описывают программу тренировок. Все столбцы не могут хранить пустое значение. Столбец id_training_programs обозначающий идентификатор значения таблицы является ключевым, уникальным и автоинкрементным.

Код создания таблицы Users приведен на рисунке 22.

```
CREATE TABLE "Users" (
	"id_user"	INTEGER NOT NULL UNIQUE,
	"login"	TEXT NOT NULL UNIQUE,
	"password"	TEXT NOT NULL,
	"name"	TEXT NOT NULL,
	"rights"	TEXT NOT NULL,
	PRIMARY KEY("id_user" AUTOINCREMENT)
);
```

Рисунок 22 – Реализация таблицы Users

Данная таблица содержит в себе 5 столбцов, 4 из которых хранят данные о пользователе. Все столбцы не могут хранить пустое значение. Столбец `id_user` обозначающий идентификатор значения таблицы является ключевым, уникальным и автоинкрементным.

5.3. Реализация функций веб-сервиса

Пользователю перед началом работы необходимо пройти авторизацию, для этого пользователю предоставляется возможность создать новый аккаунт или войти с уже имеющегося.

После того как пользователь авторизовался в зависимости от его прав доступа, ему выводится различные начальные страницы. Пользователь с правами user попадает в свой профиль, где он может видеть данные о себе и имеет возможность создать программу тренировок на основании данных, собранных заранее со столба здоровья, или же пройти анкетирование. Интерфейс профиля представлен на рисунке 23.

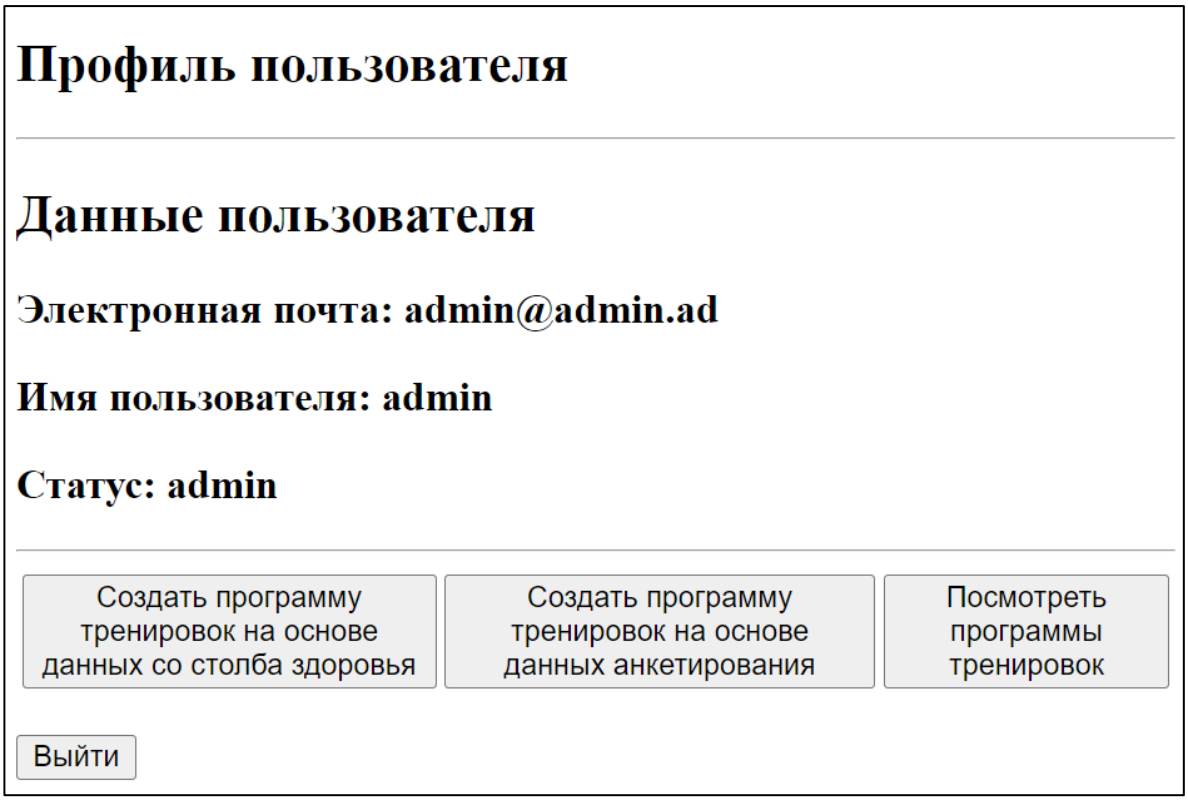


Рисунок 23 – Профиль пользователя

В зависимости от того, какую форму представления данных выбрал пользователь, ему будут представлены соответствующие форма для ввода данных, где после указания всех необходимых параметров он может нажать на кнопку создания программы, после чего форма будет отправлена на сервер, а пользователь будет перемещен обратно в профиль.

Как только форма будет доставлена на сервер, сервис зафиксирует время получения данных и внесет данные пользователя в соответствующую таблицу. После чего данные будут переданы в модуль с моделями машинного обучения.

Данный модуль представляет собой набор реализованных и описанных ранее методов машинного обучения, конфигурации которых были сохранены во внешней памяти. В момент поступления данных пользователя происходит классификация пользователя всеми доступными методами. После этого происходит голосование, где вариант, набравший большинство голосов побеждает. В случае одинаковых голосов, выбирается вариант за который проголосовали модели с большей точностью на этапе обучения. Реализация функции классификации представлена на рисунке 24.

```
def classification_form(data_form):
    flag = {0: 0, 1: 0, 2: 0}
    data = [[data_form.wrist_status, data_form.gender, da-
ta_form.physical_activity, data_form.food_consumption, data_form.weight,
data_form.age, data_form.height]]
    DecisionTree = load('model/classifier_DecisionTree_form.joblib')
    KNeighbors = load('model/classifier_KNeighbors_form.joblib')
    RandomForest = load('model/classifier_RandomForest_form.joblib')
    Vector = load('model/classifier_Vector_form.joblib')
    model = keras.models.load_model('model/model_form.h5')

    flag[DecisionTree.predict(data)[0]] += 1
    flag[KNeighbors.predict(data)[0]] += 1
    flag[RandomForest.predict(data)[0]] += 1
    flag[Vector.predict(data)[0]] += 1
    flag[model.predict(data)[0]] += 1

    flag = check(flag)

    return [key for key, value in flag.items() if value ==
max(flag.values())][0]
```

Рисунок 24 – Реализация функции классификации

После того, как было установлено к какому классу принадлежит пользователь и на основании цели тренировки, указанной пользователем, присваивается подходящая ему программа тренировки. Далее все данные записываются в базу данных для хранения. После этого пользователю становится доступна программа тренировок для просмотра. Реализация сохранения данных представлена на рисунке 25.

```
@app.route('/form_ph/save', methods=["POST"])
@login_required
def save_ph():
    cur = get_db()

    data_ph = Data_PH(temperature=float(request.form.get("temperature")),
                      height=float(request.form.get("height")),
                      weight=float(request.form.get("weight")),
BMI=float(request.form.get("BMI")),
                      pressure_1=float(request.form.get("pressure_1")),
                      pressure_2=float(request.form.get("pressure_2")),
pulse=float(request.form.get("pulse")),
                      O2=float(request.form.get("O2")),
                      strength_1=float(request.form.get("strength_1")),
                      strength_2=float(request.form.get("strength_2")),
                      fat_content=float(request.form.get("fat_content")),
                      date=datetime.today())

    data_ph.to_database(cur)
    body_type = AI.classification_ph(data_ph)
    program = Program.from_database(cur=cur, body_type=body_type,

type_program=TYPE_PROGRAM[request.form.get("type_program")])
    training_program = Train-
ing_program(id_user=current_user.user.id_user,
id_program=program.id_program,

id_data_create=data_ph.id_data_ph, id_data_reviews=-1,
                                        date=datetime.today(),
goal_us=TYPE_PROGRAM[request.form.get("type_program")])
    training_program.to_database(cur)
    return redirect(url_for('profile'))
```

Рисунок 25 – Реализация сохранения данных программы тренировок

После того как была создана программа тренировок пользователь может с ней ознакомится, нажав в профиле кнопку «Просмотреть программы тренировок». Выполнив данное действие, пользователь попадает на страницу, где ему доступны для просмотра все созданные для него программы тренировок. Пользователь может посмотреть цель созданной программы, дату создания программы, а также ознакомится с программой на

отдельной странице. Через определенный промежуток времени пользователю становится доступной возможность оставить отзыв на программу тренировки. Интерфейс страницы представлен на рисунке 26.

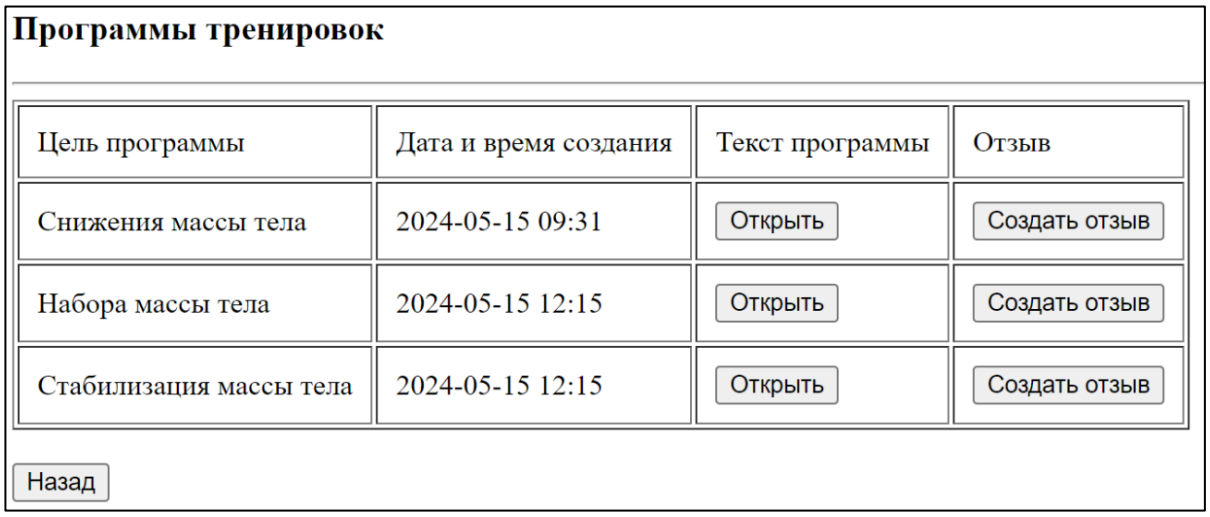
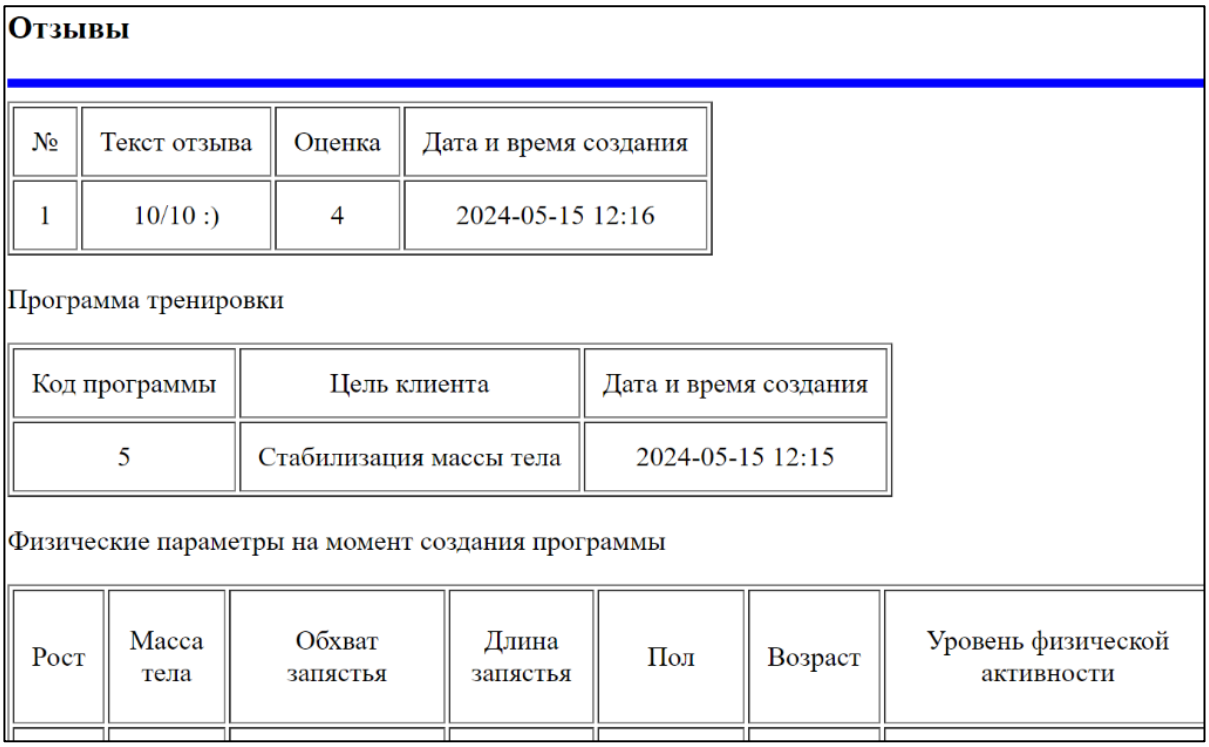


Рисунок 26 – Интерфейс просмотра созданных программ тренировок

Чтобы оставить отзыв пользователю необходимо выполнить три действия. На первом этапе, пользователь указывает на сколько балов из 10 он оценивает данную программу, после чего указывает текстовый отзыв (не более 1000 символов), после чего может перейти ко второму этапу. Второй этап представляет сбой выбор того, как пользователь хочет сохранить данные о текущем физическом состоянии, он может это сделать посредствам вновь собранных данных со столба здоровья или снова пройти анкетирование после на третьем этапе ему предоставляется выбранная им форма для ввода данных, после заполнения которой данные отправляются на сервер и сохраняются в базу данных.

После того, как в базе данных появился отзыв, он может быть обработан тренером или экспертом для корректного использования данных пользователя при дообучении имеющихся моделей. Для этого предусмотрена специальная права доступа «expert». Тренер, имеющий данные права при авторизации, попадает на специализированную страницу, где отображаются все отзывы, которые необходимо обработать. Фрагмент интерфейса представлен на рисунке 27.



Отзывы

№	Текст отзыва	Оценка	Дата и время создания
1	10/10 :)	4	2024-05-15 12:16

Программа тренировки

Код программы	Цель клиента	Дата и время создания
5	Стабилизация массы тела	2024-05-15 12:15

Физические параметры на момент создания программы

Рост	Масса тела	Обхват запястья	Длина запястья	Пол	Возраст	Уровень физической активности

Рисунок 27 – Фрагмент интерфейса эксперта

На рисунке можно увидеть сам отзыв клиента, а также информацию о созданной для пользователя программы тренировок. Кроме этого, на данной странице представлена информация о характеристиках пользователя на момент создания, а также информация о характеристиках на момент составления отзыва. Если эксперту недостаточно данных на странице, он может перейти к статистике. Там эксперт может увидеть: информацию о всех программах тренировок, которые были созданы пользователю, а также его физические показатели для каждой программы.

После того, как эксперт принял решение о том правильно ли был установлен тип телосложения пользователя и принесла ли программа желаемый эффект, он заполняет форму и данные об оценке эксперта сохраняются в базу данных. После чего данные пользователя могут применятся для дообучения моделей

В случае, если неавторизованный пользователь постарается получить доступ к какой-либо странице сервиса, то он будет перенаправлен на страницу авторизации. В случае, когда авторизованный пользователь попыта-

ется получить данные к страницам, к которым у него нет доступа, он будет перенаправлен на страницу сообщающей об этом, с возможностью вернутся на главную страницу для своей роли. Страница интерфейса представлена на рисунке 28.

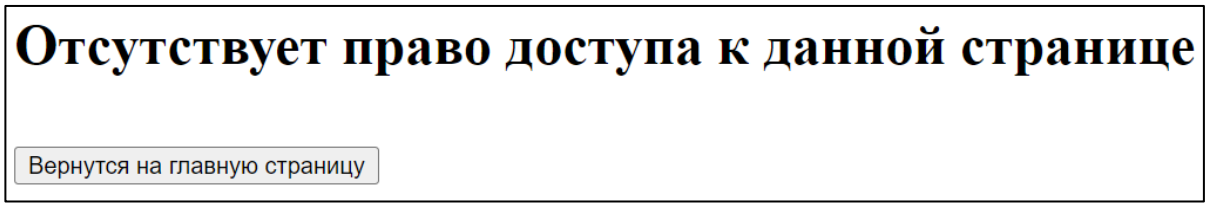


Рисунок 28 – Интерфейс запрета доступа к странице

Реализация API

Для реализации функции взаимодействия с внешними системами было реализовано открытое API, которое по GET запросу возвращает шаблон структуры JSON-документа. При POST запросе отправленного внешней системой производило классификации телосложения по данным и возвращала JSON-ответ, содержащий рекомендуемую программу тренировок. В случае отправки некорректных данных, внешней системе будет возвращено соответствующее сообщение.

Данные, полученные от внешней системы, не сохраняются.

Реализация API представлена на рисунке 29.

```
@app.route('/api', methods=["GET", "POST"])
def api_ph():
    cur = get_db()
    if request.method == 'GET':
        return jsonify(Json_out)  # Отправка json структуры запроса
    elif request.method == 'POST':
        form_data = request.get_json()
        answer = processing_api(cur, form_data)  # Обработка полученных
данных и формирование ответа
        return jsonify(answer)
```

Рисунок 29 – Реализация API

Вывод по пятой главе

В данной главе описан процесс реализации базы данных и всех функций веб-сервиса.

6. ТЕСТИРОВАНИЕ ВЕБ-СЕРВИСА

6.1. Функциональное тестирование

Функциональное тестирование – это вид тестирования, проверяющий функциональные возможности программы, те функции, которые были реализованы согласно постановке задачи. Набор тестов представлены в таблице 3. Все тесты были пройдены успешно.

Таблица 3 – Набор функциональных тестов

№	Название теста	Шаги	Ожидаемый результат	Тест Пройден?
1	Регистрация	1. Перейти на главную страницу сервиса 2. Нажать на кнопку регистрация 3. Указать данные для регистрации 4. Нажать на зарегистрироваться	Пользователю создан аккаунт, данные пользователя записаны в базу данных	ДА
2	Авторизация	1. Перейти на главную страницу сервиса 2. Нажать на кнопку авторизация 3. Указать данные для авторизации 4. Нажать на кнопку входа	Пользователь авторизирован в веб-сервисе, а также перенаправлен на домашнюю страницу пользователя	Да
3	Создание программы тренировок	1. Перейти в профиль пользователя 2. Выбрать способ задания данных пользователя 3. Указать требуемые формой данных 4. Нажать на кнопку создать программу	Пользователю создана программа тренировок, данные программы о программе сохранены в базу данных	Да
4	Просмотр программы тренировки	1. Перейти в профиль пользователя 2. Перейти на страницу с программами тренировок. 3. Выбрать из списка созданные программ требуемую 4. Нажать на кнопку открыть	Пользователь переходит на страницу, где представлен полный текст тренировки	Да
5	Создание отзыва о программе тренировки	1. Перейти в профиль пользователя 2. Перейти на страницу с программами тренировок. 3. Выбрать из списка созданные программ требуемую 4. Нажать на кнопку создать отзыв 5. Прохождение трех этапов составления отзыва 6. Нажатие кнопки отправить	Отзыв пользователя сохраняется в базу данных. Смежная с отзывом таблица обновляется.	Да

Окончание таблицы 2

№	Название теста	Шаги	Ожидаемый результат	Тест Пройден?
6	Обработка отзыва пользователя	1. Авторизироваться с правами доступа эксперт 2. Перейти на домашнюю страницу эксперта 3. Анализ данных пользователя и отзыва 4. Установка типа программы 5. Отправка формы	Данные об отзыве и смежной с ним таблице изменены. Данные доступны для дообучения	Да

6.2. Тестирование API

Целью тестирования API была проверка корректности обработки запросов внешней системы.

Первый тест включал в себя проверку корректного представления шаблона структуры JSON-документа предоставляемой по GET запросу. Результат теста приведен на рисунке 30.

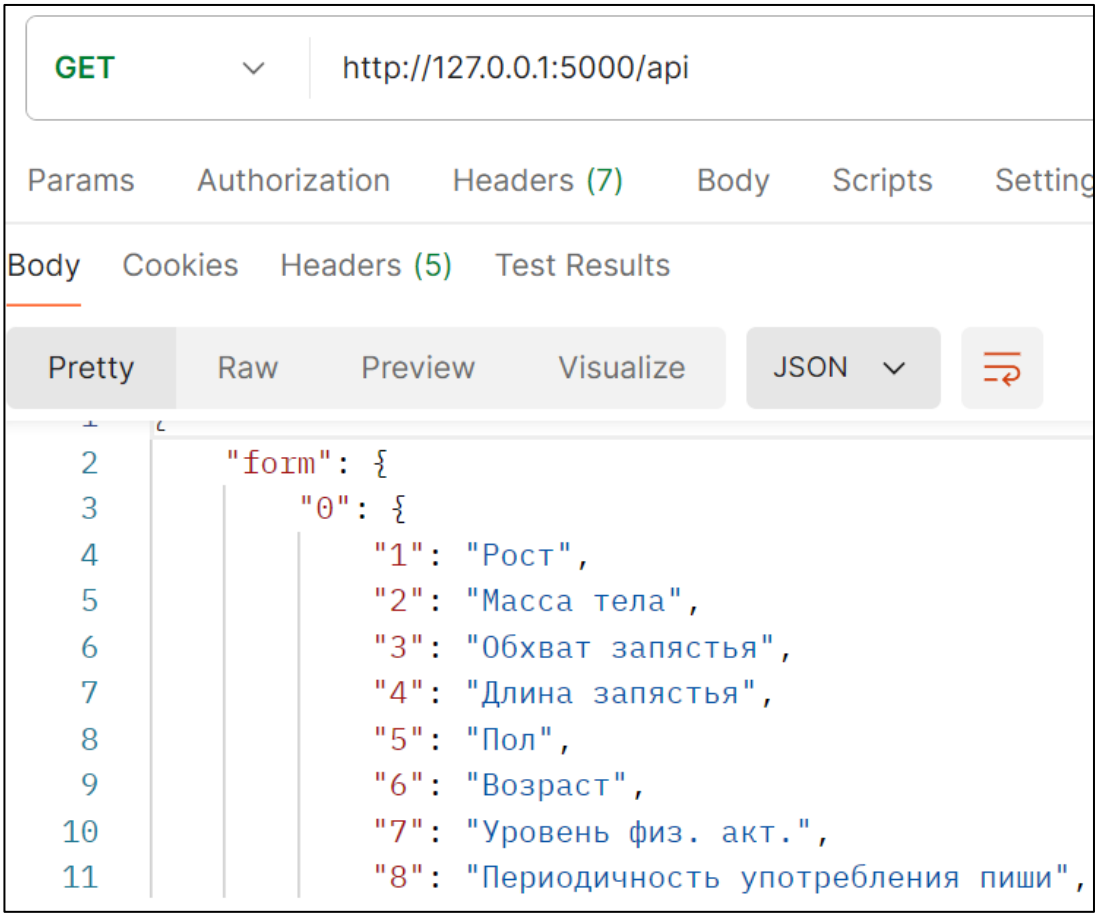


Рисунок 30 – Результат первого теста

Второй тест включал в себя проверку корректности работы предоставления программ тренировок на основании данных человека. Результат теста приведен на рисунке 31.

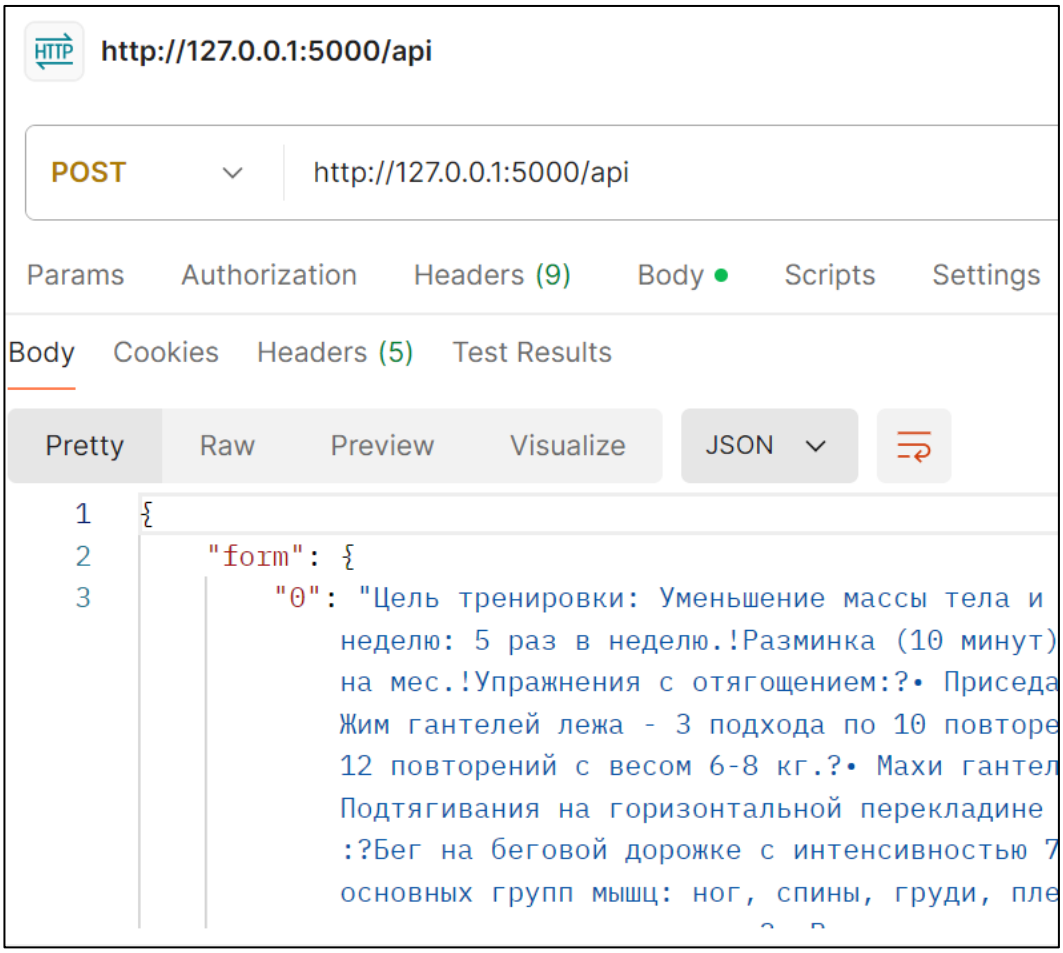


Рисунок 31 – Результат второго теста

Вывод по шестой главе

В данной главе было представлено функциональное тестирование веб-сервиса. Также было проведено тестирование открытого API. По результатам тестирования все тесты пройдены.

ЗАКЛЮЧЕНИЕ

В рамках выпускной квалификационной работы был реализован веб-сервис для создания программ тренировок пользователей фитнес центра. Были выполнены следующие задачи:

1) проведен обзор аналогичных приложений сервисов;
2) разработано приложение для сбора данных;
3) проведено тестирование приложения для сбора данных;
4) спроектирован и реализован веб-сервис;
5) проведено тестирование разработанного веб-сервиса.

В ходе работы мной были основаны методы разработки веб-сервисов с применением фреймворка Flask, были улучшены навыки программирования на языке Python.

В рамках данной работы была подготовлена научная статья: Силкина Н.С., Валиулин А.А. Применение искусственных нейронных сетей для создания программ тренировок. // 76-я научная конференция профессорско-преподавательского состава ЮУрГУ. Челябинск, ЮУрГУ, 2024 г. (принята к печати).

Было разработано приложение для сбора данных со столба здоровья, также был получен акт о внедрении в промышленную эксплуатацию данной программы.

Был получен акт о внедрении веб-сервиса в промышленную эксплантацию.

ЛИТЕРАТУРА

1. Тренировки Workout Trainer. [Электронный ресурс] URL: https://play.google.com/store/apps/details?id=com.skimble.workouts (дата обращения: 21.04.2024 г.).

2. Фитнес тренер FitProSport. [Электронный ресурс] URL: https://play.google.com/store/apps/details?id=fitness.fitprosport (дата обращения: 21.04.2024 г.).

3. Тренировки для дома. [Электронный ресурс] URL: https://play.google.com/store/apps/details?id=homeworkout.homeworkouts.noequipment (дата обращения: 21.04.2024 г.).

4. Mi Fitness (Xiaomi Wear). [Электронный ресурс] URL: https://play.google.com/store/apps/details?id=com.xiaomi.wearable&hl=ru&gl=US&pli=1 (дата обращения: 21.04.2024 г.).

5. Google Fit: трекер активности). [Электронный ресурс] URL: https://play.google.com/store/apps/details?id=com.google.android.apps.fitness&hl=ru&gl=US (дата обращения: 21.04.2024 г.).

6. Python. [Электронный ресурс] URL: https://www.python.org/ (дата обращения: 21.04.2024 г.).

7. Tesseract. [Электронный ресурс] URL: https://github.com/tesseract-ocr/tesseract (дата обращения: 21.04.2024 г.).

8. Pillow (PIL Fork) 10.1.0 documentation. [Электронный ресурс] URL: https://pillow.readthedocs.io/en/stable/ (дата обращения: 21.04.2024 г.).

9. Pytesseract 0.3.10. [Электронный ресурс] URL: https://pypi.org/project/pytesseract/ (дата обращения: 21.04.2024 г.).

10. OS – Miscellaneous operating system interfaces. [Электронный ресурс] URL: https://docs.python.org/3/library/os.html (дата обращения: 21.04.2024 г.).

11. CSV – CSV File Reading and Writing. [Электронный ресурс] URL: https://docs.python.org/3/library/csv.html (дата обращения: 21.04.2024 г.).

12. Thread-based parallelism – Python 3.12.1 documentation. [Электронный ресурс] URL: https://docs.python.org/3/library/threading.html (дата обращения: 21.04.2024 г.).

13. Модуль time – Time access and conversions. [Электронный ресурс] URL: https://docs.python.org/3/library/time.html (дата обращения: 21.04.2024 г.).

14. Welcome to Flask – Flask Documentation (3.0.x). [Электронный ресурс] URL: https://flask.palletsprojects.com/en/3.0.x/ (дата обращения: 21.04.2024 г.).

15. Jinja – Jinja Documentation (3.1.x). [Электронный ресурс] URL: https://jinja.palletsprojects.com/en/3.1.x/ (дата обращения: 21.04.2024 г.).

16. Старолетов, С. М. Основы тестирования и верификации программного обеспечения: учебное пособие. – Санкт–Петербург: Лань, 2020. – 344 с.

17. CTGAN. [Электронный ресурс] URL: https://github.com/sdv-dev/CTGAN (дата обращения: 21.04.2024 г.).

18. Овчаров Д.О. Индивидуальное дозирование физической нагрузки для разных типов телосложения спортсменов, занимающихся пауэрлифтингом. // Вестник науки, 2020. – Т. 2. – №. 2 (23). – С. 41 – 45.

19. Дерябина Г.И., Новикова А. Г. Фитнес-технологии в оздоровительной тренировке женщин на основе учета соматотипа. // Вестник Тамбовского университета. Серия: Гуманитарные науки, 2012. – №. 10. – С. 203 – 207.

20. Петров Д.В. Особенность построения тренировочного процесса в бодибилдинге с учетом различных соматотипов спортсменов. // Современное педагогическое образование, 2021. – №. 2. – С. 140 – 142.).

21. KNeighborsClassifier. [Электронный ресурс] URL: https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html (дата обращения: 21.04.2024 г.).

22. DecisionTreeClassifier. [Электронный ресурс] URL: https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html#sklearn.tree.DecisionTreeClassifier (дата обращения: 21.04.2024 г.).

23. RandomForestClassifier. [Электронный ресурс] URL: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html (дата обращения: 21.04.2024 г.).

24. SVC. [Электронный ресурс] URL: https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html#sklearn.svm.SVC (дата обращения: 21.04.2024 г.).

25. Keras: Deep Learning for humans. [Электронный ресурс] URL: https://keras.io/ (дата обращения: 21.04.2024 г.).

26. PyCharm. [Электронный ресурс] URL: https://www.jetbrains.com/ru-ru/pycharm/ (дата обращения: 21.04.2024 г.).

27. SQLite. [Электронный ресурс] URL: https://www.sqlite.org/ (дата обращения: 21.04.2024 г.).

ПРИЛОЖЕНИЕ

Архитектура разработанных нейросетевых моделей представлена на рисунке 1.

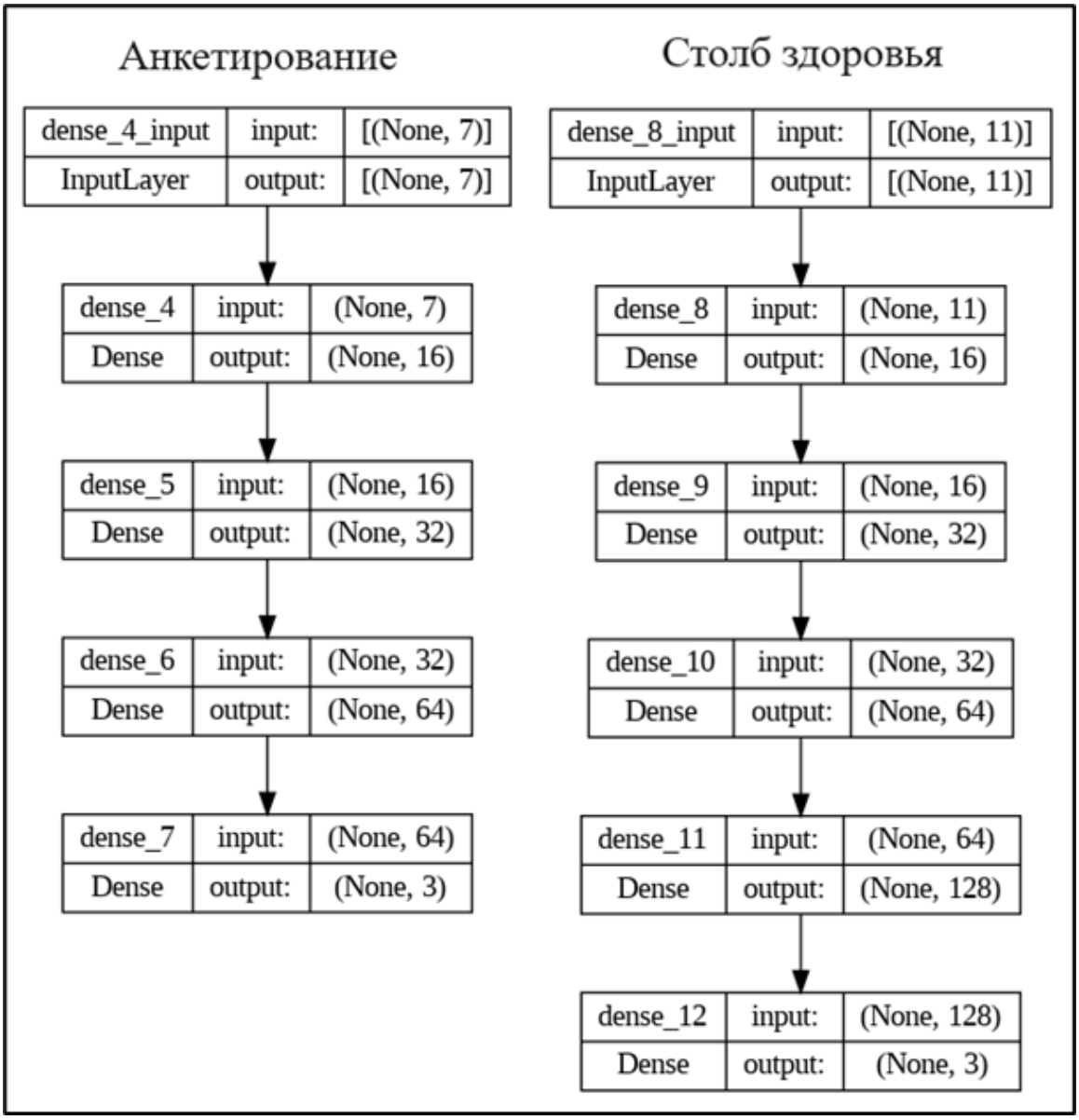


Рисунок 1 – Архитектура нейросетевых моделей