

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»  
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

*Факультет*

вычислительной техники

*Кафедра*

МОиПЭВМ

Направление подготовки 09.03.02 «Информационные системы и технологии»

Направленность Информационные системы и технологии

**БАКАЛАВРСКАЯ РАБОТА**

на тему

**Приложение для предоставления**  
**программ спортивных тренировок**

Студент

\_\_\_\_\_  
(подпись, дата)

Орешкина Арина Андреевна

\_\_\_\_\_  
(ФИО полностью)

Руководитель

\_\_\_\_\_  
(подпись, дата)

Кольчугина Е.А.

\_\_\_\_\_  
(фамилия, инициалы)

Нормоконтролёр

\_\_\_\_\_  
(подпись, дата)

Такташкин Д.В.

\_\_\_\_\_  
(фамилия, инициалы)

Работа допущена к защите (протокол заседания кафедры от 9.06.2025 г. № 12)

Заведующий кафедрой

\_\_\_\_\_  
(подпись)

Козлов А.Ю.

\_\_\_\_\_  
(фамилия, инициалы)

Работа защищена с отметкой \_\_\_\_\_ (протокол заседания ГЭК от \_\_\_\_\_ № \_\_\_\_\_)

Секретарь ГЭК

\_\_\_\_\_  
(подпись)

Попова Н.А.

\_\_\_\_\_  
(фамилия, инициалы)

Пенза, 2025

Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет

Вычислительной техники

Кафедра

МО и ПЭВМ

«Утверждаю»  
Заведующий кафедрой  
А.Ю. Козлов  
«20» декабря 2024 г

**ЗАДАНИЕ  
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ  
БАКАЛАВРА**

1. Студент Орешкина Арина Андреевна гр. 21ВИ1 факультета ВТ  
<Фамилия, имя, отчество полностью>

2. Тема работы Приложение для предоставления программ спортивных тренировок

Тема утверждена приказом ПГУ № \_\_\_\_\_ от " \_\_\_\_\_ " \_\_\_\_\_ 20 \_\_\_\_\_

2. Руководитель работы д.т.н., профессор кафедры «МОиПЭВМ» Кольчугина Е.А.

3. Задание на работу (назначение разработки, исходные данные и т.п.)

1) *Разработать приложение для предоставления программ спортивных тренировок*

2) *Среда функционирования: web-браузер под управлением ОС Windows 11*

3) *Среда разработки: ОС Windows 11*

*Языковые средства разработки: C#, JavaScript, SQL, UML*

*Инструментальные средства: MS Visual Studio 2022, NetBeans, MS SQL Server, Swagger, Draw.io, Visual Paradigm 17.1.*

*Браузеры для тестирования: Google Chrome, Mozilla Firefox, Yandex*

4) *Основные функции приложения:*

*а) регистрация и авторизация пользователей;*

*б) добавление и редактирование спортивных тренировок;*

*в) добавление и редактирование программ, состоящих из спортивных тренировок;*

*г) добавление тренировок в избранные;*

*д) рекомендация тренировок с расчетов калорий на основе веса пользователя.*

5) *Особенности: у рекомендуемых тренировок рассчитываются калории,*

|          |          |            |            |
|----------|----------|------------|------------|
| 4.       | Перечень | подлежащих | разработке |
| вопросов |          |            |            |

- 
- 
- 
- 
- 
- 

| №<br>п/п | Наименование этапов работы                       | Объем<br>работы | Срок<br>выполнения         | Подпись<br>руководителя |
|----------|--------------------------------------------------|-----------------|----------------------------|-------------------------|
| 1        | Анализ предметной области и требований к системе | 10%             | 08.04.2025 –<br>14.04.2025 |                         |
| 2        | Проектирование программных средств               | 20%             | 15.04.2025 –<br>25.04.2025 |                         |
| 3        | Реализация программных средств                   | 30%             | 26.04.2025 –<br>12.05.2025 |                         |
| 4        | Тестирование программных средств                 | 20%             | 13.05.2025 –<br>22.05.2025 |                         |
| 5        | Оформление пояснительной записки                 | 20%             | 23.05.2025 –<br>03.06.2025 |                         |

Руководитель бакалаврской работы \_\_\_\_\_  
(подпись, дата)

Задание к исполнению принял студент \_\_\_\_\_  
(подпись, дата)

Работу к защите допустить  
Декан ФВТ \_\_\_\_\_ Л.Р. Фионова  
(подпись, дата)

## Реферат

Пояснительная записка содержит 148 листов, 39 рисунков, 16 таблиц, 20 использованных источников и 6 приложений.

ПРОГРАММЫ СПОРТИВНЫХ ТРЕНИРОВОК, WEB-ПРИЛОЖЕНИЕ, ОДНОСТРАНИЧНОЕ ПРИЛОЖЕНИЕ, REST API, UML, SQL, C#, ASP.NET CORE, JAVASCRIPT, БАЗА ДАННЫХ.

Объект исследования – приложение для предоставления программ спортивных тренировок.

Цель выпускной квалификационной работы – спроектировать, затем реализовать приложение для предоставления спортивных тренировок с возможностью рекомендаций тренировок на основе веса человека.

Технологии разработки – в среде разработки Apache NetBeans была разработана клиентская часть приложения на языке программирования JavaScript. В среде разработки Microsoft Visual Studio на языке программирования C# была реализована серверная часть приложения. В качестве системы управления базами данных была выбрана Microsoft SQL Server.

Результаты работы – в рамках выпускной квалификационной работы были реализованы клиентская и серверная часть приложения, предназначенного для предоставления программ спортивных тренировок. Все предъявленные требования к созданию приложения выполнены.

|           |      |                |         |      |                                                                                        |  |  |              |      |        |     |
|-----------|------|----------------|---------|------|----------------------------------------------------------------------------------------|--|--|--------------|------|--------|-----|
|           |      |                |         |      | ПГУ 09.03.02– 8BP211.21 ПЗ                                                             |  |  |              |      |        |     |
|           |      |                |         |      |                                                                                        |  |  |              |      |        |     |
| Изм.      | Лист | № докум.       | Подпись | Дата |                                                                                        |  |  |              |      |        |     |
| Разраб.   |      | Орешкина А.А.  |         |      | Приложение для предоставления программ спортивных тренировок.<br>Пояснительная записка |  |  | Лит.         | Лист | Листов |     |
| Пров.     |      | Кольчугина Е.А |         |      |                                                                                        |  |  |              |      | 4      | 148 |
| Реценз.   |      |                |         |      |                                                                                        |  |  | Группа 21ВИ1 |      |        |     |
| Н. Контр. |      | Такташкин Д.В  |         |      |                                                                                        |  |  |              |      |        |     |
| Утв.      |      | Козлов А.Ю.    |         |      |                                                                                        |  |  |              |      |        |     |

## Содержание

|                                                                                                 |    |
|-------------------------------------------------------------------------------------------------|----|
| Введение.....                                                                                   | 7  |
| 1 Анализ требований к поставленной задаче и предметной области .....                            | 8  |
| 1.1 Постановка задачи .....                                                                     | 8  |
| 1.2 Анализ аналогов .....                                                                       | 9  |
| 1.3 Выбор методов и средств разработки .....                                                    | 13 |
| 2 Проектирование приложения для предоставления программ<br>спортивных тренировок.....           | 17 |
| 2.1 Архитектура приложения.....                                                                 | 17 |
| 2.2 Проектирование структур данных .....                                                        | 18 |
| 2.2.1 Концептуальная модель базы данных .....                                                   | 18 |
| 2.2.2 Логическая модель базы данных.....                                                        | 19 |
| 2.3 Проектирование пользовательского интерфейса .....                                           | 20 |
| 2.4 Проектирование классов приложения для предоставления<br>программ спортивных тренировок..... | 22 |
| 3 Реализация приложения для предоставления программ спортивных<br>тренировок .....              | 24 |
| 3.1 Реализация базы данных .....                                                                | 24 |
| 3.2 Реализация серверной части приложения .....                                                 | 24 |
| 3.3 Реализация клиентской части приложения .....                                                | 27 |
| 3.4 Реализация интерфейса приложения .....                                                      | 29 |
| 4 Отладка и тестирование .....                                                                  | 38 |
| 4.1 Обоснование выбора режима тестирования .....                                                | 38 |
| 4.2 Тестирование приложения для предоставления программ<br>спортивных тренировок.....           | 39 |

|       |                                                                               |     |
|-------|-------------------------------------------------------------------------------|-----|
| 4.2.1 | Функциональное тестирование .....                                             | 39  |
| 4.2.2 | Модульное тестирование .....                                                  | 40  |
| 5     | Контроль качества .....                                                       | 43  |
| 5.1   | Обоснование выбора метрик кода.....                                           | 43  |
| 5.2   | Оценка качества программного кода .....                                       | 43  |
| 5.2.1 | Метрика Холстеда.....                                                         | 43  |
| 5.2.2 | Метрика Чепина .....                                                          | 45  |
| 6     | Расчет бюджета разработки.....                                                | 47  |
| 6.1   | Планирование разработки и оценка бюджета.....                                 | 47  |
|       | Заключение .....                                                              | 49  |
|       | Список использованных источников .....                                        | 51  |
|       | Приложение А (справочное) Глоссарий.....                                      | 54  |
|       | Приложение Б (рекомендуемое) Атрибуты сущностей базы данных .....             | 56  |
|       | Приложение В (обязательное) Листинг серверной части приложения.....           | 59  |
|       | Приложение Г (обязательное) Листинг клиентской части.....                     | 94  |
|       | Приложение Д (рекомендуемое) Результаты функционального<br>тестирования ..... | 140 |
|       | Приложение Е (рекомендуемое) Листинг модульных тестов .....                   | 143 |

## Введение

Каждый день люди занимаются какой-либо умственной деятельностью, связанной с учебой или работой. Подготовка к занятиям и сидячая работа способствуют малоподвижному образу жизни, что в свою очередь негативно сказывается на здоровье. Ухудшается не только физическое состояние, но и психологическое, появляются усталость и стресс [1].

Чтобы оставаться работоспособным и успешно справляться со всеми поставленными задачами человеку необходимо заниматься физической активностью. Но не все могут позволить себе выделить время и бюджет для регулярного посещения спортзала.

Приложения для предоставления спортивных тренировок являются актуальными, так как дают возможность людям самостоятельно заниматься спортом дома в любое удобное время.

Целью выпускной квалификационной работы является разработка приложения для предоставления программ спортивных тренировок. Приложение поможет поддерживать людям физическую активность дома, предоставляя видеозаписи тренировок.

Для достижения цели необходимо выполнить следующие задачи:

- провести анализ предметной области и требований;
- провести анализ систем аналогичного назначения;
- провести проектирование структур данных, алгоритмов и интерфейсов приложения;
- реализовать приложение;
- провести тестирование.

Выполнение поставленных задач поможет разработать эффективное и удобное приложение для предоставления спортивных тренировок.

## **1 Анализ требований к поставленной задаче и предметной области**

### **1.1 Постановка задачи**

Основной задачей выпускной квалификационной работы является разработка приложения для предоставления программ спортивных тренировок.

Главная страница приложения должна содержать ленту программ. Каждая программа должна содержать название, описание и набор спортивных тренировок. Тренировки будут объединяться в программы по какому-либо общему критерию: вид спорта, сложность тренировок, продолжительность. Тренировка должна содержать название, описание, продолжительность и видеозапись упражнений. Видеозапись упражнений удобнее будет загружать из видеохостинга, например Rutube. Таким образом можно будет избежать лишней нагрузки и хранить только ссылку видеозаписи.

Приложение должно давать возможность зарегистрироваться новым клиентам и авторизовываться существующим пользователям. Необходимо добавить личный кабинет для зарегистрированных пользователей с возможностью заполнения своих личных данных таких, как дата рождения, рост и вес. Чтобы обеспечить поиск по программам и тренировкам в приложении должны быть фильтры поиска по количеству тренировок в программе, по продолжительности тренировок и по ключевым словам. Для удобства доступа к понравившимся тренировкам у пользователя должна быть возможность добавлять тренировки в избранные. Для администраторов необходимо добавить возможность добавления, редактирования и удаления как тренировок, так и программ.

В результате имеем три объекта данных для использования и хранения: тренировки, программы и пользователи. Появляется необходимость использования СУБД. Также необходимо обеспечить понятный и удобный графический интерфейс.

Описанные требования должны быть учтены при проектировании и разработке приложения для предоставления программ спортивных тренировок.

## 1.2 Анализ аналогов

Был проведен обзор систем аналогичного назначения, по результатам которого можно отметить несколько спортивных сервисов, предоставляющих программы тренировок. Для сравнения были выбраны следующие системы: FitStars, «Живи!», Nike Training Club, Freeletics.

«FitStars» – это фитнес-платформа, предоставляющая программы тренировок. Интерфейс приложения показан на рисунке 1.

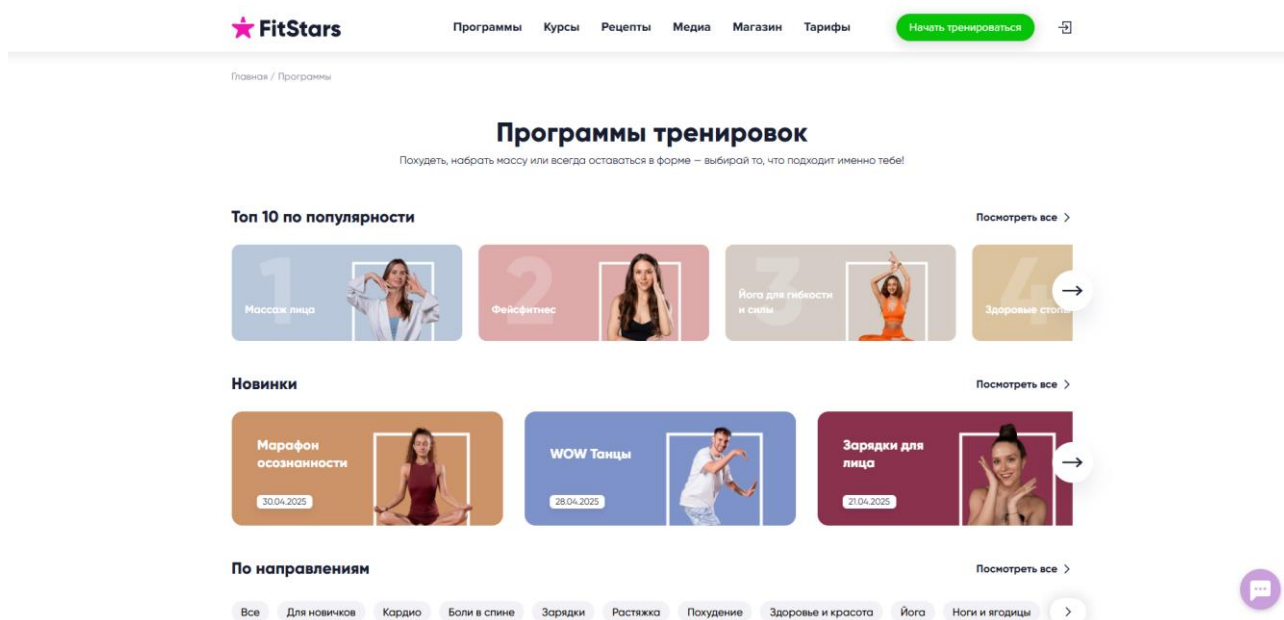


Рисунок 1 – Интерфейс приложения «FitStars»

К достоинствам этого приложения можно отнести разнообразие направлений тренировок, авторские программы, при составлении которых участвуют популярные тренеры и возможность оставлять обратную связь к пройденным программам.

Главным недостатком приложения с точки зрения пользователя является платный доступ. Сервис перегружен контентом, он предоставляет не только спортивные программы, но и медитации, рецепты, статьи на различные темы и магазин спортивного инвентаря. Чтобы получить доступ к программам

тренировок, пользователю придется переплачивать за весь функционал, которым скорее всего он не будет пользоваться в полной мере.

«Первопроходцем» среди российских спортивных приложений является сервис «Живи!». Веб-приложение «Живи!» начало функционировать еще в 2009 году. Его интерфейс представлен на рисунке 2.

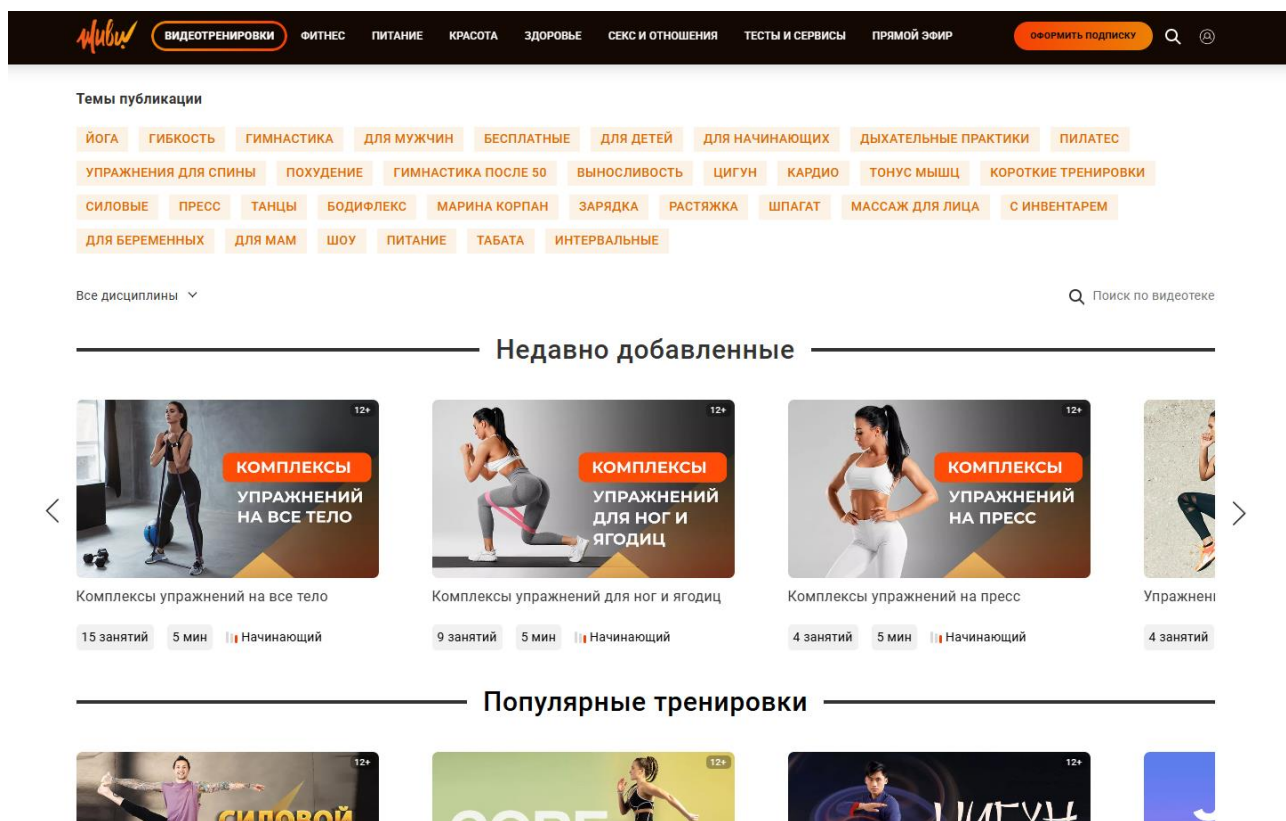


Рисунок 2 – Интерфейс приложения «Живи!»

Преимущества этого приложения можно назвать проведение прямых эфиров со спортивными тренировками, наличие бесплатного доступа к тренировкам и большой архив программ разных направлений.

К минусам спортивного сервиса можно отнести устаревший дизайн и наличие контента на темы, не связанные со здоровьем и спортом. Материалы на отвлеченные темы не несут практической пользы для клиентов приложения.

«Nike Training Club» – мобильное фитнес-приложение со спортивными тренировками от компании «Nike», которая также занимается производством спортивной одежды и обуви. Интерфейс приложения показан на рисунке 3.

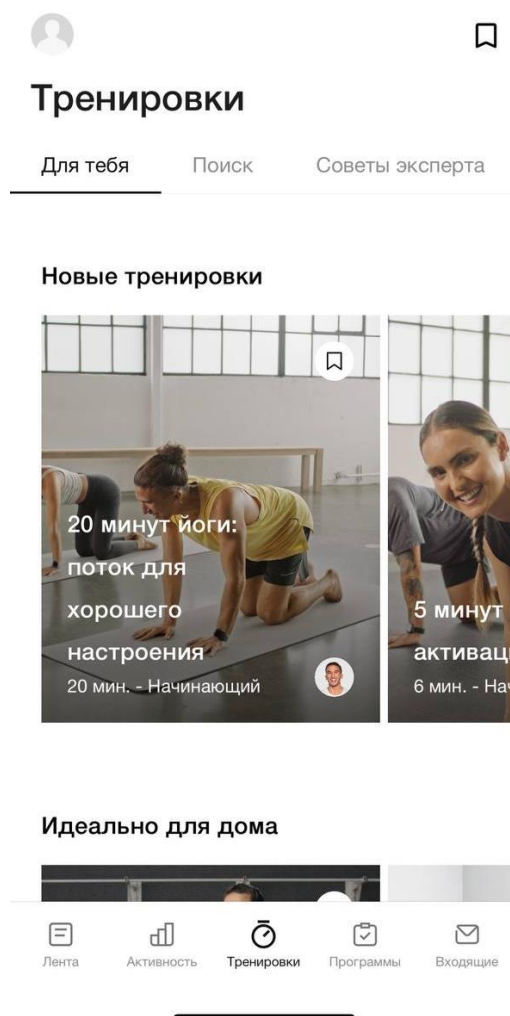


Рисунок 3 – Интерфейс приложения «Nike Training Club»

Плюсами этого приложения можно назвать большое количество видов тренировок, разбиение тренировок по категориям для разных групп мышц и спортивного снаряжения, современный дизайн. Также пользователю доступна история активности. Приложение является полностью бесплатным.

К минусам приложения можно отнести малое количество программ тренировок, наличие лишнего функционала, в частности возможность добавления друзей, и наличие только мобильной версии приложения. Не каждому пользователю удобно смотреть технику упражнений с небольшого экрана мобильного устройства.

«Freeletics» – мобильное фитнес-приложение с применением искусственного интеллекта. Интерфейс приложения изображен на рисунке 4.

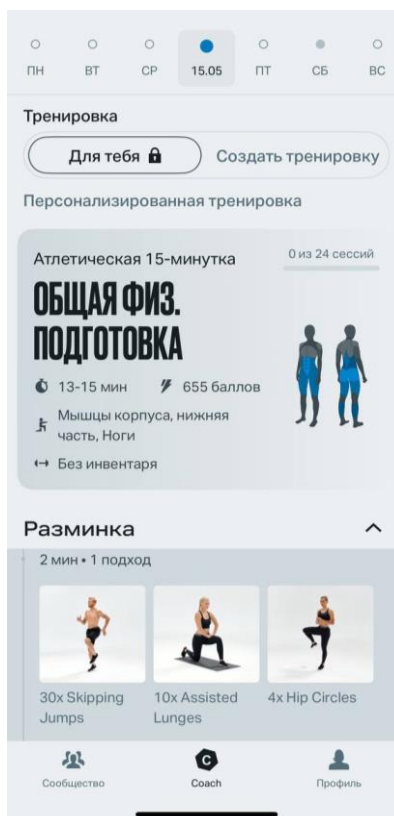


Рисунок 4 – Интерфейс приложения «Freeletics»

Отличительным аспектом этого приложения является составление тренировок искусственным интеллектом на основе данных пользователя. В приложении доступны отдельно упражнения, из которых и составляются персонализированные тренировки.

Главным минусом приложения является сильно ограниченный бесплатный доступ. Для бесплатного использования доступно небольшое количество тренировок. При старте приложения необходимо довольно долго выбирать опции и настройки для составления персонализированных тренировок. Доступна только мобильная версия. Также приложение переведено на русский язык не полностью.

Для проведения анализа функционала рассмотренных приложений аналогичного назначения по определенным условиям использовалась трехбалльная шкала оценивания со следующими значениями:

- 0 – условие не выполнено;
- 1 – условие выполнено частично;

– 2 – условие выполнено полностью.

Результаты сравнения спортивных сервисов, предоставляющих программы тренировок, представлены в таблице 1.

Таблица 1 – Результаты сравнения приложений аналогичного назначения

| Критерий / Условие                            | «FitStars» | «Живи!» | «Nike Training Club» | «Freeletics» |
|-----------------------------------------------|------------|---------|----------------------|--------------|
| 1. Поиск по программам и тренировкам          | 2          | 2       | 2                    | 1            |
| 2. Возможность выбрать избранные тренировки   | 2          | 0       | 0                    | 0            |
| 3. Бесплатный доступ                          | 0          | 1       | 2                    | 1            |
| 4. Доступ с всех устройств                    | 2          | 1       | 0                    | 0            |
| 5. Рекомендации на основе данных пользователя | 0          | 0       | 0                    | 2            |

По результатам сравнения систем аналогичного назначения больше всего баллов набрало приложение «FitStars», но при этом оно не соответствует всем условиям и остается полностью платным. При проектировании и реализации разрабатываемого приложения для предоставления программ спортивных тренировок будут учтены достоинства и недостатки проанализированных систем, что поможет разработать удобное приложение с полезным функционалом.

### 1.3 Выбор методов и средств разработки

Диаграмма вариантов использования помогает определить функциональные требования к системе, показывает типичные взаимодействия между пользователем и системой. Диаграмма дает понимание, какие действия будут выполнять пользователи, какие варианты использования будут содержать в себе другие варианты использования, а какие наоборот будут расширять [2]. Спроектированная диаграмма вариантов использования для приложения для предоставления программ спортивных тренировок представлена на рисунке 5.

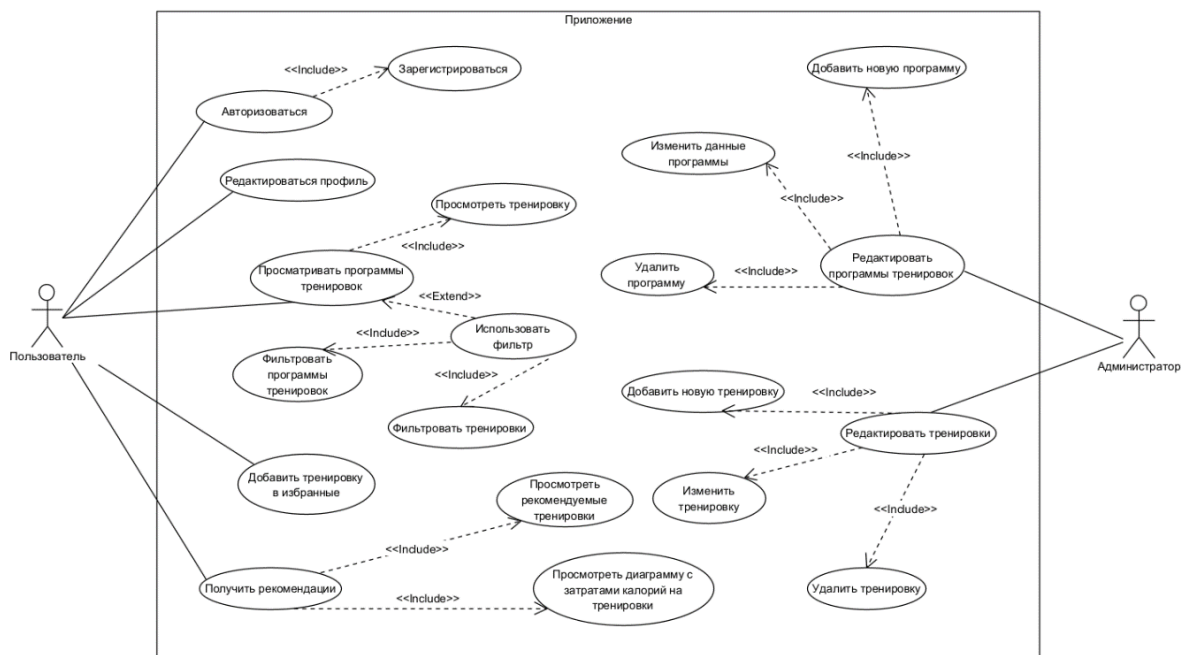


Рисунок 5 – Диаграмма вариантов использования приложения

Чтобы детализировать и описать основные действия работы наиболее важных вариантов использования были разработаны их спецификации.

Спецификация варианта использования «Просматривать программы тренировок» показана на рисунке 6.

|                                                                                                                                                                                                                                                                                                                                                   |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Прецедент: Просматривать программы тренировок                                                                                                                                                                                                                                                                                                     |
| ID: 3                                                                                                                                                                                                                                                                                                                                             |
| Краткое описание:<br>Пользователь просматривает программы тренировок в системе                                                                                                                                                                                                                                                                    |
| Главные актеры:<br>Пользователь                                                                                                                                                                                                                                                                                                                   |
| Второстепенные актеры:<br>Нет                                                                                                                                                                                                                                                                                                                     |
| Предусловие:<br>1. Пользователь авторизовался в системе<br>2. Пользователь перешел на вкладку «Программы»                                                                                                                                                                                                                                         |
| Основной поток:<br>1. Пользователь просматривает список программ тренировок<br>2. Пользователь выбирает тренировку<br>3. Пользователь начинает просматривать видеозапись тренировки<br>Альтернативные потоки:<br>1.а Пользователь просматривает отфильтрованные программы тренировок<br>1.б Пользователь просматривает отфильтрованные тренировки |
| Постусловия:<br>1. Пользователь просмотрел выбранную тренировку                                                                                                                                                                                                                                                                                   |

Рисунок 6 – Спецификация варианта использования «Просматривать программы тренировок»

Спецификация варианта использования «Добавить тренировку в избранные» представлена на рисунке 7.

|                                                                                                           |
|-----------------------------------------------------------------------------------------------------------|
| Прецедент: Добавить тренировку в избранные                                                                |
| ID: 4                                                                                                     |
| Краткое описание:<br>Пользователь просматривает тренировку и добавляет ее в избранные                     |
| Главные актеры:<br>Пользователь                                                                           |
| Второстепенные актеры:<br>Нет                                                                             |
| Предусловие:<br>1. Пользователь авторизовался в системе<br>2. Пользователь перешел на вкладку «Программы» |
| Основной поток:<br>1. Пользователь добавляет в избранные выбранную тренировку                             |
| Постусловия:<br>1. Тренировка добавлена в избранные                                                       |

Рисунок 7 – Спецификация варианта использования «Добавить тренировку в избранные»

Спецификация варианта использования «Получить рекомендации» показана на рисунке 8.

|                                                                                                                                                                                                                                                                                                     |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Прецедент: Получить рекомендации                                                                                                                                                                                                                                                                    |
| ID: 5                                                                                                                                                                                                                                                                                               |
| Краткое описание:<br>Пользователь получает список рекомендованных тренировок                                                                                                                                                                                                                        |
| Главные актеры:<br>Пользователь                                                                                                                                                                                                                                                                     |
| Второстепенные актеры:<br>Система                                                                                                                                                                                                                                                                   |
| Предусловие:<br>1. Пользователь перешел на вкладку «Личный кабинет»<br>2. Пользователь указал свой вес в личном кабинете                                                                                                                                                                            |
| Основной поток:<br>1. Пользователь указывает цель тренировок<br>2. Система подсчитывает сколько калорий потратит пользователь, выполняя каждую тренировку<br>3. Пользователь получает список рекомендованных тренировок<br>4. Система строит диаграмму с количеством расхода калорий и тренировками |
| Постусловия:<br>1. Пользователь получит список рекомендованных тренировок                                                                                                                                                                                                                           |

Рисунок 8 – Спецификация варианта использования «Получить рекомендации»

Результатом анализа функциональных требований являются варианты использования разрабатываемого приложения, которые показывают каждый шаг взаимодействия пользователей с системой. Расширение вариантов использования с помощью спецификаций позволит тщательнее спроектировать систему.

Серверная часть приложения для предоставления программ спортивных тренировок будет представлять собой многослойную монолитную архитектуру и будет реализована на языке C# в среде программирования Microsoft Visual Studio 2022. Как СУБД была выбрана Microsoft SQL Server 2022.

Клиентская часть будет написана на языке JavaScript с использованием HTML и CSS в среде программирования Apache NetBeans. Серверная и клиентская части будут взаимодействовать через REST API, что обеспечит надежность и производительность системы, а также позволит масштабировать ее при необходимости [3].

## 2 Проектирование приложения для предоставления программ спортивных тренировок

### 2.1 Архитектура приложения

Для приложения будет использоваться монолитный многослойный архитектурный подход. Архитектура будет состоять из 4 слоев:

- UI (пользовательский интерфейс);
- бизнес-логика;
- взаимодействие с БД;
- БД.

Каждый слой будет выполнять определенные задачи в рамках своей области [4]. Они будут оставаться независимыми, что позволит изменениям одного слоя не влиять на работу остальных слоев. Физическое представление показано на рисунке 9.

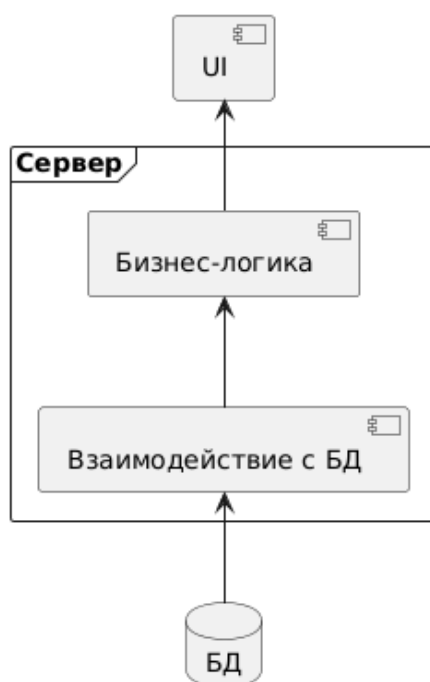


Рисунок 9 – Физическое представление архитектуры

Слой UI (пользовательский интерфейс) обеспечивает взаимодействие с пользователем. В нем обрабатываются клиентские запросы, предоставляются

хранимые данные, а также передаются в дальнейший слой бизнес-логики. Слой содержит контроллеры для авторизации и регистрации пользователей, для обработки запросов, связанных с тренировками и программами тренировок.

Слой бизнес-логики обеспечивает выполнение процессов и операций с данными, такие как создание, проверка, обновление и удаление. Функционал классов этого слоя следующий: фильтрация тренировок, создание хранимых объектов, генерация и проверка токенов доступа для пользователей.

Внутри следующего слоя происходит инкапсуляция логики взаимодействия с БД и управление сущностями. Классы этого слоя инициализируют операции с БД такие, как чтение, создание, обновление и удаление данных в БД.

Слой БД хранит все данные о пользователях, тренировках и программах тренировок. Это слой внешней базы данных.

## 2.2 Проектирование структур данных

### 2.2.1 Концептуальная модель базы данных

Проектирование базы данных следует начать с построения концептуальной модели. Концептуальная модель базы данных не зависит от физической реализации. Необходимо определить типы сущностей, определить связи между ними, выявить атрибуты каждой сущности [5].

Описание основных сущностей предметной области приложения для предоставления программ спортивных тренировок представлено в таблице 2.

Таблица 2 – Описание сущностей

| Название сущности    | Идентификатор сущности | Описание сущности                                    |
|----------------------|------------------------|------------------------------------------------------|
| Пользователь         | Users                  | Содержит сведения о пользователе                     |
| Тренировка           | Workouts               | Содержит сведения о спортивных тренировках           |
| Программа тренировок | WorkoutPrograms        | Содержит сведения о программах спортивных тренировок |

Разработанная диаграмма концептуальной модели базы данных представлена на рисунке 10.

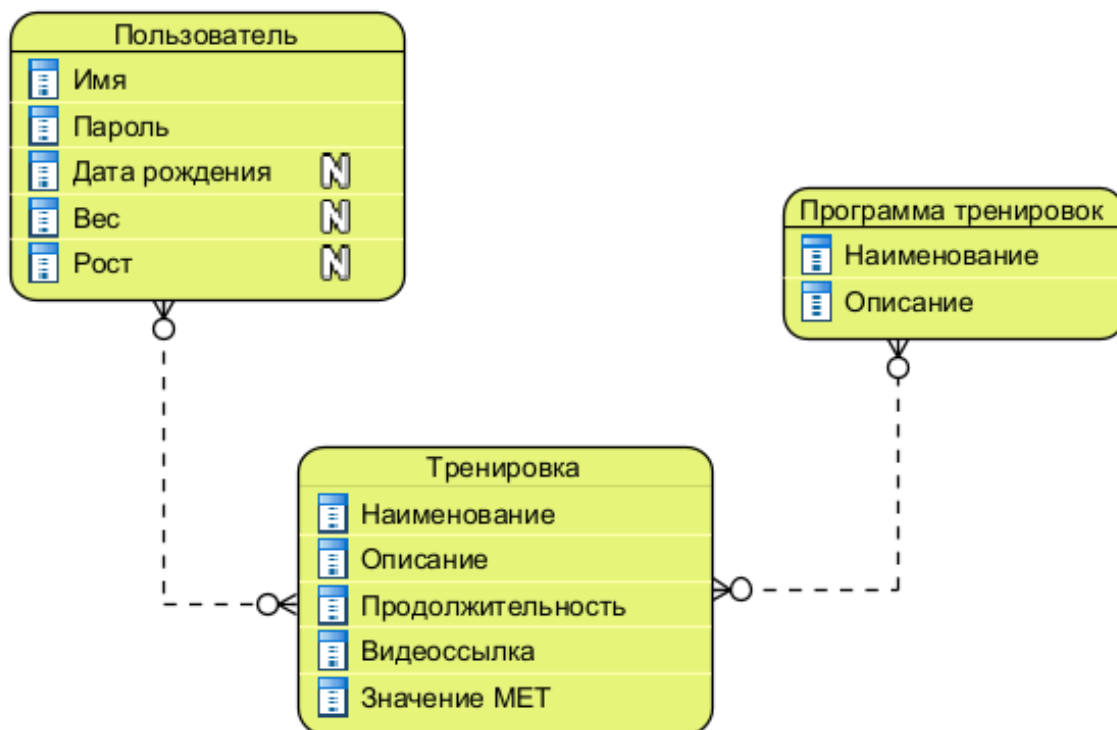


Рисунок 10 – Диаграмма концептуальной модели

Диаграмма концептуальной модели была построена в среде Visual Paradigm 17.1 в нотации IDEF1X.

### 2.2.2 Логическая модель базы данных

Логическая модель уточняет концептуальную модель базы данных, преобразовывает и дополняет атрибуты сущностей, добавляет первичные и внешние ключи [5]. Логическая модель показывает, что должно быть сделано, то есть является основой для физической модели.

Отношения «многие ко многим» были заменены промежуточными сущностями. Таким образом модель базы данных была преобразована в более подходящую форму для реляционной СУБД [6].

Логическая модель базы данных была построена на основе концептуальной модели. Диаграмма логической модели показана на рисунке 11.

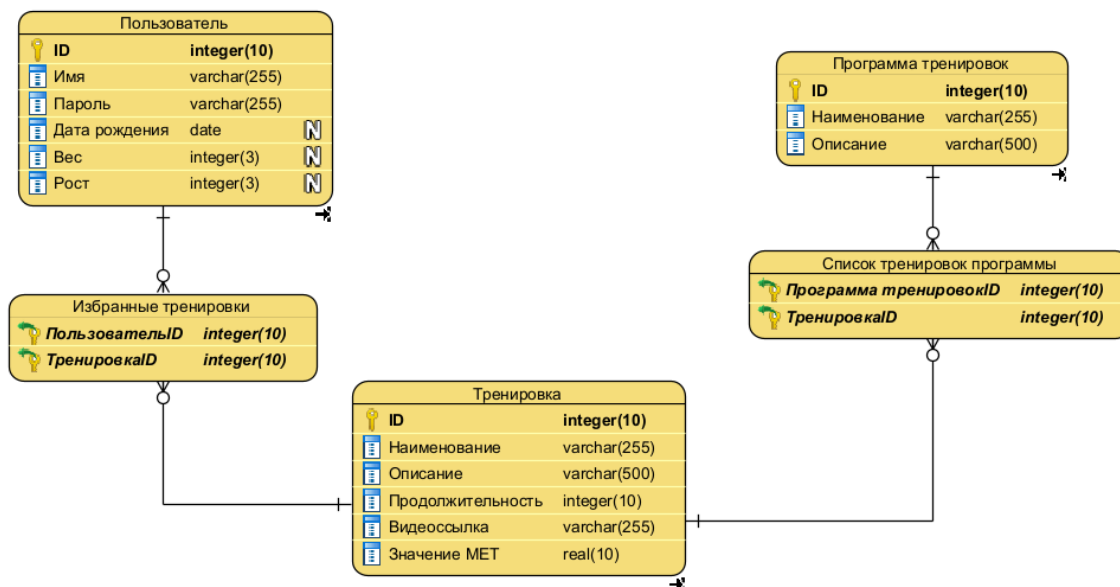


Рисунок 11 – Диаграмма логической модели

Описание атрибутов каждой сущности представлено в приложении Б.

## 2.3 Проектирование пользовательского интерфейса

Пользовательский интерфейс разрабатываемого приложения будет включать в себя одну главную страницу, то есть клиентское приложение будет одностраничным. Такое приложение будет менять только ту часть интерфейса, которая изменилась, вместо всей страницы. Одностраничное приложение с быстрым и содержательным интерфейсом будет удобным в использовании для пользователя [7].

Главная страница пользователя должна содержать вкладки входа и регистрации, личного кабинета, вкладка с лентой программ тренировок и инструментами их фильтрации, а также вкладка избранных тренировок. Во вкладке личного кабинета также должен присутствовать контент с рекомендованными тренировками. При старте тренировки должно открываться модальное окно с видеоплеером, где будет проигрываться видеозапись. Взаимосвязь разделов страницы пользователя показана на рисунке 12.

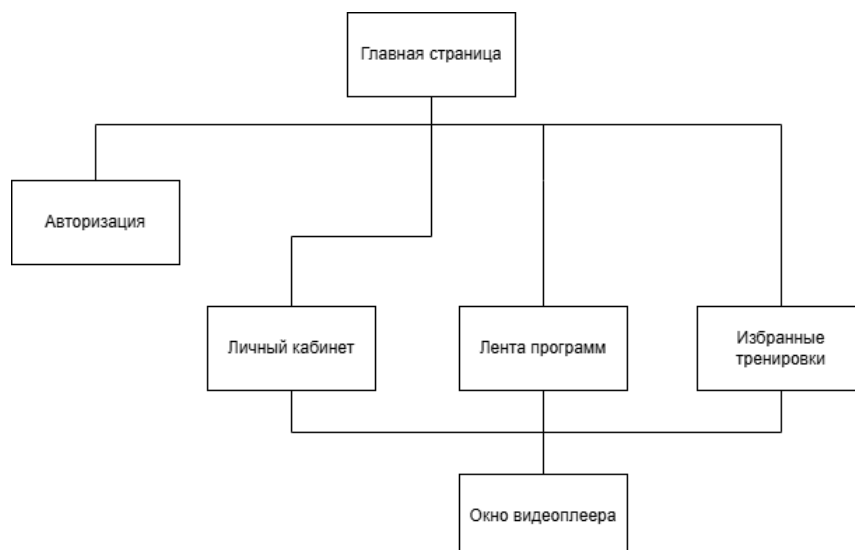


Рисунок 12 – Переходы между разделами страницы пользователя

На главной странице для администратора вкладки личного кабинета и избранных тренировок заменяются вкладками управления программами и тренировками. Взаимосвязь разделов страницы администратора показано на рисунке 13.

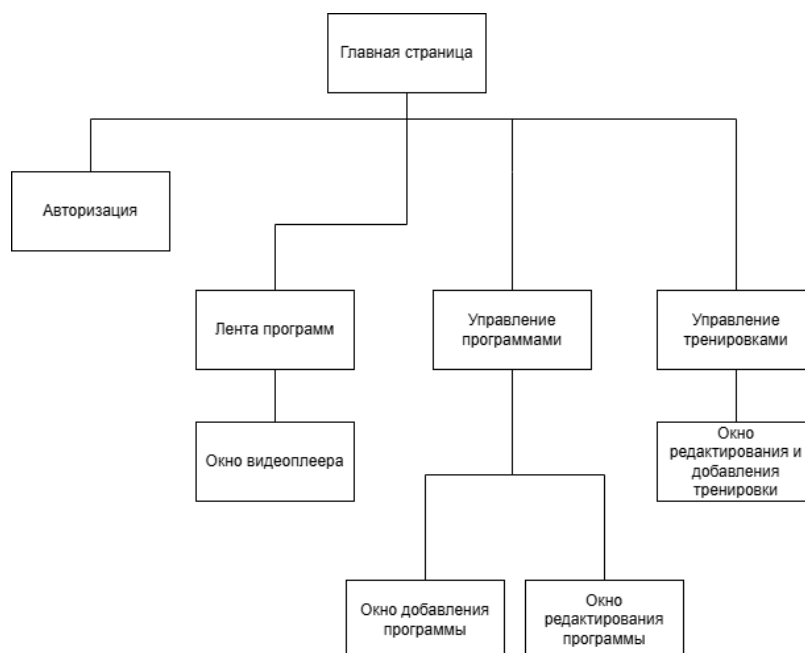


Рисунок 13 – Переходы между разделами страницы администратора

При добавлении программы будет открываться соответствующее модальное окно, как и при редактировании. Для редактирования и добавления тренировки будет использоваться одно модальное окно.

## 2.4 Проектирование классов приложения для предоставления программ спортивных тренировок

Согласно спроектированной архитектуре, была разработана диаграмма классов серверной части приложения. Диаграмма представлена на рисунке 14.

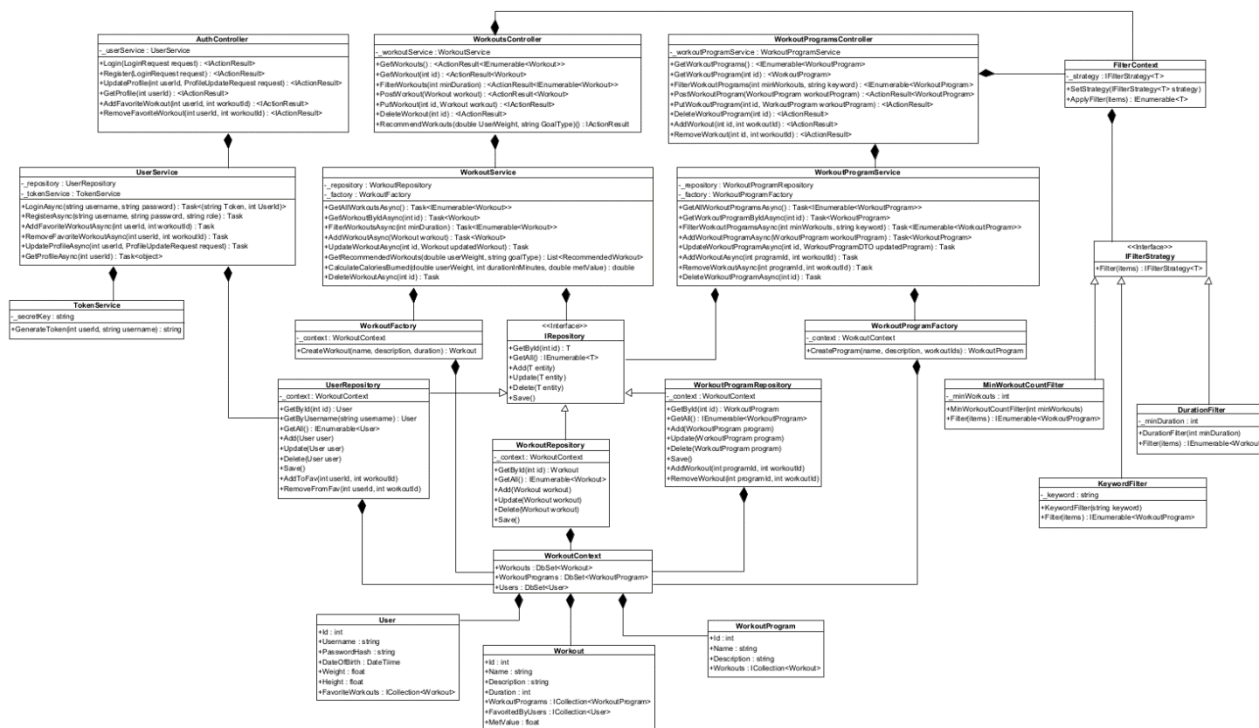


Рисунок 14 – Диаграмма классов

Описание назначения классов серверной части приложения приведено в таблице 3.

Таблица 3 – Назначение классов

| Класс             | Назначение                                               |
|-------------------|----------------------------------------------------------|
| User              | Модель, представляющая данные о пользователе             |
| Workout           | Модель, представляющая данные о тренировке               |
| WorkoutProgram    | Модель, представляющая данные о программе тренировок     |
| WorkoutContext    | Класс, реализующий взаимодействие между приложением и БД |
| UserRepository    | Класс для работы с данными о пользователе                |
| WorkoutRepository | Класс для работы с данными о тренировке                  |

Продолжение таблицы 3

| Класс                     | Назначение                                                                  |
|---------------------------|-----------------------------------------------------------------------------|
| WorkoutProgramRepository  | Класс для работы с данными о программе тренировок                           |
| IRepository               | Интерфейс, предоставляющий набор методов для работы с данными               |
| TokenService              | Класс, генерирующий токены доступа                                          |
| AuthController            | Класс, обрабатывающий запросы, связанные с пользователем                    |
| WorkoutFactory            | Класс, отвечающий за создание тренировок                                    |
| WorkoutProgramFactory     | Класс, отвечающий за создание программ тренировок                           |
| WorkoutsController        | Класс, обрабатывающий запросы, связанные с тренировками                     |
| WorkoutProgramsController | Класс, обрабатывающий запросы, связанные с программами тренировок           |
| MinWorkoutCountFilter     | Класс, реализующий фильтрацию программ тренировок                           |
| DurationFilter            | Класс, реализующий фильтрацию тренировок                                    |
| KeywordFilter             | Класс, реализующий фильтрацию по слову                                      |
| IFilterStrategy           | Интерфейс, предоставляющий набор методов для фильтрации                     |
| FilterContex              | Класс для выбора стратегии фильтрации                                       |
| UserService               | Класс, реализующий бизнес-логику управления данными о пользователях         |
| WorkoutService            | Класс, реализующий бизнес-логику управления данными о тренировках           |
| WorkoutProgramService     | Класс, реализующий бизнес-логику управления данными о программах тренировок |

В результате проектирования классов были описаны типы объектов серверной части приложения и связи между ними [2].

### 3 Реализация приложения для предоставления программ спортивных тренировок

#### 3.1 Реализация базы данных

Физическая модель базы данных отражает физическую реализацию логической модели в конкретной СУБД, определяет организацию файлов [5].

Физическая модель базы данных была построена с использованием концептуальной и логической модели. Диаграмма физической модели представлена на рисунке 15.

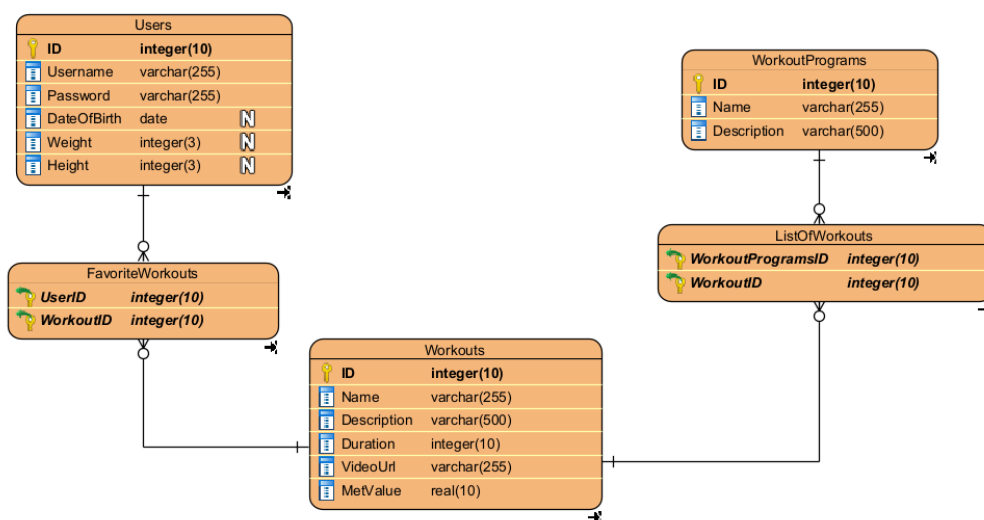


Рисунок 15 – Диаграмма физической модели

Развертывание базы данных было сделано с помощью функции миграции. При миграции модели данных из приложения синхронизируются со схемой базы данных. Миграции выполняются пошагово, что позволяет избежать потери имеющихся данных [8].

#### 3.2 Реализация серверной части приложения

Серверная часть приложения представляет собой монолит. Присутствуют классы моделей данных, отображающие сущности из базы данных. Контроллеры обрабатывают запросы от клиентской части и возвращают ответ. Сервисы

содержат функционал и обеспечивают работу между контроллерами и базой данных.

Серверная часть приложения была реализована в среде программирования Microsoft Visual Studio 2022 на языке #C. Код серверной части представлена в приложении В. Серверная часть содержит группы маршрутов для обработки HTTP-запросов. Группы маршрутов описаны в таблице 4.

Таблица 4 – Группы маршрутов

| URL-адрес                       | HTTP-метод | Описание                                         |
|---------------------------------|------------|--------------------------------------------------|
| /auth/login                     | POST       | Авторизация пользователя                         |
| /auth/register                  | POST       | Регистрация пользователя                         |
| /auth/{userId}/favorites/add    | POST       | Добавление тренировки в избранные                |
| /auth/{userId}/favorites/remove | POST       | Удаление тренировки из избранных                 |
| /auth/{userId}/update-profile   | POST       | Изменение данных о пользователе                  |
| /auth/{userId}                  | GET        | Получение данных конкретного пользователя        |
| /Workouts                       | GET        | Получение всех тренировок                        |
| /Workouts/{id}                  | GET        | Получение данных конкретной тренировки           |
| /Workouts/filter                | GET        | Получение отфильтрованных тренировок             |
| /Workouts                       | POST       | Создание новой тренировки                        |
| /Workouts/{id}                  | PUT        | Изменение данных тренировки                      |
| /Workouts/recommend             | POST       | Получение рекомендуемых тренировок               |
| /Workouts/{id}                  | DELETE     | Удаление тренировки                              |
| /WorkoutPrograms                | GET        | Получение всех программ тренировок               |
| /WorkoutPrograms/{id}           | GET        | Получение данных конкретной программы тренировок |
| /WorkoutPrograms/filter         | GET        | Получение отфильтрованных программ тренировок    |

Продолжение таблицы 4

| URL-адрес                    | HTTP-метод | Описание                              |
|------------------------------|------------|---------------------------------------|
| /WorkoutPrograms             | POST       | Создание новой программы тренировок   |
| /WorkoutPrograms/{id}        | PUT        | Изменение данных программы тренировок |
| /WorkoutPrograms/{id}/add    | POST       | Добавление тренировки в программу     |
| /WorkoutPrograms/{id}/remove | POST       | Удаление тренировки из программы      |
| /WorkoutPrograms/{id}        | DELETE     | Удаление программы тренировок         |

Метод `GetRecommendedWorkouts(double userWeight, string goalType)` класса `WorkoutService` выполняет подбор рекомендуемых для пользователя тренировок на основе его веса и цели тренировки. С помощью метода `CalculateCaloriesBurned(double userWeight, int durationInMinutes, double metValue)` подсчитываются калории, которые потратит пользователь, выполняя определенную тренировку, по следующей формуле:

$$E (\text{ккал}) = \text{MET} * \text{кг} * \text{ч},$$

где MET – метаболический эквивалент тренировки;

кг – вес пользователя в килограммах;

ч – время тренировки в часах.

После подсчета возвращается набор тренировок, на выполнение которых затрачивается количество калорий, необходимых для определенной цели.

Значение MET для каждой тренировки взяты из компендиума физических упражнений, предоставляющего стандартизированный способ измерения и классификации физической активности [9].

Диаграммы развертывания показывает физическое расположение всех составляющих программного обеспечения, на каком оборудовании они размещены [2]. Для демонстрации развертывания всех компонентов приложения была построена диаграмма развертывания, показанная на рисунке 16.

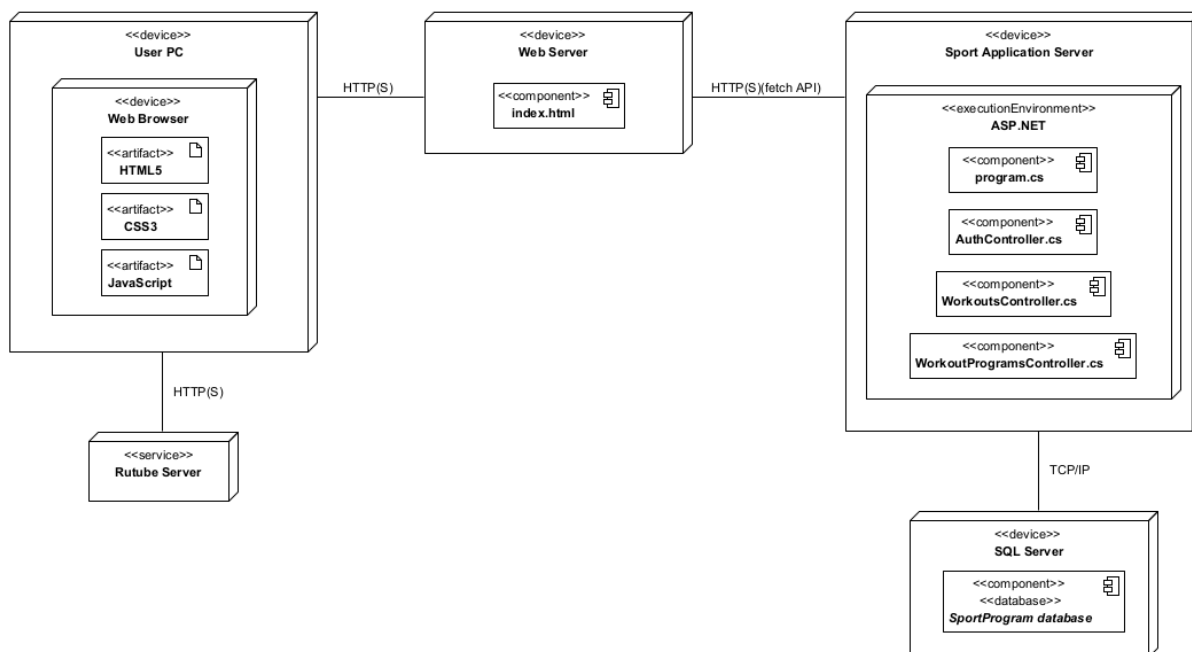


Рисунок 16 – Диаграмма развертывания

Клиентская часть приложения размещается на веб-сервере. Серверная часть приложения размещается на сервере с использованием ASP.NET Core. База данных может быть размещена как вместе с серверной частью, так и на отдельном сервере. Пользователь будет открывать клиентское приложение в браузере на своем устройстве. Клиентское приложение будет получать запросы пользователя и направлять их в серверное приложение через HTTP(S). Серверное приложение будет обрабатывать эти запросы, а также взаимодействовать с базой данных через сетевой протокол.

### 3.3 Реализация клиентской части приложения

Клиентская часть приложения для предоставления программ спортивных тренировок была реализована в среде программирования Apache NetBeans на языке JavaScript. Код клиентской части представлен в приложении Г.

Диаграмма компонентов клиентской части реализуемого приложения, показывающая взаимоотношения между ее составляющими, представлена на рисунке 17.

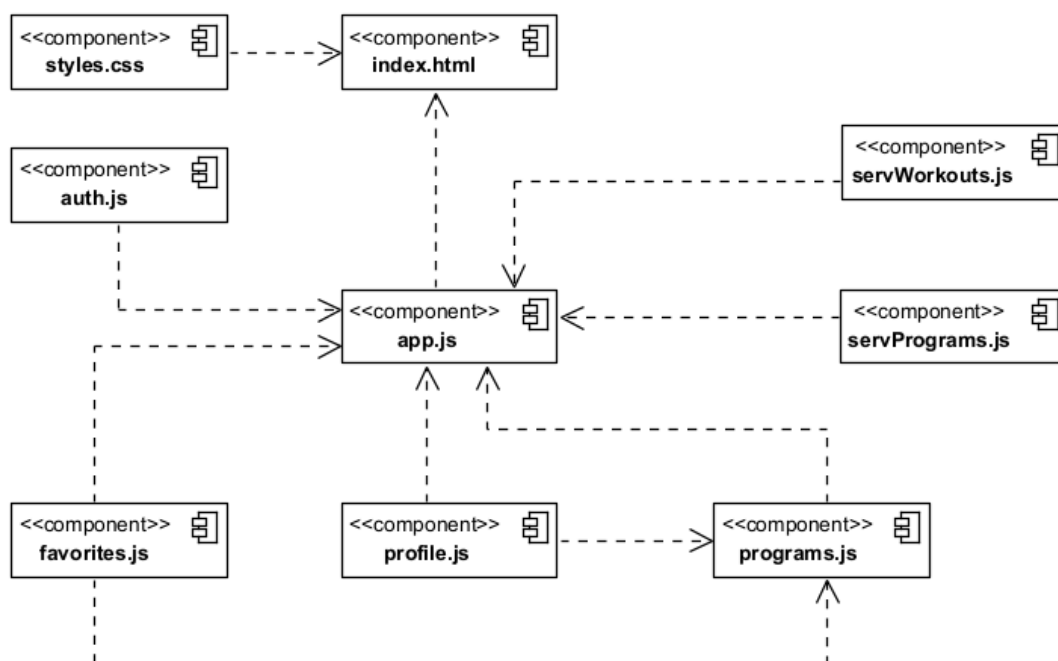


Рисунок 17 – Диаграмма компонентов клиентской части

Описание компонентов клиентской части содержится в таблице 5.

Таблица 5 – Описание компонентов клиентской части

| Компонент       | Описание                                                                         |
|-----------------|----------------------------------------------------------------------------------|
| index.html      | Входная точка клиентского приложения                                             |
| styles.css      | Компонент, содержащий стили элементов интерфейса                                 |
| app.js          | Компонент, реализующий переход между пунктами меню                               |
| auth.js         | Компонент, реализующий авторизацию и регистрацию пользователей на странице входа |
| profile.js      | Компонент, реализующий функционал страницы профиля пользователя                  |
| favorites.js    | Компонент, реализующий функционал страницы избранных тренировок                  |
| programs.js     | Компонент, реализующий функционал страницы с лентой программ тренировок          |
| servPrograms.js | Компонент, реализующий функционал страницы управления программами тренировок     |

Продолжение таблицы 5

| Компонент       | Описание                                                           |
|-----------------|--------------------------------------------------------------------|
| servWorkouts.js | Компонент, реализующий функционал страницы управления тренировками |

Для обмена данными между клиентской и серверной частью используется интерфейс Fetch API, который представляет собой актуальный и универсальный набор функций [10].

### 3.4 Реализация интерфейса приложения

В соответствии с спроектированным интерфейсом приложения было реализовано одностраничное web-приложение. При входе в приложение для пользователя будут доступны следующие разделами:

- раздел авторизации и регистрации;
- раздел личного кабинета;
- раздел ленты с программами тренировок;
- раздел избранных тренировок.

Для входа в приложение пользователь должен войти в свой аккаунт в разделе авторизации, представленном на рисунке 18.

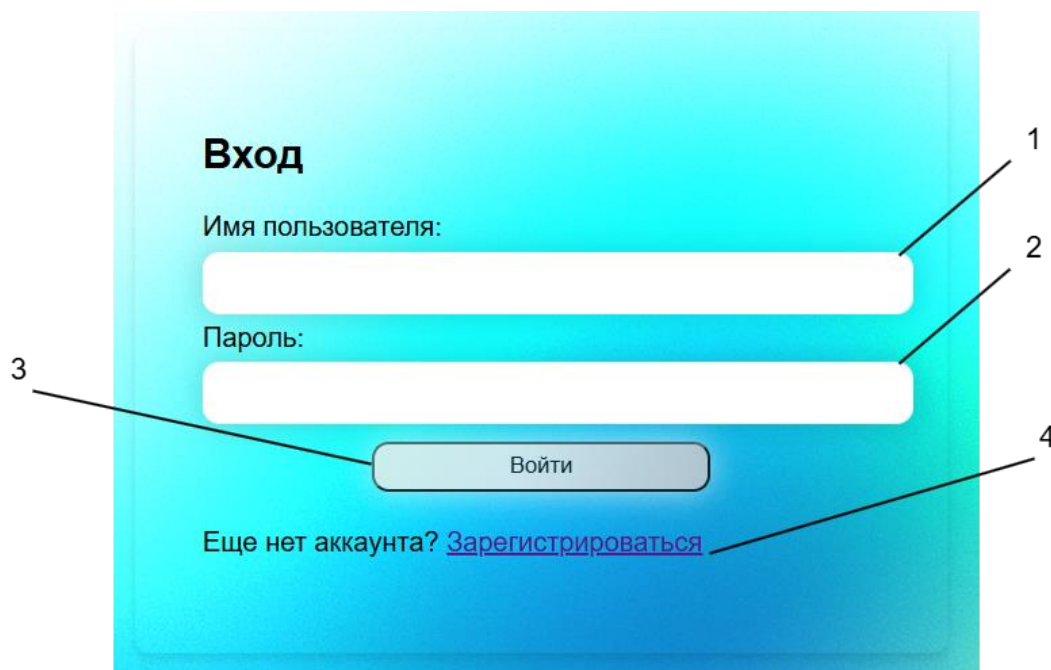


Рисунок 18 – Раздел авторизации

Раздел авторизации содержит следующие элементы:

- 1 – поле для ввода имени пользователя, то есть логина;
- 2 – поле для ввода пароля;
- 3 – кнопка входа;
- 4 – ссылка для перехода к регистрации.

Если пользователь нажмет на кнопку входа, приложение начнет проверку введенных данных от аккаунта. Если такой аккаунт существует и данные верны, приложение предоставит пользователю токен доступа и пользователь войдет в свой аккаунт.

Если пользователь еще не зарегистрирован, он может нажать на ссылку для регистрации, которая отобразит на странице раздел регистрации, показанный на рисунке 19.

The image shows a registration form on a light blue background. The title 'Регистрация' is at the top. Below it are two input fields: 'Имя пользователя:' and 'Пароль:'. A button labeled 'Зарегистрироваться' is below the password field. At the bottom, there is a link 'Войти' preceded by the text 'Уже есть аккаунт?'. Four numbered callouts point to specific elements: 1 points to the username input field, 2 points to the password input field, 3 points to the 'Зарегистрироваться' button, and 4 points to the 'Войти' link.

Рисунок 19 – Раздел регистрации

Раздел регистрации содержит следующие элементы:

- 1 – поле для ввода имени пользователя;
- 2 – поле для ввода пароля;

3 – кнопка регистрации;

4 – ссылка для перехода к разделу авторизации.

После нажатия кнопки регистрации приложение проверит введенные данные и наличие пользователя с таким именем. Если не будет обнаружено ошибок, пользователь будет зарегистрирован и возвращен к разделу авторизации.

После входа в приложение пользователь будет перенаправлен в раздел ленты программ тренировок и ему станет доступно меню. Раздел ленты программ тренировок показан на рисунке 20.

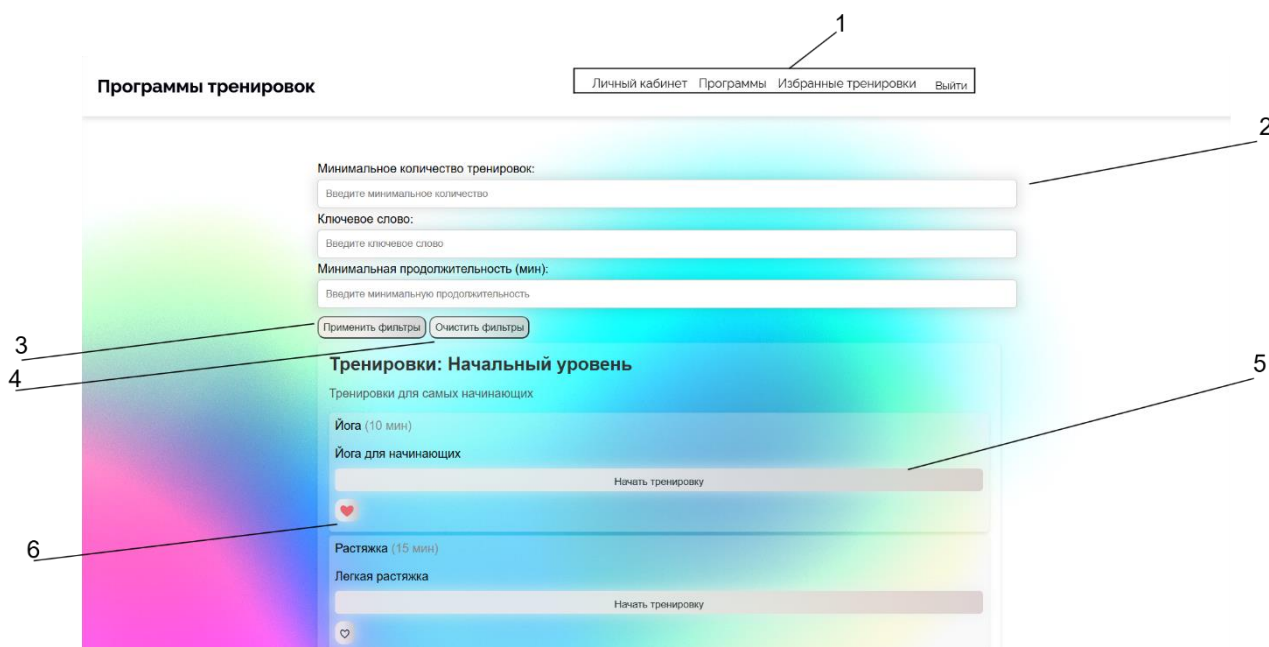


Рисунок 20 – Раздел ленты программ тренировок

В разделе содержатся следующие элементы:

1 – меню приложения;

2 – поля для ввода критериев фильтрации;

3 – кнопка для выполнения фильтрации;

4 – кнопка для очистки критериев фильтрации;

5 – кнопка для открытия тренировки;

6 – кнопка для добавления/удаления избранных тренировок.

При нажатии кнопки фильтрации будут отображены программы и тренировки, соответствующие введенным критериям. При нажатии кнопки очистки поля критериев станут пустыми и будут отображены все программы тренировок.

Если пользователь добавит тренировку в избранные, кнопка добавления/удаления будет окрашена в красный цвет, при удалении она станет прозрачной.

Если пользователь нажмет кнопку открытия тренировки, откроется модальное окно с видеоплеером, представленное на рисунке 21.

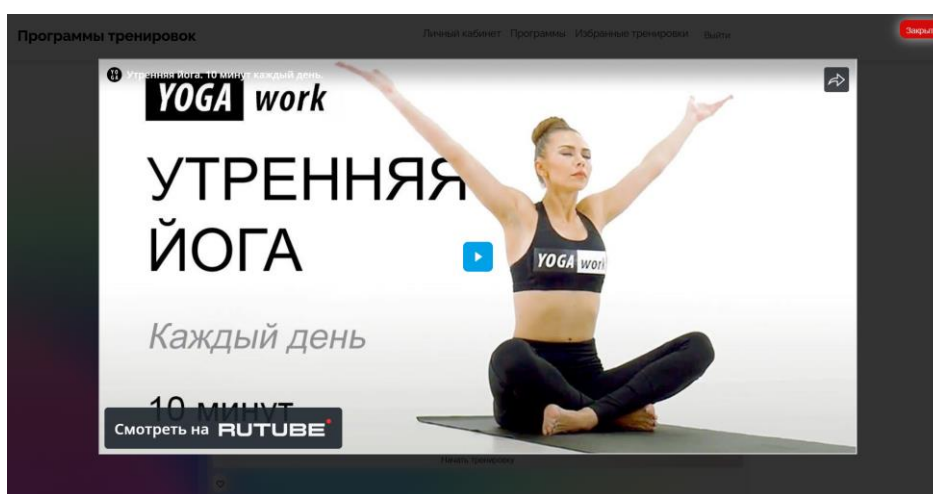


Рисунок 21 – Окно с видеоплеером

Из меню пользователь может перейти в раздел личного кабинета, представленного на рисунке 22.

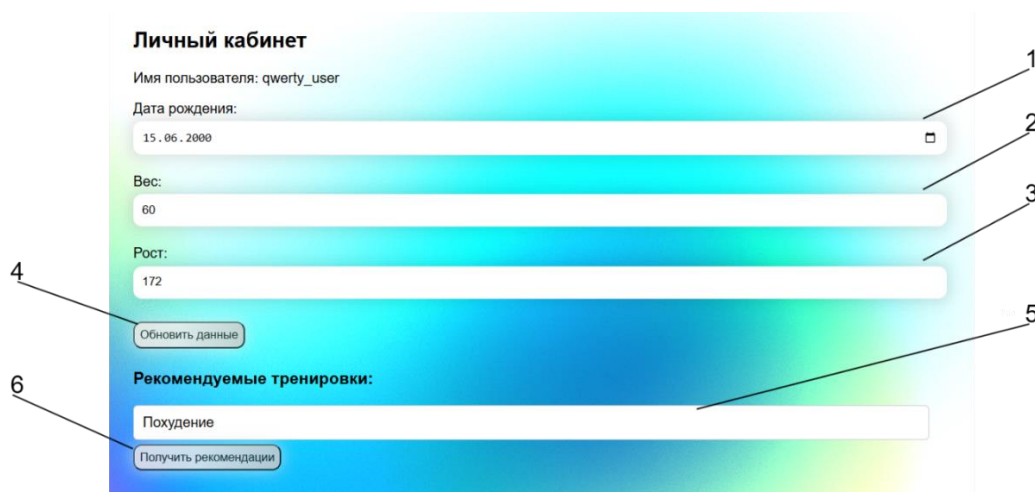


Рисунок 22 – Раздел личного кабинета

В разделе личного кабинета присутствуют следующие элементы:

- 1 – поле для ввода даты рождения;
- 2 – поле для ввода веса;
- 3 – поле для ввода роста;
- 4 – кнопка обновления данных пользователя;
- 5 – селектор выбора цели тренировок;
- 6 – кнопка для получения рекомендаций.

При нажатии кнопки обновления проверяются введенные данные, если они верны, то данные о пользователе обновляются.

Если пользователь выберет цель тренировок и нажмет кнопку для получения рекомендаций, будут отображены список рекомендуемых тренировок с количеством калорий, которые затратит пользователь при их выполнении. Также будет отображена диаграмма. Получение рекомендаций показано на рисунке 23.

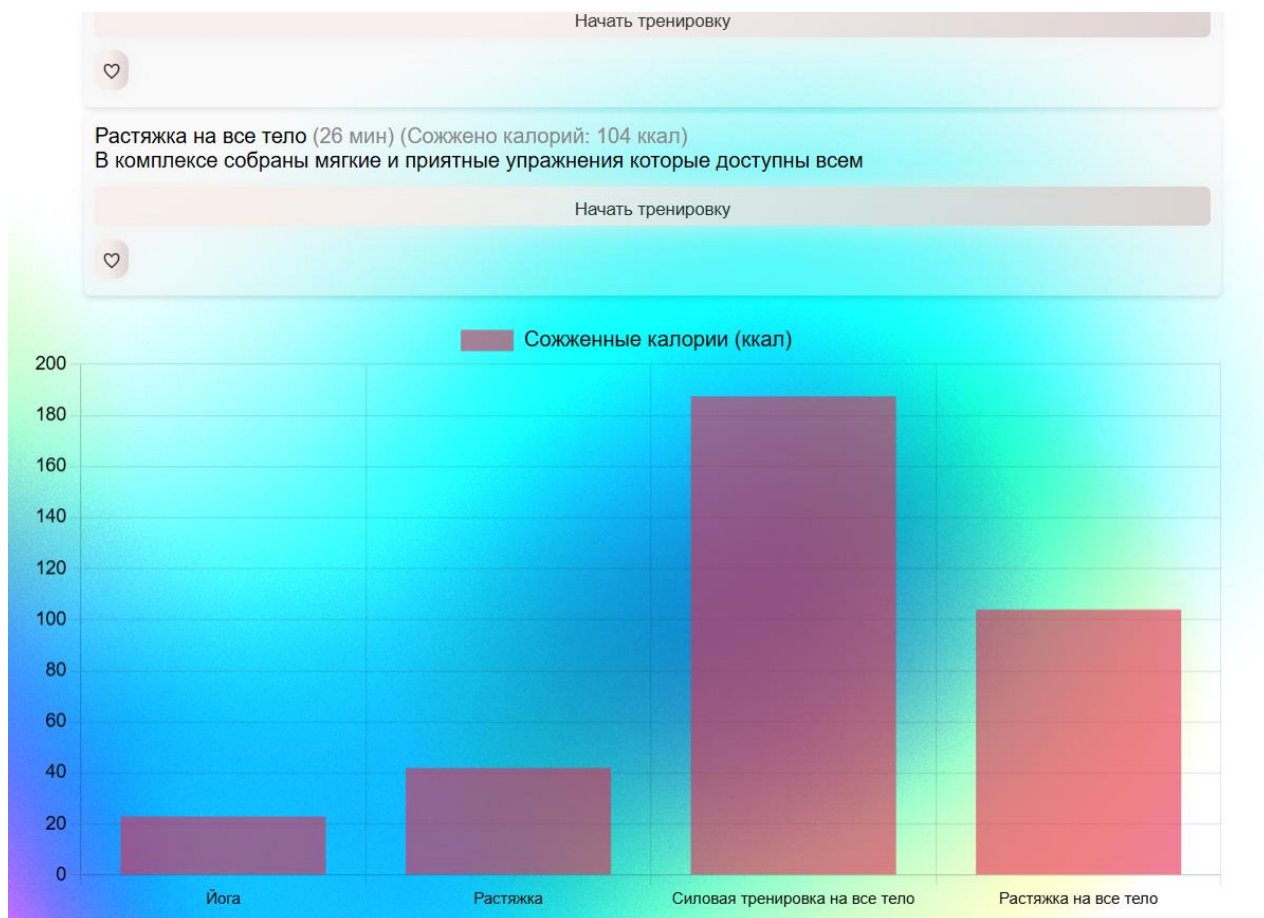


Рисунок 23 – Рекомендуемые тренировки

Также пользователь может перейти из меню в раздел избранных тренировок, показанный на рисунке 24.

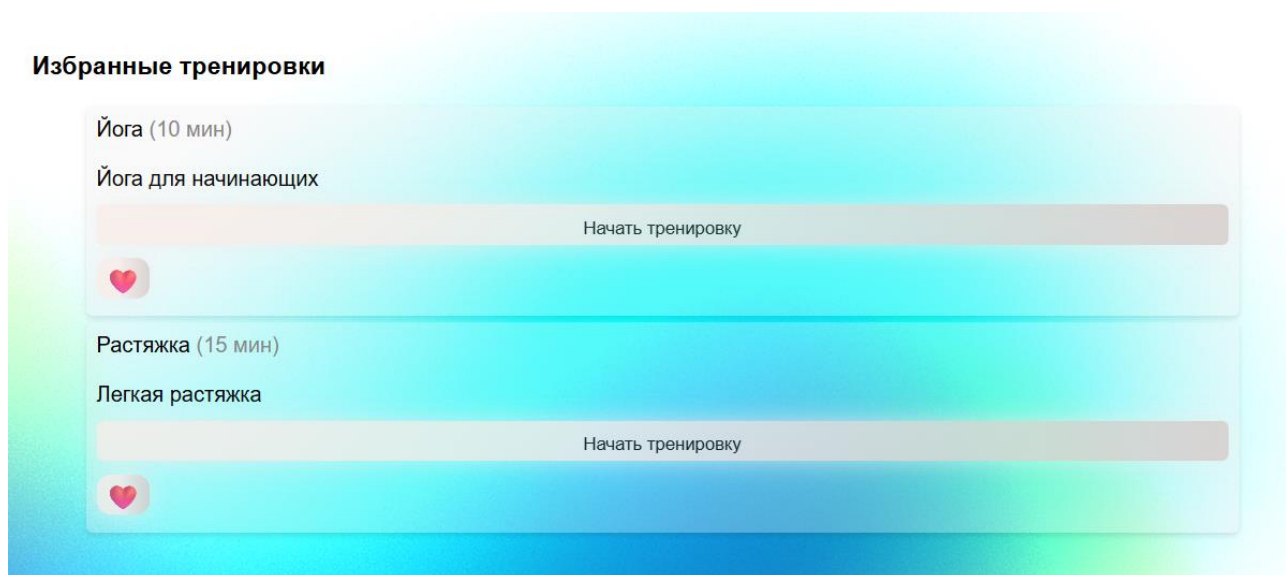


Рисунок 24 – Раздел избранных тренировок

В этом разделе отображены все тренировки, которые пользователь добавил в избранные. Как и в других разделах присутствуют кнопки для удаления тренировки из избранных и открытия тренировки.

Когда в приложение входит администратор, ему доступны разделы управления программами и управления тренировками. Раздел управления программами показан на рисунке 25.

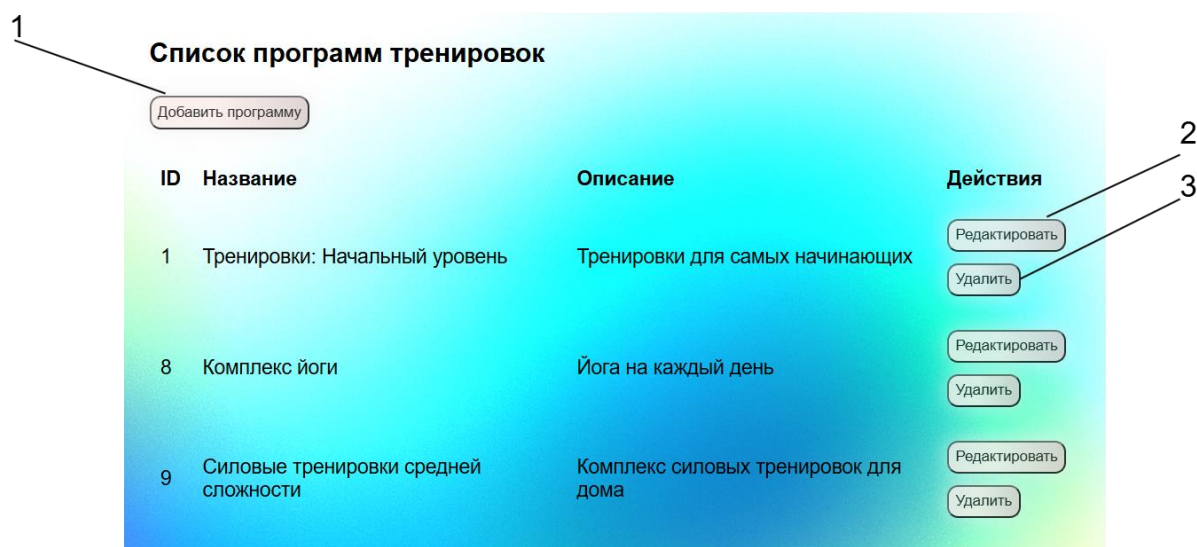


Рисунок 25 – Раздел управления программами

В разделе управления программами содержатся следующие элементы:

- 1 – кнопка добавления новой программы;
- 2 – кнопка редактирования программы;
- 3 – кнопка удаления программы.

При нажатии кнопки добавления открывается модальное окно, показанное на рисунке 26.

Рисунок 26 – Окно добавления программы

При нажатии кнопки «Сохранить» будут проверены введенные данные программы, если они верны, то программа будет добавлена.

При нажатии кнопки редактирования программы откроется модальное окно с заполненными данными изменяемой программы. Также в окне присутствуют селекторы для выбора тренировок, которые будут добавлены или удалены из программы. Окно представлено на рисунке 27.

**Редактировать программу**

Название:  
Тренировки: Начальный уровень

Описание:  
Тренировки для самых начинающих

**Добавить тренировки**

Выберите тренировки:

- Йога
- Растяжка
- Силовая тренировка на все тело
- Растяжка на все тело

Добавить выбранные тренировки

**Удалить тренировки**

Выберите тренировки:

- Йога
- Растяжка
- Силовая тренировка на все тело
- Растяжка на все тело

Удалить выбранные тренировки

Сохранить Отмена

Рисунок 27 – Окно редактирования программы

Раздел управления тренировками содержит аналогичные элементы. Раздел показан на рисунке 28.

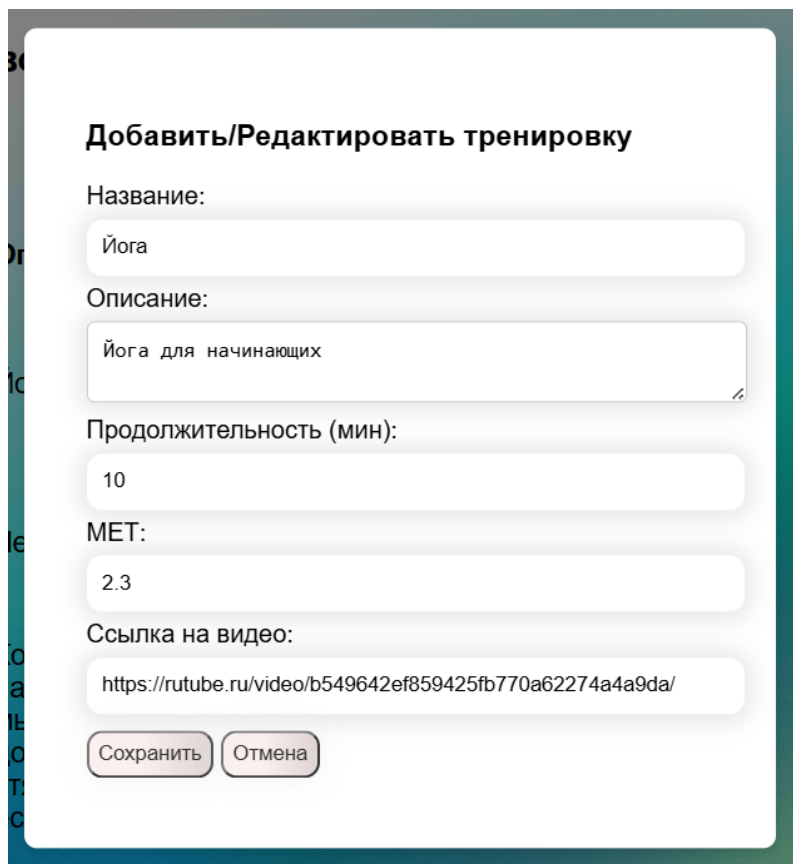
**Список тренировок**

Добавить тренировку

| ID | Название                       | Описание                                                                                                                                                          | Продолжительность (мин) | МЕТ | Действия                                    |
|----|--------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------|-----|---------------------------------------------|
| 1  | Йога                           | Йога для начинающих                                                                                                                                               | 10                      | 2.3 | <div>Редактировать</div> <div>Удалить</div> |
| 2  | Растяжка                       | Легкая растяжка                                                                                                                                                   | 15                      | 2.8 | <div>Редактировать</div> <div>Удалить</div> |
| 7  | Силовая тренировка на все тело | Комплекс силовых упражнений, направленный на проработку мышц всего тела. Вы можете добавить гантели, резинки или утяжелители к этому занятию, если это требуется. | 25                      | 7.5 | <div>Редактировать</div> <div>Удалить</div> |
| 8  | Растяжка на все тело           | В комплексе собраны мягкие и приятные упражнения которые доступны всем                                                                                            | 26                      | 4   | <div>Редактировать</div> <div>Удалить</div> |

Рисунок 28 – Раздел управления тренировками

Функционал кнопок аналогичен разделу управления программами, при нажатии откроется модальное окно, представленное на рисунке 29.



**Добавить/Редактировать тренировку**

Название:

Йога

Описание:

Йога для начинающих

Продолжительность (мин):

10

МЕТ:

2.3

Ссылка на видео:

<https://rutube.ru/video/b549642ef859425fb770a62274a4a9da/>

Сохранить Отмена

Рисунок 29 – Окно редактирования тренировки

Для построения диаграмм использовалась библиотека chart.js. Библиотека поддерживает множество видов диаграмм, что позволяет выбрать наиболее подходящее отображение данных в приложении [11].

## **4 Отладка и тестирование**

### **4.1 Обоснование выбора режима тестирования**

Задача тестирования – проанализировать приложение с целью обнаружения ошибок и повышения качества. Классификация тестирования позволяет организовать знания и способствует более быстрому планированию и разработке тест-кейсов [12]. Существует множество видов и направлений тестирования. Наибольшей известностью пользуются следующие виды:

- функциональное тестирование;
- модульное тестирование;
- интеграционное тестирование;
- системное тестирование.

Результаты функционального тестирования показывают, соответствует ли функционал программного обеспечения заявленным требованиям. Такое тестирование может включать себя проверку интерфейсов, клиент-серверных приложений и общей работы программного обеспечения [13].

Модульное тестирование заключается в проверке работоспособности отдельных модулей или частей программного обеспечения. Такое тестирование помогает быстро исправить ошибки при добавлении нового функционала в приложение, при этом не тестирую программное обеспечение полностью [14].

Интеграционное тестирование же наоборот направлено на проверку работы нескольких частей программного обеспечения, их взаимодействия.

Системным называется тестирование полного программного продукта. Проверяется соответствие требованиям всего программного обеспечения. Такое тестирование проводится уже после завершения интеграционного [15].

Было принято решение о проведении функционального и модульного тестирования разработанного приложения, что позволит оценить работу приложения со стороны пользователя, а также оценить соответствие требованиям отдельных компонентов.

## 4.2 Тестирование приложения для предоставления программ спортивных тренировок

### 4.2.1 Функциональное тестирование

Было проведено функциональное тестирование методом черного ящика на примере наименования программ тренировок. Для этого были выделены классы эквивалентности данного поля, которые представлены в таблице 6.

Таблица 6 – Классы эквивалентности

| Требования         | Правильные КЭ                                               | Неправильные КЭ          |
|--------------------|-------------------------------------------------------------|--------------------------|
| Длина наименования | 2 – 255(1)                                                  | 0 – 1(2)<br>256 – ∞(3)   |
| Структура          | Начинается с буквы(4)                                       | Начинается не с буквы(5) |
| Содержание         | Кириллические буквы(6)<br>Цифры(7)<br>Дефисы(8)<br>Точки(9) | Остальное(10)            |

В соответствии с выделенными классами эквивалентности были разработаны тест-кейсы, представленные в таблице 7.

Таблица 7 – Тест-кейсы

| № | КЭ               | Входные данные                                                                                                                                                                                                                                                                                    | Выходные данные                                      |
|---|------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------|
| 1 | 1, 4, 6, 7, 8, 9 | Йога-лайт 2.0                                                                                                                                                                                                                                                                                     | Успешно                                              |
| 2 | 2, 4, 6          | Й                                                                                                                                                                                                                                                                                                 | Ошибка: Длина названия меньше 2 (рисунок Д.1)        |
| 3 | 3, 4, 6          | Йогадлнддънрьлилдвпдв<br>Аропоыаподжопдтпдлоп<br>Опвпарпвроврпмвыраовраор<br>Выоароыраоыраоряорывоары<br>Овпроврыпорвпороярвоыраво<br>Ыраоврыоарыворапвапвоыраор<br>Впрвпрораоарвоаровраоыврарв<br>Аывопарововыаорыаорвльоарво<br>Раывыпавиарлоаривравравроарв<br>ыолрарвепавпавпваппавпавапааирр | Ошибка: Слишком длинное название (рисунок Д.2)       |
| 4 | 1, 5, 6, 9       | .Теннис                                                                                                                                                                                                                                                                                           | Ошибка: Название начинается не с буквы (рисунок Д.3) |

Продолжение таблицы 7

| №  | КЭ            | Входные данные  | Выходные данные                                              |
|----|---------------|-----------------|--------------------------------------------------------------|
| 5  | 1, 4, 6, 10   | Фитнес@         | Ошибка: Название содержит недопустимые символы (рисунок Д.4) |
| 6  | 1, 5, 7       | 3215            | Ошибка: Название должно содержать буквы (рисунок Д.5)        |
| 7  | 1, 5, 8       | ---             | Ошибка: Название должно содержать буквы                      |
| 8  | 1, 5, 9       | ...             | Ошибка: Название должно содержать буквы                      |
| 9  | 1, 5, 7, 8, 9 | 35 – 45.        | Ошибка: Название должно содержать буквы                      |
| 10 | 1, 4, 6, 7    | Пилатес2        | Успешно                                                      |
| 11 | 1, 4, 6, 8    | Йо-йога         | Успешно                                                      |
| 12 | 1, 4, 6, 9    | Сил. тренировка | Успешно                                                      |

Полученные результаты тестирования соответствуют требованиям, что говорит о корректной работе клиентской части приложения.

#### 4.2.2 Модульное тестирование

В ходе выполнения модульного тестирования был разработан тестовый проект с использованием библиотеки тестов xUnit [16] и средств среды разработки Microsoft Visual Studio. Код разработанных Unit-тестов приведен в приложении Е. План модульного тестирования приведен в таблице 8.

Таблица 8 – План модульного тестирования

| Тест       | Описание                                                       | Ожидаемый результат | Актуальный результат |
|------------|----------------------------------------------------------------|---------------------|----------------------|
| RegisterOk | Проверка правильной работы выполнения регистрации пользователя | Пройден             | Пройден              |

Продолжение таблицы 8

| Тест                            | Описание                                                                                                                      | Ожидаемый результат | Актуальный результат |
|---------------------------------|-------------------------------------------------------------------------------------------------------------------------------|---------------------|----------------------|
| LoginReturnToken                | Проверка правильной работы авторизации пользователя, в результате которой пользователю должен быть предоставлен токен доступа | Пройден             | Пройден              |
| UpdateOk                        | Проверка правильной работы выполнения обновления данных пользователя                                                          | Пройден             | Пройден              |
| GetProfile_ReturnsOk            | Проверка правильной работы возвращения данных пользователя                                                                    | Пройден             | Пройден              |
| AddFavoriteWorkout_ReturnsOk    | Проверка правильной работы добавления тренировки в избранные пользователя                                                     | Пройден             | Пройден              |
| RemoveFavoriteWorkout_ReturnsOk | Проверка правильной работы удаления тренировки из избранных пользователя                                                      | Пройден             | Пройден              |

На рисунке 30 приведены результаты выполнения модульного тестирования.

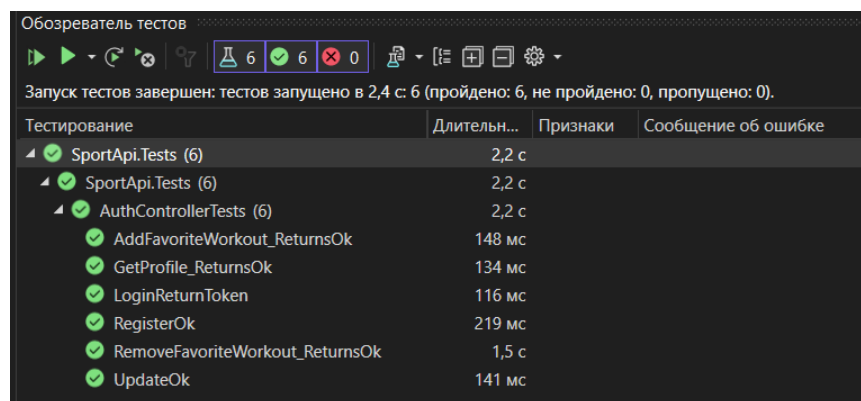


Рисунок 30 – Результаты выполнения модульных тестов

В ходе выполнения модульного тестирования ошибок в коде серверной части не было обнаружено. Из полученных результатов можно сделать вывод о корректной работе методов, так как тесты были успешно пройдены.

## **5 Контроль качества**

### **5.1 Обоснование выбора метрик кода**

Метрики кода используют для оценки качества программного обеспечения. Использование метрик дает понимание о сложности программного продукта, об объеме проделанной работы и о усилиях разработчика, приложенных в процессе разработки [17].

Для оценки качество приложения для предоставления программ спортивных тренировок будут использоваться одни из известных метрики Холстеда и Чепина.

Метрика Холстеда вычисляется на основе анализа числа строк и элементов исходного кода. Она позволяет оценить размер и объем программы, также с ее помощью можно оценить время разработки [18].

Метрика Чепина позволяет оценить качество отдельных модулей программы с помощью анализа переменных списка ввода-вывода [19].

### **5.2 Оценка качества программного кода**

#### **5.2.1 Метрика Холстеда**

Метрики Холстеда были посчитаны для метода `AddToFav` класса `UserRepository`. Этот метод реализует добавление пользователем тренировки в избранные. Код метода представлен в листинге В.17 приложения В.

Для расчета метрик Холстеда необходимо посчитать 4 характеристики программного кода: словарь программы ( $n$ ), длина программы ( $N$ ), объем программы ( $V$ ), сложность программы ( $D$ ). Словарь программы состоит из суммы числа вхождений простых операторов ( $n_1$ ) и числа вхождений простых операндов ( $n_2$ ). Длина программы состоит из суммы общего количества простых операторов ( $N_1$ ) и общего количества простых операндов ( $N_2$ ).

Были рассчитаны сумма вхождений и общее количество операторов и операндов для метода добавления тренировки в избранные. Расчеты искомых

значений приведены в таблице 9.

Таблица 9 – Расчеты значений

| №  | Операторы      | Число операторов | Операнды              | Число операндов |
|----|----------------|------------------|-----------------------|-----------------|
| 1  | public         | 1                | AddToFav              | 1               |
| 2  | void           | 1                | userId                | 2               |
| 3  | (              | 8                | workoutId             | 2               |
| 4  | )              | 8                | user                  | 4               |
| 5  | int            | 2                | _context.Users        | 1               |
| 6  | var            | 2                | u                     | 4               |
| 7  | =              | 9                | u.FavoriteWorkouts    | 1               |
| 8  | .              | 11               | u.Id                  | 1               |
| 9  | Include        | 1                | workout               | 4               |
| 10 | FirstOrDefault | 1                | _context.Workouts     | 1               |
| 11 | >              | 2                | user.FavoriteWorkouts | 2               |
| 12 | ;              | 4                |                       |                 |
| 13 | if             | 2                |                       |                 |
| 14 | return         | 1                |                       |                 |
| 15 | null           | 2                |                       |                 |
| 16 | Find           | 1                |                       |                 |
| 17 | !              | 2                |                       |                 |
| 18 | &&             | 1                |                       |                 |
| 19 | Contains       | 1                |                       |                 |
| 20 | {              | 2                |                       |                 |
| 21 | }              | 2                |                       |                 |
| 22 | Add            | 1                |                       |                 |
|    | Итого:         | 65               | Итого:                | 23              |

Были рассчитаны характеристики метрик Холстеда с использованием полученных значений. Результаты расчета приведены в таблице 10.

Таблица 10 – Метрики Холстеда

| Характеристика               | Значение |
|------------------------------|----------|
| Словарь (n)                  | 33       |
| Длина (N)                    | 88       |
| Объем (V)                    | 443,9    |
| Вычислительная сложность (D) | 23       |
| Усилия (H)                   | 10209,7  |

Вычисленные метрики Холстеда используются для расчета комплексного показателя качества кода. Показатель рассчитывается по следующей формуле:

$$MI = \text{MAX}(0, (171 - 5.2 * \ln(V) - 0.23 * CC - 16.2 * \ln(\text{LoC})) * 100 / 171),$$

где  $V$  – объем кода Холстеда;

$CC$  – цикломатическая сложность кода, являющаяся суммой точек принятия решения и базового пути;

$\text{LoC}$  – количество строк кода.

Цикломатическая сложность ( $CC$ ) метода равна 4, а количество строк кода ( $\text{LoC}$ ) равно 14. Таким образом комплексный показатель качества кода ( $MI$ ) равен 56, это означает, что код относительно удобно поддерживать.

### 5.2.2 Метрика Чепина

Метрика Чепина будет рассчитана также для метода `AddToFav` класса `UserRepository`, код которого находится в листинге 17 в приложении Д.

Для расчета этой метрики множество переменных списка ввода-вывода разбиваются на четыре группы по функциональности:

- переменные, которые не изменяются, а служат как контейнеры для данных ( $P$ );
- переменные, изменяемые в процессе выполнения программы ( $M$ );
- переменные, управляющие потоком программы ( $C$ );
- переменные, которые не используются и не играют принципиальной роли ( $T$ ).

Особенностью использования этой метрики является необходимость учета переменных в каждой группе, поскольку они могут выполнять несколько задач.

Формула расчета метрики Чепина записывается в следующем виде:

$$Q = a_1 * P + a_2 * M + a_3 * C + a_4 * T,$$

где  $a_1, a_2, a_3, a_4$ , – весовые коэффициенты.

Значения коэффициентов связаны с влиянием, которая оказывает каждая группа переменных на сложность программы. Наибольшее влияние на программу оказывает группа переменных  $C$ , поэтому ее коэффициент равен

$a_3=3$ . Остальные коэффициенты имеют значения:  $a_1=1$ ;  $a_2=2$ ;  $a_4=0.5$ . Несмотря на то, что группа переменных Т не влияет на сложность программы, ее весовой коэффициент больше нуля, так как она может затруднять чтение программы.

Распределение переменных по группам показано в таблице 11.

Таблица 11 – Метрика Чепина

| Переменные          | Полная метрика Чепина                                               |                                            |                                            |   |
|---------------------|---------------------------------------------------------------------|--------------------------------------------|--------------------------------------------|---|
| Группы переменных   | P                                                                   | M                                          | C                                          | T |
| Переменные в группе | userId,<br>workoutId,<br>u,<br>_context.Users,<br>_context.Workouts | user,<br>workout,<br>user.FavoriteWorkouts | user,<br>workout,<br>user.FavoriteWorkouts | – |
| Значения переменных | 5                                                                   | 3                                          | 3                                          | 0 |
| Значение метрики    | Q = 20                                                              |                                            |                                            |   |

Полученное значение метрики говорит об умеренной сложности кода метода, что является хорошим результатом.

## 6 Расчет бюджета разработки

### 6.1 Планирование разработки и оценка бюджета

Для планирования и визуализации управления задачами разработки приложения для предоставления программ спортивных тренировок была использована диаграмма Ганта [20]. Диаграмма, представленная на рисунке 31, была построена в программе ProjectLibre.

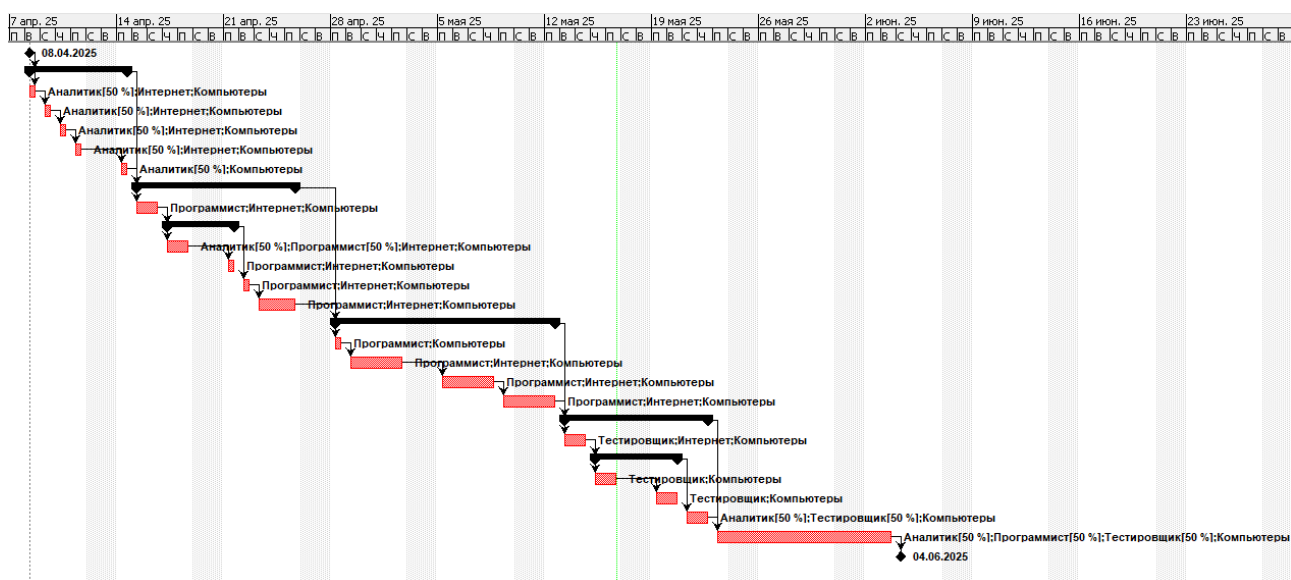


Рисунок 31 – Диаграмма Ганта

За дату начала разработки было взято 8 апреля 2025, дата окончания – 4 июня. Общая продолжительность разработки составила 41 день.

Чтобы оценить бюджет разработки, были назначены ресурсы и стоимость их использования. Ресурсы показаны на рисунке 32.

| Иконка      | Название    | Тип      | Стандартная ставка | Ставка сверхурочных |
|-------------|-------------|----------|--------------------|---------------------|
| Аналитик    | Аналитик    | Работа   | 500/час            | 750/час             |
| Программист | Программист | Работа   | 450/час            | 675/час             |
| Тестировщик | Тестировщик | Работа   | 320/час            | 480/час             |
| Интернет    | Интернет    | Материал | 0                  |                     |
| Компьютеры  | Компьютеры  | Материал | 0                  |                     |

Рисунок 32 – Данные о ресурсах

Всего было выделено 27 задач, описание и сроки выполнения которых представлены на рисунке 33.

| Название                                                                            | Продолжи...    | Начало                   | Окончание                |
|-------------------------------------------------------------------------------------|----------------|--------------------------|--------------------------|
| Начало проекта                                                                      | 0 дней         | 08.04.2025, 08:00        | 08.04.2025, 08:00        |
| <b>Постановка задачи и анализ требований</b>                                        | <b>5 дней</b>  | <b>08.04.2025, 08:00</b> | <b>14.04.2025, 17:00</b> |
| Постановка задачи                                                                   | 1 день         | 08.04.2025, 08:00        | 08.04.2025, 17:00        |
| Анализ систем аналогичного назначения                                               | 1 день         | 09.04.2025, 08:00        | 09.04.2025, 17:00        |
| Формулировка требований                                                             | 1 день         | 10.04.2025, 08:00        | 10.04.2025, 17:00        |
| Выбор и обоснование технологий                                                      | 1 день         | 11.04.2025, 08:00        | 11.04.2025, 17:00        |
| Планирование разработки и оценка бюджета                                            | 1 день         | 14.04.2025, 08:00        | 14.04.2025, 17:00        |
| <b>Проектирование приложения для предоставления программ спортивных тренировок</b>  | <b>9 дней</b>  | <b>15.04.2025, 08:00</b> | <b>25.04.2025, 17:00</b> |
| Архитектура приложения                                                              | 2 дней         | 15.04.2025, 08:00        | 16.04.2025, 17:00        |
| <b>Проектирование структур данных</b>                                               | <b>3 дней</b>  | <b>17.04.2025, 08:00</b> | <b>21.04.2025, 17:00</b> |
| Концептуальная модель базы данных                                                   | 2 дней         | 17.04.2025, 08:00        | 18.04.2025, 17:00        |
| Логическая модель базы данных                                                       | 1 день         | 21.04.2025, 08:00        | 21.04.2025, 17:00        |
| Проектирование пользовательского интерфейса                                         | 1 день         | 22.04.2025, 08:00        | 22.04.2025, 17:00        |
| Проектирование классов приложения для предоставления программ спортивных тренировок | 3 дней         | 23.04.2025, 08:00        | 25.04.2025, 17:00        |
| <b>Реализация приложения для предоставления программ спортивных тренировок</b>      | <b>11 дней</b> | <b>28.04.2025, 08:00</b> | <b>12.05.2025, 17:00</b> |
| Реализация базы данных                                                              | 1 день         | 28.04.2025, 08:00        | 28.04.2025, 17:00        |
| Реализация серверной части приложения                                               | 4 дней         | 29.04.2025, 08:00        | 02.05.2025, 17:00        |
| Реализация клиентской части приложения                                              | 4 дней         | 05.05.2025, 08:00        | 08.05.2025, 17:00        |
| Реализация интерфейса приложения                                                    | 2 дней         | 09.05.2025, 08:00        | 12.05.2025, 17:00        |
| <b>Контроль качества приложения</b>                                                 | <b>8 дней</b>  | <b>13.05.2025, 08:00</b> | <b>22.05.2025, 17:00</b> |
| Обоснование выбора режима тестирования и метрик кода                                | 2 дней         | 13.05.2025, 08:00        | 14.05.2025, 17:00        |
| <b>Тестирование приложения для предоставления программ спортивных тренировок</b>    | <b>4 дней</b>  | <b>15.05.2025, 08:00</b> | <b>20.05.2025, 17:00</b> |
| Функциональное тестирование                                                         | 2 дней         | 15.05.2025, 08:00        | 16.05.2025, 17:00        |
| Модульное тестирование                                                              | 2 дней         | 19.05.2025, 08:00        | 20.05.2025, 17:00        |
| Оценка качества программного кода                                                   | 2 дней         | 21.05.2025, 08:00        | 22.05.2025, 17:00        |
| Оформление пояснительной записки                                                    | 8 дней         | 23.05.2025, 08:00        | 03.06.2025, 17:00        |
| Конец проекта                                                                       | 0 дней         | 04.06.2025, 08:00        | 04.06.2025, 08:00        |

Рисунок 33 – Планирование разработки

Оценка денежных затрат по задачам разработки представлена на рисунке 34.

| Название                                                                            | Оставшаяся стоимость |
|-------------------------------------------------------------------------------------|----------------------|
| Начало проекта                                                                      | 0,00 Р               |
| <b>Постановка задачи и анализ требований</b>                                        | <b>10000,00 Р</b>    |
| Постановка задачи                                                                   | 2000,00 Р            |
| Анализ систем аналогичного назначения                                               | 2000,00 Р            |
| Формулировка требований                                                             | 2000,00 Р            |
| Выбор и обоснование технологий                                                      | 2000,00 Р            |
| Планирование разработки и оценка бюджета                                            | 2000,00 Р            |
| <b>Проектирование приложения для предоставления программ спортивных тренировок</b>  | <b>32800,00 Р</b>    |
| Архитектура приложения                                                              | 7200,00 Р            |
| <b>Проектирование структур данных</b>                                               | <b>11200,00 Р</b>    |
| Концептуальная модель базы данных                                                   | 7600,00 Р            |
| Логическая модель базы данных                                                       | 3600,00 Р            |
| Проектирование пользовательского интерфейса                                         | 3600,00 Р            |
| Проектирование классов приложения для предоставления программ спортивных тренировок | 10800,00 Р           |
| <b>Реализация приложения для предоставления программ спортивных тренировок</b>      | <b>39600,00 Р</b>    |
| Реализация базы данных                                                              | 3600,00 Р            |
| Реализация серверной части приложения                                               | 14400,00 Р           |
| Реализация клиентской части приложения                                              | 14400,00 Р           |
| Реализация интерфейса приложения                                                    | 7200,00 Р            |
| <b>Контроль качества приложения</b>                                                 | <b>21920,00 Р</b>    |
| Обоснование выбора режима тестирования и метрик кода                                | 5120,00 Р            |
| <b>Тестирование приложения для предоставления программ спортивных тренировок</b>    | <b>10240,00 Р</b>    |
| Функциональное тестирование                                                         | 5120,00 Р            |
| Модульное тестирование                                                              | 5120,00 Р            |
| Оценка качества программного кода                                                   | 6560,00 Р            |
| Оформление пояснительной записки                                                    | 10160,00 Р           |
| Конец проекта                                                                       | 0,00 Р               |

Рисунок 34 – Оценка бюджета

По результатам оценки бюджеты общая сумма денежных затрат на разработку составит 114480 рублей.

## Заключение

В ходе выполнения выпускной квалификационной работы было разработано и реализовано приложение, предназначенное для предоставления программ спортивных тренировок.

На этапе анализа требований было проведено исследование предметной области, выполнен сравнительный анализ систем-аналогов, также были определены функциональные и нефункциональные требования к разрабатываемому приложению.

Было проведено планирование и оценка бюджета разработки приложения. В результате выполненных расчетов предполагаемые расходы на реализацию приложения составили 114480 рублей, а общая продолжительность всех этапов разработки составила 41 день.

В результате проведения этапа проектирования были разработаны концептуальная и логическая модели базы данных, определена архитектура приложения, выделены классы серверной части и проанализированы требования к интерфейсу.

Разработка приложения для предоставления программ спортивных тренировок включила реализацию базы данных в Microsoft SQL Server, создание серверного приложения в монолитном архитектурном стиле с использованием ASP.NET Core в среде Microsoft Visual Studio, а также разработку клиентского приложения в среде Apache NetBeans.

Этап отладки и тестирования приложения включил проведение функционального и модульного тестирования. Модульное тестирование было проведено с использованием библиотеки xUnit в среде Microsoft Visual Studio. Результаты тестирования подтвердили корректную работу приложения и его соответствие функциональным требованиям.

На стадии контроля качества был произведен расчет метрик Холстеда, Чепина и комплексного показателя качества кода, которые продемонстрировали надлежащий уровень качества разработанного приложения.

Таким образом, главная цель выпускной квалификационной работы – разработка приложения для предоставления программ спортивных тренировок – была успешно достигнута, и все поставленные задачи выполнены. Разработанное приложение демонстрирует высокий уровень качества и полностью соответствует предъявленным требованиям.

### Список использованных источников

1. Оразгелдиева О., Хандурдыев Б. Роль физической активности в формировании здорового образа жизни // Наука и мировоззрение. 2025. №38. [Электронный ресурс]. URL: <https://cyberleninka.ru/article/n/rol-fizicheskoy-aktivnosti-v-formirovanii-zdorovogo-obraza-zhizni> (дата обращения: 10.04.2025);
2. Фаулер М. UML. Основы, 3-е издание. – Пер. с англ. – СПб: Символ-Плюс, 2004. – 192 с.;
3. Экскурс в REST API // Systems Education. [Электронный ресурс]. URL: <https://systems.education/rest-api-postman> (дата обращения: 17.04.2025);
4. Ричардс М., Форд Н. Фундаментальный подход к программной архитектуре: паттерны, свойства, проверенные методы. – СПб.: Питер, 2023. – 448 с.;
5. Коннолли Т., Бегг К. Базы данных. Проектирование, реализация и сопровождение. Теория и практика. 3-е издание. : Пер. с англ. – М. : Издательский дом «Вильямс», 2003. – 1440 с.;
6. Кренке, Д. М. Теория и практика построения баз данных. 8-е изд. / Д. Крэнке. – СПб.: Питер, 2003. – 800 с.;
7. Миковски, М. С. Разработка одностраничных веб-приложений / М. С. Миковски, Дж. К. Пауэлл ; пер. с англ. А. А. Слинкина. – 2-е изд., эл. – Москва : ДМК Пресс, 2023. – 514 с.;
8. Обзор миграций – EF Core // Microsoft Learn [Электронный ресурс]. URL: <https://learn.microsoft.com/ru-ru/ef/core/managing-schemas/migrations/?tabs=dotnet-core-cli> (дата обращения: 6.05.2025);
9. Compendium of Physical Activities // Journal of Sport and Health Science [Электронный ресурс]. URL: <https://racompendium.com/> (дата обращения: 15.05.2025);
10. Fetch API – Интерфейсы веб API // MDN [Электронный ресурс]. URL: [https://developer.mozilla.org/ru/docs/Web/API/Fetch\\_API](https://developer.mozilla.org/ru/docs/Web/API/Fetch_API) (дата обращения: 18.05.2025);

11. Chart.js и Canvas обзор инструментов для визуализации // Uplab [Электронный ресурс]. URL: <https://www.uplab.ru/blog/chart-js-i-canvas-obzor-instrumentov-dlya-vizualizacii/> (дата обращения: 19.05.2025);
12. Куликов, С. С. Тестирование программного обеспечения. Базовый курс / С. С. Куликов. – 3-е изд. – Минск: Четыре четверти, 2020. – 312 с.;
13. Функциональное тестирование: основы, виды и применение // Prog Academy [Электронный ресурс]. URL: <https://prog.academy.ru/blog/functional-testing> (дата обращения: 22.05.2025);
14. Unit-тестирование(модульное): что это и для чего используется // Skillfactory [Электронный ресурс]. URL: <https://blog.skillfactory.ru/glossary/unit-testirovanie/> (дата обращения: 22.05.2025);
15. Что такое системное тестирование, зачем оно нужно и как его проводить? // Хабр [Электронный ресурс]. URL: <https://habr.com/ru/articles/807813/> (дата обращения: 22.05.2025);
16. Модульное тестирование C# в .NET с помощью dotnet test и xUnit // Microsoft Learn [Электронный ресурс]. URL: <https://learn.microsoft.com/ru-ru/dotnet/core/testing/unit-testing-csharp-with-xunit> (дата обращения: 28.05.2025);
17. Программный код и его метрики // Хабр [Электронный ресурс]. URL: <https://habr.com/ru/companies/intel/articles/106082/> (дата обращения: 23.05.2025);
18. Аверьянов А. В., Кошель И. Н., Кузнецов В. В. Применение метрик Холстеда для количественного оценивания характеристик программ ЭВМ // Приборостроение. 2019. №11. [Электронный ресурс]. URL: <https://cyberleninka.ru/article/n/primenenie-metrik-holsteda-dlya-kolichestvennogo-otsenivaniya-harakteristik-programm-evm> (дата обращения: 25.05.2025);
19. Евсеев В. В. Применение программных метрик кода на раннем этапе жизненного цикла программного обеспечения // ВЕЖПТ. 2011. №2 (49). [Электронный ресурс]. URL: <https://cyberleninka.ru/article/n/primenenie-programmnyh-metrik-koda-na-rannem-etape-zhiznennogo-tsikla-programmnogo-obespecheniya> (дата обращения: 25.05.2025);

20. Что такое диаграмма Ганта и как её построить // Яндекс Практикум [Электронный ресурс]. URL: <https://practicum.yandex.ru/blog/diagramma-ganta/> (дата обращения: 22.04.2025).

**Приложение А**  
**(справочное)**  
**Глоссарий**

БД (База данных) – это имеющая название совокупность данных, которая отражает состояние объектов и их отношений в рассматриваемой предметной области.

MET – метаболический эквивалент, оценивается как энергопотребление данной деятельности, деленное на энергопотребление покоя: 1 MET соответствует 3,5 мл O<sub>2</sub>/кг/мин.

СУБД (Система управления базами данных) – это совокупность программных и лингвистических средств общего или специального назначения, обеспечивающих управление созданием и использованием баз данных.

API (Application Programming Interface) – набор готовых классов, функций, процедур, констант и структур, предоставляемых приложением или операционной системой для использования во внешних программных продуктах.

REST (Representational State Transfer) – это архитектурная концепция, определяющая набор ограничений для построения масштабируемых распределенных систем. Основными принципами REST является ресурсно-ориентированная архитектура, использование стандартных HTTP-методов для манипуляции ресурсами, передача данных без сохранения состояния на сервере, унифицированный интерфейс доступа.

.NET – это платформа для разработчиков, состоящая из инструментов, языков программирования и библиотек для создания множества различных типов приложений. ASP.NET Core расширяет платформу разработчика .NET инструментами и библиотеками, специально предназначенными для создания веб-приложений.

**Приложение Б**  
**(рекомендуемое)**  
**Атрибуты сущностей базы данных**

Таблица Б.1 – Атрибуты сущности «Пользователь»

| Идентификатор столбца | Содержательное описание               | Тип данных | Размерность           | Область допустимых значений | Возможность значения Null | Роль | Пример     |
|-----------------------|---------------------------------------|------------|-----------------------|-----------------------------|---------------------------|------|------------|
| ID                    | Уникальный идентификатор пользователя | integer    | 10                    | 0-9                         | нет                       | PK   | 1245       |
| Имя                   | Имя пользователя в системе            | varchar    | 255                   | a-z A-z 0-9 . _ -           | нет                       |      | user_123   |
| Пароль                | Пароль пользователя                   | varchar    | 255                   | a-z A-z 0-9 . _ - ! ? @ * + | нет                       |      | Parol123!  |
| Дата рождения         | Дата рождения пользователя            | date       | 01.01.1950-01.01.2020 |                             | да                        |      | 14.04.2000 |
| Вес                   | Вес пользователя                      | integer    | 3                     | 0-9                         | да                        |      | 75         |
| Рост                  | Рост пользователя                     | integer    | 3                     | 0-9                         | да                        |      | 175        |

Таблица Б.2 – Атрибуты сущности «Тренировка»

| Идентификатор столбца | Содержательное описание                | Тип данных | Размерность | Область допустимых значений       | Возможность значения Null | Роль | Пример                                              |
|-----------------------|----------------------------------------|------------|-------------|-----------------------------------|---------------------------|------|-----------------------------------------------------|
| ID                    | Уникальный идентификатор тренировки    | integer    | 10          | 0-9                               | нет                       | PK   | 1245                                                |
| Наименование          | Название тренировки                    | varchar    | 255         | a-я A-Я 0-9 . , -                 | нет                       |      | Йога 2.0                                            |
| Описание              | Описание тренировки                    | varchar    | 500         | a-z A-z 0-9 . , -                 | нет                       |      | Легкая тренировка для зарядки                       |
| Продолжительность     | Продолжительность тренировки в минутах | integer    | 10          | 0-9                               | нет                       |      | 15                                                  |
| Видеоссылка           | Ссылка на видеозапись тренировки       | varchar    | 255         | a-z A-z 0-9 . _ - ~ ? @ : / # & = | нет                       |      | <a href="https://rutube.ru/">https://rutube.ru/</a> |
| Значение MET          | Значение MET тренировки                | real       | 10          | 0-9                               | нет                       |      | 3.2                                                 |

Таблица Б.3 – Атрибуты сущности «Программа тренировок»

| Идентификатор столбца | Содержательное описание                       | Тип данных | Размерность | Область допустимых значений | Возможность значения Null | Роль | Пример                           |
|-----------------------|-----------------------------------------------|------------|-------------|-----------------------------|---------------------------|------|----------------------------------|
| ID                    | Уникальный идентификатор программы тренировок | integer    | 10          | 0-9                         | нет                       | PK   | 1245                             |
| Наименование          | Название программы тренировок                 | varchar    | 255         | а-я А-Я 0-9 . , -           | нет                       |      | Йога для начинающих              |
| Описание              | Описание программы тренировок                 | varchar    | 500         | а-z А-z 0-9 . , -           | нет                       |      | Легкие тренировки на каждый день |

Таблица Б.4 – Атрибуты сущности «Избранные тренировки»

| Идентификатор столбца | Содержательное описание               | Тип данных | Размерность | Область допустимых значений | Возможность значения Null | Роль   | Пример |
|-----------------------|---------------------------------------|------------|-------------|-----------------------------|---------------------------|--------|--------|
| ПользовательID        | Уникальный идентификатор пользователя | integer    | 10          | 0-9                         | нет                       | PK, FK | 1245   |
| ТренировкаID          | Уникальный идентификатор тренировки   | integer    | 10          | 0-9                         | нет                       | PK, FK | 1245   |

Таблица Б.5 – Атрибуты сущности «Список тренировок программы»

| Идентификатор столбца  | Содержательное описание                       | Тип данных | Размерность | Область допустимых значений | Возможность значения Null | Роль   | Пример |
|------------------------|-----------------------------------------------|------------|-------------|-----------------------------|---------------------------|--------|--------|
| Программа тренировкиID | Уникальный идентификатор программы тренировок | integer    | 10          | 0-9                         | нет                       | PK, FK | 1245   |
| ТренировкаID           | Уникальный идентификатор тренировки           | integer    | 10          | 0-9                         | нет                       | PK, FK | 1245   |

## **Приложение В**

**(обязательное)**

**Листинг серверной части приложения**

### Листинг В.1 – Файл AuthController.cs

```

using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using SportApi.Models;
using SportApi.Services;

namespace SportApi.Controllers
{
    [ApiController]
    [Route("api/auth")]
    public class AuthController : ControllerBase
    {
        private readonly UserService _userService;

        public AuthController(UserService userService)
        {
            _userService = userService;
        }

        public class LoginRequest
        {
            public string username { get; set; }
            public string password { get; set; }
            public string? role { get; set; }
        }

        [HttpPost("login")]
        public async Task<IActionResult> Login([FromBody]
LoginRequest request)
        {
            try
            {
                var result = await
_userService.LoginAsync(request.username, request.password);
                return Ok(new { Token = result.Token, UserId =
result.UserId });
            }
            catch (UnauthorizedAccessException ex)
            {
                return Unauthorized(ex.Message);
            }
        }

        [HttpPost("register")]
        public async Task<IActionResult> Register([FromBody]
LoginRequest request)
        {
            try
            {
                await _userService.RegisterAsync(request.username,
request.password, request.role);
            }
            catch (UnauthorizedAccessException ex)
            {
                return Unauthorized(ex.Message);
            }
        }
    }
}

```

```

        return Ok("Регистрация прошла успешно.");
    }
    catch (ArgumentException ex)
    {
        return BadRequest(ex.Message);
    }
}

[Authorize]
[HttpPost("{userId}/favorites/add")]
public async Task<IActionResult> AddFavoriteWorkout(int
userId, int workoutId)
{
    try
    {
        await _userService.AddFavoriteWorkoutAsync(userId,
workoutId);
        return Ok("Тренировка добавлена в избранное.");
    }
    catch (KeyNotFoundException ex)
    {
        return NotFound(ex.Message);
    }
    catch (InvalidOperationException ex)
    {
        return BadRequest(ex.Message);
    }
}

[Authorize]
[HttpPost("{userId}/favorites/remove")]
public async Task<IActionResult> RemoveFavoriteWorkout(int
userId, int workoutId)
{
    try
    {
        await
        _userService.RemoveFavoriteWorkoutAsync(userId, workoutId);
        return Ok("Тренировка удалена из избранного.");
    }
    catch (KeyNotFoundException ex)
    {
        return NotFound(ex.Message);
    }
}

[Authorize]
[HttpPost("{userId}/update-profile")]
public async Task<IActionResult> UpdateProfile(int userId,
[FromBody] ProfileUpdateRequest request)
{
    try

```

```

        {
            await _userService.UpdateProfileAsync(userId,
request);
            return Ok("Профиль обновлен.");
        }
        catch (KeyNotFoundException ex)
        {
            return NotFound(ex.Message);
        }
    }

    public class ProfileUpdateRequest
    {
        public DateTime? dateOfBirth { get; set; }
        public float? weight { get; set; }
        public float? height { get; set; }
    }

    [Authorize]
    [HttpGet("{userId}")]
    public async Task<IActionResult> GetProfile(int userId)
    {
        try
        {
            var profile = await
_userService.GetProfileAsync(userId);
            return Ok(profile);
        }
        catch (KeyNotFoundException ex)
        {
            return NotFound(ex.Message);
        }
    }
}
}

```

## Листинг В.2 – Файл Repository.cs

```

using Microsoft.EntityFrameworkCore;
using SportApi.Models;

namespace SportApi.Controllers
{
    public interface IRepository<T>
    {
        T GetById(int id);
        IEnumerable<T> GetAll();
        void Add(T entity);
        void Update(T entity);
        void Delete(T entity);
        void Save();
    }

    public class WorkoutRepository : IRepository<Workout>

```

```

    {
        private readonly WorkoutContext _context;

        public WorkoutRepository(WorkoutContext context)
        {
            _context = context;
        }

        public Workout GetById(int id) =>
            _context.Workouts.Find(id);

        public IEnumerable<Workout> GetAll() =>
            _context.Workouts.ToList();

        public void Add(Workout workout) =>
            _context.Workouts.Add(workout);

        public void Update(Workout workout) =>
            _context.Workouts.Update(workout);

        public void Delete(Workout workout) =>
            _context.Workouts.Remove(workout);

        public void Save() => _context.SaveChanges();
    }

    public class WorkoutProgramRepository :
        IRepository<WorkoutProgram>
    {
        private readonly WorkoutContext _context;

        public WorkoutProgramRepository(WorkoutContext context)
        {
            _context = context;
        }

        public WorkoutProgram GetById(int id)
        {
            return _context.WorkoutPrograms
                .Include(p => p.Workouts)
                .FirstOrDefault(p => p.Id == id);
        }

        public IEnumerable<WorkoutProgram> GetAll()
        {
            return _context.WorkoutPrograms
                .Include(p => p.Workouts)
                .ToList();
        }

        public void Add(WorkoutProgram program)
        {

```

```

        _context.WorkoutPrograms.Add(program);
    }

    public void AddWorkout(int programId, int workoutId)
    {
        var program = _context.WorkoutPrograms
            .Include(p => p.Workouts)
            .FirstOrDefault(p => p.Id == programId);

        if (program == null) return;

        var workout = _context.Workouts.Find(workoutId);
        if (workout != null &&
!program.Workouts.Contains(workout))
        {
            program.Workouts.Add(workout);
        }
    }

    public void RemoveWorkout(int programId, int workoutId)
    {
        var program = _context.WorkoutPrograms
            .Include(p => p.Workouts)
            .FirstOrDefault(p => p.Id == programId);

        if (program == null) return;

        var workout = _context.Workouts.Find(workoutId);
        if (workout != null &&
program.Workouts.Contains(workout))
        {
            program.Workouts.Remove(workout);
        }
    }

    public void Update(WorkoutProgram program)
    {
        _context.WorkoutPrograms.Update(program);
    }

    public void Delete(WorkoutProgram program)
    {
        _context.WorkoutPrograms.Remove(program);
    }

    public void Save()
    {
        _context.SaveChanges();
    }
}

public class UserRepository : IRepository<User>
{

```

```

        private readonly WorkoutContext _context;

public UserRepository(WorkoutContext context)
    {
        _context = context;
    }
    public User GetById(int id)
    {
        return _context.Users
            .Include(u => u.FavoriteWorkouts)
            .FirstOrDefault(u => u.Id == id);
    }
    public User GetByUsername(string username)
    {
        return _context.Users.FirstOrDefault(u => u.Username
== username);
    }
    public IEnumerable<User> GetAll()
    {
        return _context.Users.ToList();
    }
    public void Add(User user)
    {
        _context.Users.Add(user);
    }
    public void Update(User user)
    {
        _context.Users.Update(user);
    }
    public void Delete(User user)
    {
        _context.Users.Remove(user);
    }
    public void AddToFav(int userId, int workoutId)
    {
        var user = _context.Users
            .Include(u => u.FavoriteWorkouts)
            .FirstOrDefault(u => u.Id == userId);

        if (user == null) return;

        var workout = _context.Workouts.Find(workoutId);
        if (workout != null &&
!user.FavoriteWorkouts.Contains(workout))
        {
            user.FavoriteWorkouts.Add(workout);
        }
    }
    public void RemoveFromFav(int userId, int workoutId)
    {
        var user = _context.Users
            .Include(u => u.FavoriteWorkouts)

```

```

        .FirstOrDefault(u => u.Id == userId);

        if (user == null) return;

        var workout = _context.Workouts.Find(workoutId);
        if (workout != null &&
user.FavoriteWorkouts.Contains(workout))
        {
            user.FavoriteWorkouts.Remove(workout);
        }
    }
    public void Save()
    {
        _context.SaveChanges();
    }
}
}

```

### Листинг В.3 – Файл Strategy.cs

```

using SportApi.Models;

namespace SportApi.Controllers
{
    public interface IFilterStrategy<T>
    {
        IEnumerable<T> Filter(IEnumerable<T> items);
    }

    public class DurationFilter : IFilterStrategy<Workout>
    {
        private readonly int _minDuration;

        public DurationFilter(int minDuration)
        {
            _minDuration = minDuration;
        }

        public IEnumerable<Workout> Filter(IEnumerable<Workout>
items)
        {
            return items.Where(w => w.Duration >= _minDuration);
        }
    }

    public class KeywordFilter : IFilterStrategy<WorkoutProgram>
    {
        private readonly string _keyword;

        public KeywordFilter(string keyword)
        {
            _keyword = keyword?.ToLower();
        }
    }
}

```

```

    }

    public IEnumerable<WorkoutProgram>
Filter(IEnumerable<WorkoutProgram> items)
    {
        return items.Where(program =>
            program.Name?.ToLower().Contains(_keyword) ==
true
            program.Description?.ToLower().Contains(_keyword)
== true

            program.Workouts.Any(workout =>
                workout.Name?.ToLower().Contains(_keyword) ==
true ||
workout.Description?.ToLower().Contains(_keyword) == true
            )
        );
    }
}

public class MinWorkoutCountFilter :
IFilterStrategy<WorkoutProgram>
{
    private readonly int _minWorkouts;

    public MinWorkoutCountFilter(int minWorkouts)
    {
        _minWorkouts = minWorkouts;
    }

    public IEnumerable<WorkoutProgram>
Filter(IEnumerable<WorkoutProgram> items)
    {
        return items.Where(p => p.Workouts.Count >=
_minWorkouts);
    }
}

public class FilterContext<T>
{
    private IFilterStrategy<T> _strategy;

    public void SetStrategy(IFilterStrategy<T> strategy)
    {
        _strategy = strategy;
    }

    public IEnumerable<T> ApplyFilter(IEnumerable<T> items)
    {
        if (_strategy == null)
        {

```

```

        throw new InvalidOperationException("Filter
strategy is not set.");
    }
    return _strategy.Filter(items);
}
}
}

```

#### Листинг В.4 – Файл WorkoutProgramsController.cs

```

using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using SportApi.Models;
using SportApi.Services;
using static SportApi.Controllers.WorkoutProgramsController;

namespace SportApi.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class WorkoutProgramsController : ControllerBase
    {
        private readonly WorkoutProgramService
        _workoutProgramService;

        public WorkoutProgramsController(WorkoutProgramService
        workoutProgramService)
        {
            _workoutProgramService = workoutProgramService;
        }

        // GET: api/workoutprograms
        [Authorize]
        [HttpGet]
        public async
        Task<ActionResult<IEnumerable<WorkoutProgram>>>
        GetWorkoutPrograms()
        {
            try
            {
                var programs = await
                _workoutProgramService.GetAllWorkoutProgramsAsync();
                return Ok(programs);
            }
            catch (Exception ex)
            {
                return StatusCode(500, $"Internal server error:
{ex.Message}");
            }
        }

        // GET: api/workoutprograms/{id}
    }
}

```

```

        [Authorize]
        [HttpGet("{id}")]
        public async Task<ActionResult<WorkoutProgram>>
GetWorkoutProgram(int id)
        {
            try
            {
                var program = await
_workoutProgramService.GetWorkoutProgramByIdAsync(id);
                return Ok(program);
            }
            catch (KeyNotFoundException ex)
            {
                return NotFound(ex.Message);
            }
            catch (Exception ex)
            {
                return StatusCode(500, $"Internal server error:
{ex.Message}");
            }
        }

        // GET: api/workoutprograms/filter
        [Authorize]
        [HttpGet("filter")]
        public async
Task<ActionResult<IEnumerable<WorkoutProgram>>>
FilterWorkoutPrograms([FromQuery] int? minWorkouts, [FromQuery]
string? keyword)
        {
            try
            {
                var filteredPrograms = await
_workoutProgramService.FilterWorkoutProgramsAsync(minWorkouts,
keyword);
                return Ok(filteredPrograms);
            }
            catch (ArgumentException ex)
            {
                return BadRequest(ex.Message);
            }
            catch (Exception ex)
            {
                return StatusCode(500, $"Internal server error:
{ex.Message}");
            }
        }

        public class WorkoutProgramDTO
        {
            public string Name { get; set; }
            public string Description { get; set; }

```

```

        public ICollection<int> Workouts { get; set; } = new
List<int>();
    }

    // POST: api/WorkoutPrograms
    [Authorize]
    [HttpPost]
    public async Task<ActionResult<WorkoutProgram>>
PostWorkoutProgram([FromBody] WorkoutProgramDTO workoutProgramDTO)
    {
        try
        {
            var workoutProgram = new WorkoutProgram
            {
                Name = workoutProgramDTO.Name,
                Description = workoutProgramDTO.Description,
                Workouts = workoutProgramDTO.Workouts
                .Select(workoutId => new Workout { Id = workoutId
})) // Преобразуем ID в объекты Workout
                .ToList()
            };
            var newProgram = await
_workoutProgramService.AddWorkoutProgramAsync(workoutProgram);
            return CreatedAtAction(nameof(GetWorkoutProgram),
new { id = newProgram.Id }, newProgram);
        }
        catch (ArgumentException ex)
        {
            return BadRequest(ex.Message);
        }
        catch (Exception ex)
        {
            return StatusCode(500, $"Internal server error:
{ex.Message}");
        }
    }

    // PUT: api/WorkoutPrograms/{id}
    [Authorize]
    [HttpPut("{id}")]
    public async Task<IActionResult> PutWorkoutProgram(int id,
[FromBody] WorkoutProgramDTO workoutProgram)
    {
        try
        {
            await
_workoutProgramService.UpdateWorkoutProgramAsync(id,
workoutProgram);
            return NoContent();
        }
        catch (KeyNotFoundException ex)
        {

```

```

        return NotFound(ex.Message);
    }
    catch (ArgumentException ex)
    {
        return BadRequest(ex.Message);
    }
    catch (Exception ex)
    {
        return StatusCode(500, $"Internal server error:
{ex.Message}");
    }
}

// [Authorize]
[HttpPost("{id}/add")]
public async Task<IActionResult> AddWorkout(int id, int
workoutId)
{
    try
    {
        await _workoutProgramService.AddWorkoutAsync(id,
workoutId);
        return Ok("Тренировка добавлена в программу.");
    }
    catch (KeyNotFoundException ex)
    {
        return NotFound(ex.Message);
    }
    catch (InvalidOperationException ex)
    {
        return BadRequest(ex.Message);
    }
}

// [Authorize]
[HttpPost("{id}/remove")]
public async Task<IActionResult> RemoveWorkout(int id, int
workoutId)
{
    try
    {
        await
_workoutProgramService.RemoveWorkoutAsync(id, workoutId);
        return Ok("Тренировка удалена из программы.");
    }
    catch (KeyNotFoundException ex)
    {
        return NotFound(ex.Message);
    }
}

// DELETE: api/WorkoutPrograms/{id}

```

```

        [Authorize]
        [HttpDelete("{id}")]
        public async Task<IActionResult> DeleteWorkoutProgram(int
id)
        {
            try
            {
                await
_workoutProgramService.DeleteWorkoutProgramAsync(id);
                return NoContent();
            }
            catch (KeyNotFoundException ex)
            {
                return NotFound(ex.Message);
            }
            catch (Exception ex)
            {
                return StatusCode(500, $"Internal server error:
{ex.Message}");
            }
        }
    }
}

```

### Листинг В.5 – Файл WorkoutsController.cs

```

using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using SportApi.Models;
using SportApi.Services;

namespace SportApi.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class WorkoutsController : ControllerBase
    {
        private readonly WorkoutService _workoutService;
        public WorkoutsController(WorkoutService workoutService)
        {
            _workoutService = workoutService;
        }

        // GET: api/workouts
        [Authorize]
        [HttpGet]
        public async Task<ActionResult<IEnumerable<Workout>>>
GetWorkouts()
        {
            try
            {

```

```

        var workouts = await
_workoutService.GetAllWorkoutsAsync();
        return Ok(workouts);
    }
    catch (Exception ex)
    {
        return StatusCode(500, $"Internal server error:
{ex.Message}");
    }
}

// GET: api/workouts/{id}
[Authorize]
[HttpGet("{id}")]
public async Task<ActionResult<Workout>> GetWorkout(int
id)
{
    try
    {
        var workout = await
_workoutService.GetWorkoutByIdAsync(id);
        return Ok(workout);
    }
    catch (KeyNotFoundException ex)
    {
        return NotFound(ex.Message);
    }
    catch (Exception ex)
    {
        return StatusCode(500, $"Internal server error:
{ex.Message}");
    }
}

// GET: api/workouts/filter
[Authorize]
[HttpGet("filter")]
public async Task<ActionResult<IEnumerable<Workout>>>
FilterWorkouts([FromQuery] int? minDifficulty, [FromQuery] int?
minDuration)
{
    try
    {
        var filteredWorkouts = await
_workoutService.FilterWorkoutsAsync(minDifficulty, minDuration);
        return Ok(filteredWorkouts);
    }
    catch (ArgumentException ex)
    {
        return BadRequest(ex.Message);
    }
    catch (Exception ex)

```

```

        {
            return StatusCode(500, $"Internal server error:
{ex.Message}");
        }
    }

    public class WorkoutDTO
    {
        public string Name { get; set; }
        public string Description { get; set; }
        public int Duration { get; set; }
        public float MetValue { get; set; }
        public string VideoUrl { get; set; }
    }

    // POST: api/Workouts
    [Authorize]
    [HttpPost]
    public async Task<ActionResult<Workout>>
PostWorkout([FromBody] Workout workout)
    {
        try
        {
            var newWorkout = await
_workoutService.AddWorkoutAsync(workout);
            return CreatedAtAction(nameof(GetWorkout), new {
id = newWorkout.Id }, newWorkout);
        }
        catch (ArgumentException ex)
        {
            return BadRequest(ex.Message);
        }
        catch (Exception ex)
        {
            return StatusCode(500, $"Internal server error:
{ex.Message}");
        }
    }

    // PUT: api/Workouts/{id}
    [Authorize]
    [HttpPut("{id}")]
    public async Task<IAActionResult> PutWorkout(int id,
[FromBody] Workout workout)
    {
        if (id != workout.Id)
        {
            return BadRequest("ID в запросе не совпадает с ID
объекта.");
        }
    }

```

```

        try
        {
            await _workoutService.UpdateWorkoutAsync(id,
workout);
            return NoContent();
        }
        catch (KeyNotFoundException ex)
        {
            return NotFound(ex.Message);
        }
        catch (Exception ex)
        {
            return StatusCode(500, $"Internal server error:
{ex.Message}");
        }
    }

[HttpPost("recommend")]
public IActionResult RecommendWorkouts(double UserWeight,
string GoalType)
{
    try
    {
        if (UserWeight <= 0 ||
string.IsNullOrEmpty(GoalType))
        {
            return BadRequest("Некорректные данные
запроса.");
        }

        var recommendations =
_workoutService.GetRecommendedWorkouts(UserWeight, GoalType);
        return Ok(recommendations);
    }
    catch (Exception ex)
    {
        return StatusCode(500, $"Ошибка сервера:
{ex.Message}");
    }
}

// DELETE: api/Workouts/{id}
[Authorize]
[HttpDelete("{id}")]
public async Task<IActionResult> DeleteWorkout(int id)
{
    try
    {
        await _workoutService.DeleteWorkoutAsync(id);
        return NoContent();
    }
    catch (KeyNotFoundException ex)

```

```

        {
            return NotFound(ex.Message);
        }
        catch (Exception ex)
        {
            return StatusCode(500, $"Internal server error:
{ex.Message}");
        }
    }
}

```

### Листинг В.6 – Файл User.cs

```

namespace SportApi.Models
{
    public class User
    {
        public int Id { get; set; }
        public string Username { get; set; }
        public string PasswordHash { get; set; }
        public string Role { get; set; }
        public DateTime? DateOfBirth { get; set; }
        public float? Weight { get; set; }
        public float? Height { get; set; }
        public ICollection<Workout> FavoriteWorkouts { get; set; }
    }
}
= new List<Workout>();

```

### Листинг В.7 – Файл Workout.cs

```

using System.Text.Json.Serialization;

namespace SportApi.Models
{
    public class Workout
    {
        public int Id { get; set; }
        public string Name { get; set; }
        public string Description { get; set; }
        public int Duration { get; set; }
        public float MetValue { get; set; }
        public string VideoUrl { get; set; }

        [JsonIgnore]
        public ICollection<WorkoutProgram> WorkoutPrograms { get;
set; } = new List<WorkoutProgram>();

        [JsonIgnore]
        public ICollection<User> FavoritedByUsers { get; set; } =
new List<User>();
    }
}

```

```
    }
}
```

### Листинг В.8 – Файл WorkoutProgram.cs

```
using System.Collections.Generic;
using System.Text.Json.Serialization;
namespace SportApi.Models
{
    public class WorkoutProgram
    {
        public int Id { get; set; }
        public string Name { get; set; }
        public string Description { get; set; }
        public ICollection<Workout> Workouts { get; set; } = new
List<Workout>();
    }
}
```

### Листинг В.9 – Файл WorkoutContext.cs

```
using Microsoft.EntityFrameworkCore;
using SportApi.Models;
namespace SportApi.Models
{
    public class WorkoutContext : DbContext
    {
        public WorkoutContext(DbContextOptions<WorkoutContext>
options)
            : base(options)
        {
        }

        public DbSet<Workout> Workouts { get; set; }
        public DbSet<WorkoutProgram> WorkoutPrograms { get; set; }
        public DbSet<User> Users { get; set; }
    }
}
```

### Листинг В.10 – Файл WorkoutProgramFactory.cs

```
namespace SportApi.Models
{
    public class WorkoutProgramFactory
    {
        private readonly WorkoutContext _context;
        public WorkoutProgramFactory(WorkoutContext context)
        {
            _context = context;
        }
        public WorkoutProgram CreateProgram(string name, string
description, List<int> workoutIds)
        {
        }
    }
}
```

```

        var workouts = _context.Workouts.Where(w =>
workoutIds.Contains(w.Id)).ToList();

        if (workouts.Count != workoutIds.Count)
        {
            throw new ArgumentException("Некоторые тренировки
не существуют.");
        }

        return new WorkoutProgram
        {
            Name = name,
            Description = description,
            Workouts = workouts
        };
    }
}

public class WorkoutFactory
{
    private readonly WorkoutContext _context;

    public WorkoutFactory(WorkoutContext context)
    {
        _context = context;
    }

    public Workout CreateWorkout(string name, string
description, int duration, float metValue, string videoUrl)
    {
        if (string.IsNullOrEmpty(name))
        {
            throw new ArgumentException("Название тренировки
не может быть пустым.");
        }

        if (duration <= 0)
        {
            throw new ArgumentException("Продолжительность
должна быть больше 0.");
        }

        if (!string.IsNullOrEmpty(videoUrl) &&
!Uri.IsWellFormedUriString(videoUrl, UriKind.Absolute))
        {
            throw new ArgumentException("Недействительная
ссылка на видео.");
        }

        return new Workout
        {
            Name = name,
            Description = description,

```

```

        Duration = duration,
        MetValue = metValue,
        VideoUrl = videoUrl
    };
}
}
}

```

### Листинг В.11 – Файл TokenService.cs

```

using Microsoft.AspNetCore.Mvc;
using Microsoft.IdentityModel.Tokens;
using System.IdentityModel.Tokens.Jwt;
using System.Security.Claims;
using System.Text;

namespace SportApi.Services
{
    public class TokenService
    {
        private readonly string _secretKey;

        public TokenService(string secretKey)
        {
            _secretKey = secretKey;
        }

        public string GenerateToken(int userId, string username)
        {
            var claims = new[]
            {
                new Claim(JwtRegisteredClaimNames.Sub,
                    userId.ToString()),
                new Claim(JwtRegisteredClaimNames.UniqueName,
                    username),
                new Claim(JwtRegisteredClaimNames.Jti,
                    Guid.NewGuid().ToString())
            };

            var key = new
                SymmetricSecurityKey(Encoding.UTF8.GetBytes(_secretKey));
            var credentials = new SigningCredentials(key,
                SecurityAlgorithms.HmacSha256);

            var token = new JwtSecurityToken(
                issuer: "SportApi",
                audience: "SportApiClient",
                claims: claims,
                expires: DateTime.UtcNow.AddHours(1),
                signingCredentials: credentials
            );

```

```

        return new
JwtSecurityTokenHandler().WriteToken(token);
    }
}

```

### Листинг В.12 – Файл UserService.cs

```

using SportApi.Controllers;
using SportApi.Models;
using static SportApi.Controllers.AuthController;

namespace SportApi.Services
{
    public class UserService
    {
        private readonly UserRepository _repository;
        private readonly TokenService _tokenService;

        public UserService(UserRepository repository, TokenService
tokenService)
        {
            _repository = repository;
            _tokenService = tokenService;
        }

        public async Task<(string Token, int UserId)>
LoginAsync(string username, string password)
        {
            var user = _repository.GetByUsername(username);

            if (user == null ||
!BCrypt.Net.BCrypt.Verify(password, user.PasswordHash))
            {
                throw new UnauthorizedAccessException("Неверное
имя пользователя или пароль.");
            }

            var token = _tokenService.GenerateToken(user.Id,
user.Username);
            return (token, user.Id);
        }

        public async Task RegisterAsync(string username, string
password, string? role)
        {
            if (_repository.GetByUsername(username) != null)
            {
                throw new ArgumentException("Пользователь с таким
именем уже существует.");
            }
        }
    }
}

```

```

        var hashedPassword =
BCrypt.Net.BCrypt.HashPassword(password);
        var user = new User { Username = username,
PasswordHash = hashedPassword, Role = role };
        _repository.Add(user);
        _repository.Save();
    }

    public async Task AddFavoriteWorkoutAsync(int userId, int
workoutId)
    {
        var user = _repository.GetById(userId);
        if (user == null)
        {
            throw new KeyNotFoundException("Пользователь не
найден.");
        }

        if (user.FavoriteWorkouts.Any(w => w.Id == workoutId))
        {
            throw new InvalidOperationException("Тренировка
уже добавлена в избранное.");
        }

        _repository.AddToFav(userId, workoutId);
        _repository.Save();
    }

    public async Task RemoveFavoriteWorkoutAsync(int userId,
int workoutId)
    {
        var user = _repository.GetById(userId);
        if (user == null)
        {
            throw new KeyNotFoundException("Пользователь не
найден.");
        }

        _repository.RemoveFromFav(userId, workoutId);
        _repository.Save();
    }

    public async Task UpdateProfileAsync(int userId,
ProfileUpdateRequest request)
    {
        var user = _repository.GetById(userId);
        if (user == null)
        {
            throw new KeyNotFoundException("Пользователь не
найден.");
        }
    }

```

```

        user.DateOfBirth = request.dateOfBirth;
        user.Weight = request.weight;
        user.Height = request.height;

        _repository.Save();
    }

    public async Task<object> GetProfileAsync(int userId)
    {
        var user = _repository.GetById(userId);
        if (user == null)
        {
            throw new KeyNotFoundException("Пользователь не
найден.");
        }

        return new
        {
            user.Username,
            user.Role,
            user.DateOfBirth,
            user.Weight,
            user.Height,
            user.FavoriteWorkouts
        };
    }
}

```

### Листинг В.13 – Файл UserService.cs

```

using SportApi.Controllers;
using SportApi.Models;
using static SportApi.Controllers.AuthController;

namespace SportApi.Services
{
    public class UserService
    {
        private readonly UserRepository _repository;
        private readonly TokenService _tokenService;

        public UserService(UserRepository repository, TokenService
tokenService)
        {
            _repository = repository;
            _tokenService = tokenService;
        }

        public async Task<(string Token, int UserId)>
LoginAsync(string username, string password)
        {
            var user = _repository.GetByUsername(username);

```

```

        if (user == null ||
!BCrypt.Net.BCrypt.Verify(password, user.PasswordHash))
        {
            throw new UnauthorizedAccessException("Неверное
имя пользователя или пароль.");
        }

        var token = _tokenService.GenerateToken(user.Id,
user.Username);
        return (token, user.Id);
    }

    public async Task RegisterAsync(string username, string
password, string? role)
    {
        if (_repository.GetByUsername(username) != null)
        {
            throw new ArgumentException("Пользователь с таким
именем уже существует.");
        }

        var hashedPassword =
BCrypt.Net.BCrypt.HashPassword(password);
        var user = new User { Username = username,
PasswordHash = hashedPassword, Role = role };
        _repository.Add(user);
        _repository.Save();
    }

    public async Task AddFavoriteWorkoutAsync(int userId, int
workoutId)
    {
        var user = _repository.GetById(userId);
        if (user == null)
        {
            throw new KeyNotFoundException("Пользователь не
найден.");
        }

        if (user.FavoriteWorkouts.Any(w => w.Id == workoutId))
        {
            throw new InvalidOperationException("Тренировка
уже добавлена в избранное.");
        }

        _repository.AddToFav(userId, workoutId);
        _repository.Save();
    }

    public async Task RemoveFavoriteWorkoutAsync(int userId,
int workoutId)

```

```

        {
            var user = _repository.GetById(userId);
            if (user == null)
            {
                throw new KeyNotFoundException("Пользователь не
найден.");
            }

            _repository.RemoveFromFav(userId, workoutId);
            _repository.Save();
        }

        public async Task UpdateProfileAsync(int userId,
ProfileUpdateRequest request)
        {
            var user = _repository.GetById(userId);
            if (user == null)
            {
                throw new KeyNotFoundException("Пользователь не
найден.");
            }

            user.DateOfBirth = request.dateOfBirth;
            user.Weight = request.weight;
            user.Height = request.height;

            _repository.Save();
        }

        public async Task<object> GetProfileAsync(int userId)
        {
            var user = _repository.GetById(userId);
            if (user == null)
            {
                throw new KeyNotFoundException("Пользователь не
найден.");
            }

            return new
            {
                user.Username,
                user.Role,
                user.DateOfBirth,
                user.Weight,
                user.Height,
                user.FavoriteWorkouts
            };
        }
    }
}

```

## Листинг В.14 – Файл WorkoutProgramService.cs

```

using SportApi.Controllers;
using SportApi.Models;
using Microsoft.EntityFrameworkCore;
using static SportApi.Controllers.WorkoutProgramsController;

namespace SportApi.Services
{
    public class WorkoutProgramService
    {
        private readonly WorkoutProgramRepository _repository;
        private readonly WorkoutProgramFactory _factory;
        private readonly WorkoutContext _context;

        public WorkoutProgramService(WorkoutProgramRepository
repository, WorkoutProgramFactory factory, WorkoutContext context)
        {
            _repository = repository;
            _factory = factory;
            _context = context;
        }

        public async Task<IEnumerable<WorkoutProgram>>
GetAllWorkoutProgramsAsync()
        {
            return _repository.GetAll();
        }

        public async Task<WorkoutProgram>
GetWorkoutProgramByIdAsync(int id)
        {
            var program = _repository.GetById(id);
            if (program == null)
            {
                throw new KeyNotFoundException("Программа
тренировок не найдена.");
            }
            return program;
        }

        public async Task<IEnumerable<WorkoutProgram>>
FilterWorkoutProgramsAsync(int? minWorkouts, string? keyword)
        {
            var context = new FilterContext<WorkoutProgram>();

            if (!string.IsNullOrEmpty(keyword))
            {
                context.SetStrategy(new KeywordFilter(keyword));
            }
            else if (minWorkouts.HasValue)
            {

```

```

        context.SetStrategy(new
MinWorkoutCountFilter(minWorkouts.Value));
    }
    else
    {
        throw new ArgumentException("No filter criteria
provided.");
    }

    var filteredPrograms =
context.ApplyFilter(_context.WorkoutPrograms.Include(p =>
p.Workouts));
    return filteredPrograms.ToList();
}

public async Task<WorkoutProgram>
AddWorkoutProgramAsync(WorkoutProgram workoutProgram)
{
    var newProgram = _factory.CreateProgram(
        workoutProgram.Name,
        workoutProgram.Description,
        workoutProgram.Workouts.Select(w => w.Id).ToList()
    );

    _repository.Add(newProgram);
    _repository.Save();

    return newProgram;
}

public async Task UpdateWorkoutProgramAsync(int id,
WorkoutProgramDTO updatedProgram)
{
    var program = _repository.GetById(id);
    if (program == null)
    {
        throw new KeyNotFoundException("Программа
тренировок не найдена.");
    }

    program.Name = updatedProgram.Name;
    program.Description = updatedProgram.Description;

    _repository.Save();
}

public async Task AddWorkoutAsync(int programId, int
workoutId)
{
    var program = _repository.GetById(programId);
    if (program == null)
    {

```

```

        throw new KeyNotFoundException("Программа
тренировок не найдена.");
    }

    if (program.Workouts.Any(w => w.Id == workoutId))
    {
        throw new InvalidOperationException("Тренировка
уже добавлена в программу.");
    }

    _repository.AddWorkout(programId, workoutId);
    _repository.Save();
}

public async Task RemoveWorkoutAsync(int programId, int
workoutId)
{
    var program = _repository.GetById(programId);
    if (program == null)
    {
        throw new KeyNotFoundException("Программа
тренировок не найдена.");
    }

    _repository.RemoveWorkout(programId, workoutId);
    _repository.Save();
}

public async Task DeleteWorkoutProgramAsync(int id)
{
    var program = _repository.GetById(id);
    if (program == null)
    {
        throw new KeyNotFoundException("Программа
тренировок не найдена.");
    }

    _repository.Delete(program);
    _repository.Save();
}
}
}

```

### Листинг В.15 – Файл WorkoutService.cs

```

using Microsoft.EntityFrameworkCore;
using SportApi.Controllers;
using SportApi.Models;

namespace SportApi.Services
{
    public class WorkoutService
    {

```

```

        private readonly WorkoutRepository _repository;
        private readonly WorkoutFactory _factory;

        public WorkoutService(WorkoutRepository repository,
WorkoutFactory factory)
        {
            _repository = repository;
            _factory = factory;
        }

        public async Task<IEnumerable<Workout>>
GetAllWorkoutsAsync()
        {
            return _repository.GetAll();
        }

        public async Task<Workout> GetWorkoutByIdAsync(int id)
        {
            var workout = _repository.GetById(id);
            if (workout == null)
            {
                throw new KeyNotFoundException("Тренировка не
найдена.");
            }
            return workout;
        }

        public async Task<IEnumerable<Workout>>
FilterWorkoutsAsync(int? minDifficulty, int? minDuration)
        {
            var context = new FilterContext<Workout>();
            var workouts = _repository.GetAll().AsQueryable();

            if (minDuration.HasValue)
            {
                context.SetStrategy(new
DurationFilter(minDuration.Value));
            }
            else
            {
                throw new ArgumentException("Не указаны критерии
фильтрации.");
            }
            var filteredWorkouts =
context.ApplyFilter(workouts.AsEnumerable()).AsQueryable();

            return filteredWorkouts.ToList();
        }

        public async Task<Workout> AddWorkoutAsync(Workout
workout)
        {

```

```

        var newWorkout = _factory.CreateWorkout(
            workout.Name,
            workout.Description,
            workout.Duration,
            workout.MetValue,
            workout.VideoUrl);

        _repository.Add(newWorkout);
        _repository.Save();

        return newWorkout;
    }

    public async Task UpdateWorkoutAsync(int id, Workout
updatedWorkout)
    {
        _repository.Update(updatedWorkout);
        _repository.Save();
    }

    public List<RecommendedWorkout>
GetRecommendedWorkouts(double userWeight, string goalType)
    {
        if (userWeight <= 0 || string.IsNullOrEmpty(goalType))
        {
            throw new ArgumentException("Некорректные данные
запроса.");
        }

        var allWorkouts = _repository.GetAll();

        var recommendations = allWorkouts
            .Select(workout =>
            {
                var caloriesBurned =
CalculateCaloriesBurned(userWeight, workout.Duration,
workout.MetValue);
                return new RecommendedWorkout
                {
                    Id = workout.Id,
                    Name = workout.Name,
                    Description = workout.Description,
                    Duration = workout.Duration,
                    MetValue = workout.MetValue,
                    VideoUrl = workout.VideoUrl,
                    CaloriesBurned = caloriesBurned
                };
            })
            .Where(workout =>
            {
                if (goalType == "weight-loss" &&
workout.CaloriesBurned > 300)

```

```

        {
            return true;
        }
        else if (goalType == "maintain" &&
workout.CaloriesBurned <= 300)
        {
            return true;
        }
        return false;
    })
    .Take(10)
    .ToList();

    return recommendations;
}

private double CalculateCaloriesBurned(double userWeight,
int durationInMinutes, double metValue)
{
    double durationInHours = durationInMinutes / 60.0;
    return Math.Round(metValue * userWeight *
durationInHours, 2);
}

public class RecommendedWorkout
{
    public int Id { get; set; }
    public string Name { get; set; }
    public string Description { get; set; }
    public int Duration { get; set; }
    public double MetValue { get; set; }
    public string VideoUrl { get; set; }
    public double CaloriesBurned { get; set; }
}

public async Task DeleteWorkoutAsync(int id)
{
    var workout = _repository.GetById(id);
    if (workout == null)
    {
        throw new KeyNotFoundException("Тренировка не
найдена.");
    }

    _repository.Delete(workout);
    _repository.Save();
}
}
}

```

### Листинг В.16 – Файл Program.cs

```
using SportApi.Models;
```

```

using Microsoft.EntityFrameworkCore;
using SportApi.Controllers;
using Microsoft.AspNetCore.Authentication.JwtBearer;
using Microsoft.IdentityModel.Tokens;
using SportApi.Services;
using System.Text;
using Microsoft.Extensions.Options;
using Microsoft.OpenApi.Models;

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddDbContext<WorkoutContext>(options =>

options.UseSqlServer(builder.Configuration.GetConnectionString("De
faultConnection")));
builder.Services.AddScoped<WorkoutProgramFactory>();
builder.Services.AddScoped<WorkoutFactory>();
builder.Services.AddScoped<WorkoutRepository>();
builder.Services.AddScoped<WorkoutProgramRepository>();
builder.Services.AddScoped<UserRepository>();
builder.Services.AddScoped<UserService>();
builder.Services.AddScoped<WorkoutService>();
builder.Services.AddScoped<WorkoutProgramService>();

builder.Services.AddControllers();
builder.Services.AddEndpointsApiExplorer();
builder.Services.AddSwaggerGen(options =>
{
    options.AddSecurityDefinition("Bearer", new
OpenApiSecurityScheme
    {
        Name = "Authorization",
        Type = SecuritySchemeType.Http,
        Scheme = "bearer",
        BearerFormat = "JWT",
        In = ParameterLocation.Header,
        Description = "Введите токен в формате Bearer {your
token}"
    });

    options.AddSecurityRequirement(new OpenApiSecurityRequirement
    {
        {
            new OpenApiSecurityScheme
            {
                Reference = new OpenApiReference
                {
                    Type = ReferenceType.SecurityScheme,
                    Id = "Bearer"
                }
            },
            new string[] { }
        }
    });

```

```

        }
    });
});

builder.Services.AddCors(options =>
{
    options.AddPolicy("AllowAll", builder =>
    {
        builder.AllowAnyOrigin()
            .AllowAnyMethod()
            .AllowAnyHeader();
    });
});

var secretKey = "MySuperSecretKey1234567890!ExtraLongKey";
builder.Services.AddSingleton(new TokenService(secretKey));

builder.Services.AddAuthentication(JwtBearerDefaults.AuthenticationScheme)
    .AddJwtBearer(options =>
    {
        options.TokenValidationParameters = new
TokenValidationParameters
        {
            ValidateIssuer = true,
            ValidateAudience = true,
            ValidateLifetime = true,
            ValidateIssuerSigningKey = true,
            ValidIssuer = "SportApi",
            ValidAudience = "SportApiClient",
            IssuerSigningKey = new
SymmetricSecurityKey(Encoding.UTF8.GetBytes(secretKey))
        };
    });

builder.Services.AddAuthorization();
builder.Services.AddControllers();

var app = builder.Build();

if (app.Environment.IsDevelopment())
{
    app.UseSwagger();
    app.UseSwaggerUI();
}

app.UseCors("AllowAll");

app.UseHttpsRedirection();

app.UseAuthentication();
app.UseAuthorization();

```

```

app.MapControllers();

app.Run();

public partial class Program { }

```

### Листинг В.17 – Метод AddToFav

```

    public void AddToFav(int userId, int workoutId)
    {
        var user = _context.Users
            .Include(u => u.FavoriteWorkouts)
            .FirstOrDefault(u => u.Id == userId);

        if (user == null) return;

        var workout = _context.Workouts.Find(workoutId);
        if (workout != null &&
!user.FavoriteWorkouts.Contains(workout))
        {
            user.FavoriteWorkouts.Add(workout);
        }
    }

```

**Приложение Г**  
**(обязательное)**  
**Листинг клиентской части**

## Листинг Г.1 – Файл index.html

```

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <title>Программы тренировок</title>
  <link rel="preconnect" href="https://fonts.googleapis.com">
  <link rel="preconnect" href="https://fonts.gstatic.com"
crossorigin>
  <link
href="https://fonts.googleapis.com/css2?family=League+Spartan&fami
ly=Raleway:ital,wght@0,100..900;1,100..900&display=swap"
rel="stylesheet">
  <link href="css/styles.css" rel="stylesheet" type="text/css"/>
</head>
<body>
  <header>
    <h1>Программы тренировок</h1>
    <nav id="menu" style="display: none">
      <ul>
        <li id="profile"><a href="#profile">Личный
кабинет</a></li>
        <li><a href="#programs">Программы</a></li>
        <li id="fav"><a href="#favorites">Избранные
тренировки</a></li>
        <li id="adminp"><a href="#adminprograms">Управление
Программами</a></li>
        <li id="adminw"><a href="#adminworkouts">Управление
Тренировками</a></li>
        <li><button id="logout-btn">Выйти</button></li>
      </ul>
    </nav>
    <div>

    </div>
  </header>
  <div class="container">
    <main id="content-container">
      <div id="auth-container">
        <form id="login-form">
          <h2>Вход</h2>
          <label for="login-username">Имя пользователя:</label>
          <input type="text" id="login-username" required>
          <label for="login-password">Пароль:</label>
          <input type="password" id="login-password" required>
          <div style="text-align:center">
            <button type="submit">Войти</button>
          </div>
        </form>
      </div>
    </main>
  </div>

```

```

        <p>Еще нет аккаунта? <a href="#" id="switch-to-
register">Зарегистрироваться</a></p>
    </form>

    <form id="register-form" style="display: none">
        <h2>Регистрация</h2>
        <label for="register-username">Имя
пользователя:</label>
        <input type="text" id="register-username" required>
        <label for="register-password">Пароль:</label>
        <input type="password" id="register-password"
required>
        <div style="text-align:center">
            <button type="submit">Зарегистрироваться</button>
        </div>
        <p>Уже есть аккаунт? <a href="#" id="switch-to-
login">Войти</a></p>
    </form>
</div>
</main>
</div>

<script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
<script src="js/servPrograms.js" type="module"></script>
<script src="js/servWorkouts.js" type="module"></script>
<script src="js/auth.js" type="module"></script>
<script src="js/profile.js" type="module"></script>
<script src="js/programs.js" type="module"></script>
<script src="js/favorites.js" type="module"></script>
<script src="js/app.js" type="module"></script>

</body>
</html>

```

## Листинг Г.2 – Файл styles.css

```

body {
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 0;
    background: url("../images/10.png");
    background-attachment: fixed;
}

html{
    height: 100%;
}

header {
    font-family: "Raleway", sans-serif;
    background: white;
    padding: 10px 20px;
}

```

```

display: flex;
justify-content: space-between;
align-items: center;
position: fixed;
top: 0;
left: 0;
width: 100%;
box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
z-index: 1000;
}

header h1 {
  text-shadow:
    0 0 black,
    0 0 #4242A6,
    0 0 #F23D70;
  color: transparent;
  transition: all 1s;
  text-align: left;
  font-size: 24px;
}

header h1:hover {
  text-shadow:
    -10px 10px black,
    0 0 #4242A6,
    10px -10px #F23D70;
}

#menu ul {
  list-style: none;
  display: flex;
  gap: 15px;
  margin: 0;
  padding: 0;
  white-space: nowrap;
}

#menu li a {
  color: black;
  text-decoration: none;
  font-size: 16px;
  transition: color 0.3s ease;
}

#menu li a:hover {
  color: #7C207C;
}

.container {
  max-width: 900px;
  margin: 0 auto;
}

```

```

    margin-top: 100px;
    padding: 20px;
}

#auth-container {
    max-width: 400px;
    margin: 50px auto;
    padding: 40px;
    border-radius: 8px;
    box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
}

#auth-container button{
    text-align: center;
    width: 50%;
}

input{
    padding:10px;
    border-radius:10px;
    border:0;
    border-bottom:1px solid #eee;
    box-shadow:0 0 15px 4px rgba(0,0,0,0.1);
    width: 100%;
    margin: 5px 0;
}

button {
    text-align: center;
    transition: 0.5s;
    background-size: 200% auto;
    height: 30px;
    color: black;
    box-shadow: 0 0 20px #eee;
    border-radius: 10px;
    background-image: linear-gradient(90deg, rgb(249, 236, 233) 21%,
rgb(155, 143, 144));
    margin: 5px auto;
    opacity: 0.8;
}

button:hover {
    background-position: center;
}

#logout-btn {
    all: unset;
    background-color: rgba(255, 255, 255, 0.5);
    color: black;
    border: none;
    padding: 5px 10px;
    font-size: 14px;
}

```

```

border-radius: 5px;
cursor: pointer;
transition: background-color 0.3s ease, color 0.3s ease;
}

select {
  appearance: none;
  -webkit-appearance: none;
  -moz-appearance: none;
  background-color: #fff;
  border: 1px solid #ccc;
  border-radius: 5px;
  padding: 10px;
  font-size: 16px;
  width: 100%;
  cursor: pointer;
  outline: none;
  transition: border-color 0.3s ease, box-shadow 0.3s ease;
}

select:focus {
  border-color: #4CAF50;
  box-shadow: 0 0 5px rgba(76, 175, 80, 0.5);
}

select::after {
  content: '▼';
  position: absolute;
  top: 50%;
  right: 10px;
  transform: translateY(-50%);
  pointer-events: none;
  font-size: 12px;
  color: #aaa;
}

textarea {
  width: 100%;
  padding: 10px;
  margin: 5px 0;
  border-radius: 5px;
  border: 1px solid #ccc;
  box-shadow: 0 0 15px 4px rgba(0,0,0,0.1);
}

.program {
  background-color: rgba(250, 250, 250, 0.2);
  padding: 15px;
  margin-bottom: 20px;
  border-radius: 8px;
  box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
}

```

```

.program h2 {
  margin: 0;
  color: #333;
}

.program p {
  color: #555;
}

.workout-list {
  margin-top: 10px;
  list-style-type: none;
  padding-left: 0;
}

.workout-item {
  list-style-type: none;
  background-color: rgba(240, 240, 240, 0.3);
  margin: 5px 0;
  padding: 8px;
  border-radius: 5px;
  box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
}

.workout-item span {
  color: #888;
}

.start-button {
  color: black;
  padding: 10px 20px;
  border: none;
  border-radius: 5px;
  cursor: pointer;
  text-align: center;
  display: block;
  width: 100%;
  margin-top: 10px;
}

#filter-container {
  margin-top: 20px;
}

#filter-container input {
  width: 100%;
  padding: 10px;
  margin: 5px 0;
  border-radius: 5px;
  border: 1px solid #ccc;
}

```

```

#video-container, #favorite-video-container {
  text-align: center;
  position: fixed;
  top: 0;
  left: 0;
  width: 100%;
  height: 100%;
  background: rgba(0, 0, 0, 0.8);
  z-index: 1000;
}

#rutube-video, #favorite-rutube-video {
  width: 80%;
  height: 80%;
  margin: auto;
}

#close-video, #close-favorite-video {
  position: absolute;
  top: 10px;
  right: 10px;
  padding: 10px;
  background: red;
  color: white;
  border: none;
  cursor: pointer;
}

video {
  max-width: 100%;
  border-radius: 10px;
}

.favorite-button {
  background-color: transparent;
  border: none;
  font-size: 20px;
  color: black;
  cursor: pointer;
  transition: all 0.3s ease;
}

.favorite-button.added {
  color: #ff4d4d;
}

.favorite-button:hover {
  transform: scale(1.2);
}

table {

```

```

        width: 100%;
        border-collapse: separate;
        margin: 20px 0;
        font-size: 18px;
        text-align: left;
    }

    #add-program-form,
    #edit-program-form,
    #edit-workout-form {
        position: fixed;
        top: 50%;
        left: 50%;
        transform: translate(-50%, -50%);
        background-color: white;
        padding: 40px;
        border-radius: 8px;
        box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
        z-index: 1000;
        width: 400px;
        max-width: 90%;
    }

    body.modal-open::before {
        content: '';
        position: fixed;
        top: 0;
        left: 0;
        width: 100%;
        height: 100%;
        background-color: rgba(0, 0, 0, 0.5);
        z-index: 999;
    }

    td, th {
        padding: 10px;
    }

```

### Листинг Г.3 – Файл app.js

```

import { loadProfile } from './profile.js';
import { loadWorkoutPrograms } from './programs.js';
import { loadFavoriteWorkouts } from './favorites.js';
import { loadPrograms } from './servPrograms.js';
import { loadServWorkouts } from './servWorkouts.js';

document.addEventListener("DOMContentLoaded", function () {
    const menuLinks = document.querySelectorAll('#menu a');

    menuLinks.forEach(link => {
        link.addEventListener('click', function (event) {
            event.preventDefault();
            const section = this.getAttribute('href').substring(1);

```

```

        loadSection(section);
    });
});

// Объект с контентом для каждого раздела
const sections = {
    profile: `
        <div id="profile-container">
            <h2>Личный кабинет</h2>
            <form id="profile-form">
                <p>Имя пользователя: <span id="profile-
username"></span></p>
                <p>Дата рождения: <input type="date" id="profile-
dateOfBirth" /></p>
                <p>Вес: <input type="number" id="profile-weight"
/></p>
                <p>Рост: <input type="number" id="profile-height"
/></p>
                <button type="button" id="updateButton">Обновить
данные</button>
            </form>
        </div>
        <div id="recommendations-container">
            <h3>Рекомендуемые тренировки:</h3>
            <form id="video-container" style="display:none;">
                <button id="close-video">Закрыть</button>
                <iframe id="rutube-video" style="display: block;"
allowfullscreen></iframe>
            </form>
            <select id="goal-type">
                <option value="weight-loss">Похудение</option>
                <option value="maintain">Поддержание формы</option>
            </select>
            <button id="get-recommendations">Получить
рекомендации</button>
            <ul id="recommended-workouts-list"></ul>
        </div>
        <canvas id="recommendation-chart" width="400"
height="200"></canvas>
    `,
    programs: `
        <div id="programs">
            <form id="filter-container">
                <!-- Фильтр для программ тренировок -->
                <label for="min-workouts">Минимальное количество
тренировок:</label>
                <input type="number" id="min-workouts"
placeholder="Введите минимальное количество" />

                <label for="keyword">Ключевое слово:</label>
                <input type="text" id="keyword" placeholder="Введите
ключевое слово" />
            </form>
        </div>
    `
};

```

```

        <!-- Фильтр для тренировок -->
        <label for="min-duration">Минимальная
продолжительность (мин):</label>
        <input type="number" id="min-duration"
placeholder="Введите минимальную продолжительность" />

        <button id="apply-filters">Применить
фильтры</button>
        <button id="clear-filters">Очистить фильтры</button>
    </form>

    <form id="video-container" style="display:none;">
        <button id="close-video">Закрыть</button>
        <iframe id="rutube-video" style="display: block;"
allowfullscreen></iframe>
    </form>

    <!-- Секция списка программ -->
    <form id="programs-list">
        <p>Loading...</p>
    </form>
</div>
,
favorites: `
    <div id="favorites">
        <h3>Избранные тренировки</h3>
        <ul id="favorite-workouts-list"></ul>

        <div id="favorite-video-container"
style="display:none;">
            <iframe id="favorite-rutube-video" style="display:
block;" allowfullscreen></iframe>
            <button id="close-favorite-video">Закрыть </button>
        </div>
    </div>
,
adminprograms: `
    <div id="admin-actions">
        <div id="program-list-container">
            <h2>Список программ тренировок</h2>
            <button id="add-program-btn">Добавить
программу</button>
            <table id="program-list" style="width: 100%; border-
collapse: collapse;">
                <thead>
                    <tr>
                        <th>ID</th>
                        <th>Название</th>
                        <th>Описание</th>
                        <th>Действия</th>
                    </tr>

```

```

        </thead>
        <tbody>
            <!-- Программы будут динамически добавляться
здесь -->
        </tbody>
    </table>
</div>

<!-- Форма добавления программы -->
<div id="add-program-form" style="display: none; ">
    <h3>Добавить программу</h3>
    <form id="program-form">
        <div id="add-error" style="color: red; margin-
bottom: 10px; display: none;">ошибка</div>
        <label for="program-name">Название:</label>
        <input type="text" id="program-name" required />
        <br />
        <label for="program-description">Описание:</label>
        <textarea id="program-description"
required></textarea>
        <br />
        <label for="workout-select">Выберите
тренировки:</label>
        <select id="workout-select" multiple style="width:
100%; height: 100px;">
            <!-- Опции будут динамически загружены -->
        </select>
        <br />
        <button type="submit">Сохранить</button>
        <button type="button" id="cancel-add-
program">Отмена</button>
    </form>
</div>

<!-- Форма редактирования программы -->
<div id="edit-program-form" style="display: none;">
    <h3>Редактировать программу</h3>
    <form id="edit-program-form-content">
        <input type="hidden" id="edit-program-id" />
        <div id="edit-error" style="color: red; margin-
bottom: 10px; display: none;"></div>
        <label for="edit-program-name">Название:</label>
        <input type="text" id="edit-program-name" required
/>
        <br />
        <label for="edit-program-
description">Описание:</label>
        <textarea id="edit-program-description"
required></textarea>
        <div>
            <h3>Добавить тренировки</h3>

```

```

        <label for="add-workout-select">Выберите
тренировки:</label>
        <select id="add-workout-select" multiple>
            <!-- Опции будут динамически загружены -->
        </select>
        <button id="add-workouts-btn">Добавить выбранные
тренировки</button>

        <h3>Удалить тренировки</h3>
        <label for="remove-workout-select">Выберите
тренировки:</label>
        <select id="remove-workout-select" multiple>
            <!-- Опции будут динамически загружены -->
        </select>
        <button id="remove-workouts-btn">Удалить
выбранные тренировки</button>
    </div>
    <button type="submit">Сохранить</button>
    <button type="button" id="cancel-edit-
program">Отмена</button>
    </form>
</div>
</div>
,
adminworkouts: `
    <div id="admin-workouts">
        <h2>Список тренировок</h2>
        <!-- Кнопка добавления тренировки -->
        <button id="add-workout-button">Добавить
тренировку</button>

        <!-- Форма добавления/редактирования тренировки -->
        <form id="edit-workout-form" style="display: none;">
            <h3>Добавить/Редактировать тренировку</h3>
            <input type="hidden" id="workout-id" />
            <label for="workout-name">Название:</label>
            <input type="text" id="workout-name" required />

            <label for="workout-description">Описание:</label>
            <textarea id="workout-description"
required></textarea>

            <label for="workout-duration">Продолжительность
(мин):</label>
            <input type="number" id="workout-duration" required
/>

            <label for="workout-metValue">MET:</label>
            <input type="number" step="0.1" id="workout-
metValue" required />

```

```

        <label for="workout-video-url">Ссылка на
видео:</label>
        <input type="url" id="workout-video-url"
placeholder="https://example.com/video.mp4" />

        <button type="submit">Сохранить</button>
        <button type="button" id="cancel-
edit">Отмена</button>
    </form>

    <!-- Список тренировок -->
    <table id="workouts-list">
        <thead>
            <tr>
                <th>ID</th>
                <th>Название</th>
                <th>Описание</th>
                <th>Продолжительность (мин)</th>
                <th>МЕТ</th>
                <th>Действия</th>
            </tr>
        </thead>
        <tbody>
            <!-- Строки с тренировками будут добавляться сюда
-->

            </tbody>
        </table>
    </div>
    ,
};

// Функция для загрузки контента
function loadSection(section) {
    const container = document.getElementById('content-
container');

    if (sections[section]) {
        container.innerHTML = sections[section];
        initializeSectionLogic(section);
    } else {
        container.innerHTML = '<p>Раздел не найден.</p>';
    }
}

// Функция для инициализации логики для каждого раздела
function initializeSectionLogic(section) {
    switch (section) {
        case 'profile':
            loadProfile();
            break;
        case 'programs':
            loadWorkoutPrograms();

```

```

        break;
    case 'favorites':
        loadFavoriteWorkouts();
        break;
    case 'adminprograms':
        loadPrograms();
        break;
    case 'adminworkouts':
        loadServWorkouts();
        break;
    default:
        console.log('Неизвестный раздел');
    }
}
});

```

#### Листинг Г.4 – Файл auth.js

```

document.addEventListener("DOMContentLoaded", function () {
    const apiUrl = 'http://localhost:5047/api';
    const authContainer = document.getElementById('auth-container');
    const loginForm = document.getElementById('login-form');
    const registerForm = document.getElementById('register-form');
    const switchToRegister = document.getElementById('switch-to-register');
    const switchToLogin = document.getElementById('switch-to-login');

    switchToRegister.addEventListener('click', (event) => {
        event.preventDefault();
        loginForm.style.display = 'none';
        registerForm.style.display = 'block';
    });

    switchToLogin.addEventListener('click', (event) => {
        event.preventDefault();
        registerForm.style.display = 'none';
        loginForm.style.display = 'block';
    });

    // Обработка формы регистрации
    document.getElementById('register-form').addEventListener('submit', async function (event) {
        event.preventDefault();
        const username = document.getElementById('register-username').value;
        const password = document.getElementById('register-password').value;
        const role = "user";

        try {
            const response = await fetch(`${apiUrl}/auth/register`,

```

```

        method: 'POST',
        headers: { 'Content-Type': 'application/json' },
        body: JSON.stringify({ username, password, role })
    });
    if (!response.ok) throw new Error('Ошибка регистрации.');
```

alert('Регистрация успешна!');

window.location.reload();

```

    } catch (error) {
        alert(`Ошибка: ${error.message}`);
    }
});

// Обработка формы авторизации
document.getElementById('login-form').addEventListener('submit',
async function (event) {
    event.preventDefault();
    const username = document.getElementById('login-
username').value;
    const password = document.getElementById('login-
password').value;

    try {
        const response = await fetch(`${apiBaseUrl}/auth/login`, {
            method: 'POST',
            headers: { 'Content-Type': 'application/json' },
            body: JSON.stringify({ username, password })
        });

        if (!response.ok) throw new Error('Ошибка авторизации.');
```

const data = await response.json();

localStorage.setItem('authToken', data.token);

localStorage.setItem('userId', data.userId);

if (Array.isArray(data.favoriteWorkouts)) {

const favoriteWorkoutIds =

data.favoriteWorkouts.map(workout => workout.id);

localStorage.setItem('favoriteWorkoutIds',

JSON.stringify(favoriteWorkoutIds));

} else{

localStorage.setItem('favoriteWorkoutIds', []);

}

alert('Вход выполнен успешно!');

checkAdmin();

document.getElementById('menu').style.display = 'block';

document.getElementById('logout-btn').style.display =

'block';

```

    } catch (error) {
        alert(`Ошибка: ${error.message}`);
        console.error('Login Error:', error);
    }
});

```

```

// Функция для проверки, является ли пользователь
администратором
async function checkAdmin() {
  const token = localStorage.getItem('authToken');
  const userId = localStorage.getItem('userId');

  if (!token || !userId) {
    alert("Пожалуйста, войдите в систему.");
    return;
  }
  try {
    const response = await
fetch(`${apiBaseUrl}/auth/${userId}`, {
    headers: {
      'Authorization': `Bearer ${token}`
    }
  });
    const user = await response.json();
    if (user.role === 'admin') {
      document.getElementById('adminp').style.display =
'block';
      document.getElementById('adminw').style.display =
'block';
      document.getElementById('profile').style.display =
'none';
      document.getElementById('fav').style.display = 'none';
      return;
    }
    if (user.role === 'user') {
      document.getElementById('adminp').style.display =
'none';
      document.getElementById('adminw').style.display =
'none';
      document.getElementById('profile').style.display =
'block';
      document.getElementById('fav').style.display = 'block';
      return;
    }
  } catch (error) {
    alert("Ошибка проверки пользователя.");
    console.error(error);
  }
}

document.getElementById('logout-
btn').addEventListener('click', () => {
  localStorage.removeItem('authToken');
  alert('Вы успешно вышли.');
```

document.getElementById('menu').style.display = 'none';

```

  window.location.reload();
});

```

```
});
});
```

### Листинг Г.5 – Файл favorites.js

```
import { addFavoriteWorkout } from './programs.js';
import { removeFavoriteWorkout } from './programs.js';
import { startWorkout } from './programs.js';
const apiUrl = 'http://localhost:5047/api';
const token = localStorage.getItem('authToken');
const userId = localStorage.getItem('userId');

export async function loadFavoriteWorkouts() {
  try {
    const userResponse = await
fetch(`${apiUrl}/auth/${userId}`, {
    headers: { 'Authorization': `Bearer ${token}` }
  });
    if (!userResponse.ok) throw new Error('Ошибка загрузки
данных пользователя.');
```

```
    const userData = await userResponse.json();
    const favoriteWorkouts = userData.favoriteWorkouts || [];
    const workoutsResponse = await
fetch(`${apiUrl}/workouts`, {
    headers: { 'Authorization': `Bearer ${token}` }
  });
    if (!workoutsResponse.ok) throw new Error('Ошибка загрузки
данных.');
```

```
    const allWorkouts = await workoutsResponse.json();
    displayFavoriteWorkouts(favoriteWorkouts);
  } catch (error) {
    alert(`Ошибка: ${error.message}`);
    console.error('Favorite Workouts Error:', error);
  }
}

function displayFavoriteWorkouts(favoriteWorkouts) {
  const favoriteWorkoutsList =
document.getElementById('favorite-workouts-list');
  favoriteWorkoutsList.innerHTML = '';

  if (!Array.isArray(favoriteWorkouts) ||
favoriteWorkouts.length === 0) {
    favoriteWorkoutsList.innerHTML = '<p>Нет избранных
тренировок.</p>';
    return;
  }

  favoriteWorkouts.forEach(workout => {
    const workoutItem = document.createElement('li');
    workoutItem.classList.add('workout-item');
    workoutItem.innerHTML = `
      ${workout.name} <span>${workout.duration}
мин)</span>
```

```

        <br>
        <br>
        ${workout.description}
    `;

    // Кнопка "Начать тренировку"
    const startButton = document.createElement('button');
    startButton.textContent = 'Начать тренировку';
    startButton.classList.add('start-button');
    startButton.onclick = function () {
        startWorkout(workout.name, workout.videoUrl,
'favorites-section');
    };
    workoutItem.appendChild(startButton);

    // Кнопка "Добавить/Удалить из избранного"
    const favoriteButton =
document.createElement('button');
    favoriteButton.classList.add('favorite-button');

    const userId = localStorage.getItem('userId');
    const favoriteWorkoutIds =
JSON.parse(localStorage.getItem('favoriteWorkoutIds') || '[]');
    favoriteButton.textContent =
favoriteWorkoutIds.includes(workout.id) ? '❤️' : '♡';

    if (favoriteWorkoutIds.includes(workout.id)) {
        favoriteButton.classList.add('added');
    }

    // Логика для добавления/удаления из избранного
    favoriteButton.onclick = async () => {
        if (favoriteWorkoutIds.includes(workout.id)) {
            await removeFavoriteWorkout(userId, workout.id);

favoriteWorkoutIds.splice(favoriteWorkoutIds.indexOf(workout.id),
1);

            favoriteButton.textContent = '♡';
            favoriteButton.classList.remove('added');
        } else {
            await addFavoriteWorkout(userId, workout.id);
            favoriteWorkoutIds.push(workout.id);
            favoriteButton.textContent = '❤️';
            favoriteButton.classList.add('added');
        }
        localStorage.setItem('favoriteWorkoutIds',
JSON.stringify(favoriteWorkoutIds));
        loadFavoriteWorkouts();
    };
    workoutItem.appendChild(favoriteButton);
    favoriteWorkoutsList.appendChild(workoutItem);
});

```

```

        document.getElementById('close-favorite-
video').addEventListener('click', () => {
            event.preventDefault();
            const videoContainer =
document.getElementById('favorite-video-container');
            const rutubeIframe = document.getElementById('favorite-
rutube-video');
            videoContainer.style.display = 'none';
            rutubeIframe.src = '';
        });
    }

```

### Листинг Г.6 – Файл profile.js

```

import { addFavoriteWorkout } from './programs.js';
import { removeFavoriteWorkout } from './programs.js';
import { startWorkout } from './programs.js';
const apiUrl = 'http://localhost:5047/api';
const token = localStorage.getItem('authToken');
export async function loadProfile() {
    const apiUrl = 'http://localhost:5047/api';
    const profileUsername = document.getElementById('profile-
username');
    const profileDateOfBirth =
document.getElementById('profile-dateOfBirth');
    const profileWeight = document.getElementById('profile-
weight');
    const profileHeight = document.getElementById('profile-
height');

    const token = localStorage.getItem('authToken');
    if (!token) {
        alert('Пожалуйста, войдите в систему.');
```

return;

```

    }

    const userId = localStorage.getItem('userId');

    try {
        const response = await
fetch(`${apiUrl}/auth/${userId}`, {
            headers: { 'Authorization': `Bearer ${token}` }
        });

        if (!response.ok) throw new Error('Ошибка загрузки
данных профиля.');
```

const profile = await response.json();

```

        profileUsername.textContent = profile.username;
        profileDateOfBirth.value = profile.dateOfBirth ?
profile.dateOfBirth.split('T')[0] : '';
        profileWeight.value = profile.weight || '';

```

```

        profileHeight.value = profile.height || '';
    } catch (error) {
        alert(`Ошибка: ${error.message}`);
        console.error('Fetch Error:', error);
    }
    profileLogic();
}

async function profileLogic(){
    const apiBaseUrl = 'http://localhost:5047/api';
    const profileUsername = document.getElementById('profile-username');
    const profileDateOfBirth =
document.getElementById('profile-dateOfBirth');
    const profileWeight = document.getElementById('profile-weight');
    const profileHeight = document.getElementById('profile-height');
    const videoContainer = document.getElementById('video-container');
    const closeButton = document.getElementById('close-video');

    // Функция для обновления профиля
    const updateButton =
document.getElementById('updateButton');
    updateButton.addEventListener('click', async function
(event) {
        event.preventDefault();

        const token = localStorage.getItem('authToken');
        const userId = localStorage.getItem('userId');
        const dateOfBirth = profileDateOfBirth.value;
        const weight = parseFloat(profileWeight.value);
        const height = parseFloat(profileHeight.value);

        if (!token) {
            alert('Пожалуйста, войдите в систему.');
```

```

        headers: {
            'Content-Type': 'application/json',
            'Authorization': `Bearer ${token}`
        },
        body: JSON.stringify(profileUpdateRequest)
    });

    if (!response.ok) throw new Error('Ошибка обновления
профиля.');
```

```

        alert('Профиль успешно обновлен!');
    } catch (error) {
        alert(`Ошибка: ${error.message}`);
        console.error('Update Profile Error:', error);
    }
});

document.getElementById('get-
recommendations').addEventListener('click', async function () {
    const goalType = document.getElementById('goal-
type').value;
    const userWeight = profileWeight.value || null;

    if (!userWeight) {
        alert('Пожалуйста, укажите ваш вес.');
```

```

        return;
    }
    try {
        const recommendedWorkouts = await
recommendWorkoutsBasedOnGoal(userWeight, goalType);
        const recommendedList =
document.getElementById('recommended-workouts-list');
        recommendedList.innerHTML = '';

        if (recommendedWorkouts.length === 0) {
            recommendedList.innerHTML = '<li>Нет подходящих
тренировок.</li>';
            return;
        }

        recommendedWorkouts.forEach(workout => {
            const workoutItem = document.createElement('li');
            workoutItem.classList.add('workout-item');
            workoutItem.innerHTML = `
                ${workout.name} <span>(${workout.duration}
мин)</span>
                <span>(Сожжено калорий:
${workout.caloriesBurned} ккал)</span>
                <br>
                ${workout.description}
            `;

```

```

        // Кнопка "Начать тренировку"
        const startButton =
document.createElement('button');
        startButton.textContent = 'Начать тренировку';
        startButton.classList.add('start-button');
        startButton.onclick = function () {
            event.preventDefault();
            startWorkout(workout.name, workout.videoUrl,
'programs');
        };
        workoutItem.appendChild(startButton);

        // Кнопка "Добавить/Удалить из избранного"
        const favoriteButton =
document.createElement('button');
        const userId = localStorage.getItem('userId');
        const favoriteWorkoutIds =
JSON.parse(localStorage.getItem('favoriteWorkoutIds') || '[]');

        favoriteButton.classList.add('favorite-button');
        favoriteButton.textContent =
favoriteWorkoutIds.includes(workout.id) ? '❤️' : '♡';

        if (favoriteWorkoutIds.includes(workout.id)) {
            favoriteButton.classList.add('added');
        }

        favoriteButton.onclick = async () => {
            event.preventDefault();
            if (favoriteWorkoutIds.includes(workout.id)) {
                await removeFavoriteWorkout(userId, workout.id);

                favoriteWorkoutIds.splice(favoriteWorkoutIds.indexOf(workout.id),
                1);

                favoriteButton.textContent = '♡';
                favoriteButton.classList.remove('added');
            } else {
                await addFavoriteWorkout(userId, workout.id);
                favoriteWorkoutIds.push(workout.id);
                favoriteButton.textContent = '❤️';
                favoriteButton.classList.add('added');
            }
            localStorage.setItem('favoriteWorkoutIds',
JSON.stringify(favoriteWorkoutIds));
        };
        workoutItem.appendChild(favoriteButton);
        recommendedList.appendChild(workoutItem);
    });
    generateRecommendationChart(recommendedWorkouts);
} catch (error) {
    console.error('Ошибка при получении рекомендаций:',
error);

```

```

        alert('Произошла ошибка при загрузке рекомендаций.');
```

```
    }
  });
```

```

    closeButton.addEventListener('click', () => {
      event.preventDefault();
      videoContainer.style.display = 'none';
      const rutubeIframe = document.getElementById('rutube-
video');
      rutubeIframe.src = '';
    });
  };

```

```

    async function recommendWorkoutsBasedOnGoal(userWeight,
goalType) {
      if (!userWeight) return [];
      const params = new URLSearchParams();
      params.append('userWeight', userWeight);
      params.append('goalType', goalType);
      try {
        const response = await
fetch(`${apiBaseUrl}/workouts/recommend?${params.toString()}`, {
          method: 'POST',
          headers: {
            'Content-Type': 'application/json',
            'Authorization': `Bearer ${token}`
          }
        });
      } catch (error) {
        if (!response.ok) throw new Error('Ошибка загрузки
данных.');
```

```

      return await response.json();
    } catch (error) {
      alert(`Ошибка: ${error.message}`);
      console.error('Ошибка при загрузке тренировок:', error);
      return [];
    }
  }

  function generateRecommendationChart(recommendedWorkouts) {
    const ctx = document.getElementById('recommendation-
chart').getContext('2d');
    ctx.fillStyle="black";
    new Chart(ctx, {
      type: 'bar',
      data: {
        labels: recommendedWorkouts.map(workout =>
workout.name),
        datasets: [{
          label: 'Сожженные калории (ккал)',
          data: recommendedWorkouts.map(workout =>
workout.caloriesBurned),
          backgroundColor: 'rgba(229, 43, 80, 0.6)'
        }]
      }
    });
  }

```

```

    }]
  },
  options: {
    responsive: true,
    scales: {
      y: { beginAtZero: true,
        ticks: {
          color: 'black',
          font: {
            size: 14
          }
        }
      },
      x: {
        ticks: {
          color: 'black'
        }
      }
    },
    plugins: {
      legend: {
        labels: {
          color: 'black',
          font: {
            size: 16
          }
        }
      }
    }
  }
});
}

```

### Листинг Г.7 – Файл programs.js

```

export async function loadWorkoutPrograms() {
  const apiBaseUrl = 'http://localhost:5047/api';
  const programsList = document.getElementById('programs-
list');
  const token = localStorage.getItem('authToken');
  if (!token) {
    programsList.innerHTML = '<p>Пожалуйста, войдите в
систему.</p>';
    return;
  }
  try {
    const response = await
fetch(`${apiBaseUrl}/workoutprograms`, {
      headers: { 'Authorization': `Bearer ${token}` }
    });
    if (!response.ok) throw new Error('Ошибка загрузки
данных.');
```

```

        displayPrograms(data);
        programsLogic();
    } catch (error) {
        alert(`Ошибка: ${error.message}`);
        localStorage.removeItem('authToken');
        console.error('Fetch Error:', error);
    }
}

// Функция для отображения программ тренировок
function displayPrograms(data) {
    const programsList = document.getElementById('programs-
list');
    if (data.length === 0) {
        programsList.innerHTML = '<p>Нет доступных программ
тренировок.</p>';
        return;
    }
    programsList.innerHTML = '';
    data.forEach(program => {
        const programDiv = document.createElement('div');
        programDiv.classList.add('program');

        const programTitle = document.createElement('h2');
        programTitle.textContent = program.name;

        const programDescription = document.createElement('p');
        programDescription.textContent = program.description;

        const workoutList = document.createElement('ul');
        workoutList.classList.add('workout-list');

        program.workouts.forEach(workout => {
            const workoutItem = document.createElement('li');
            workoutItem.classList.add('workout-item');
            workoutItem.innerHTML = `
                ${workout.name} <span>(${workout.duration}
мин)</span>
                <br>
                <br>
                ${workout.description}
            `;

            // Кнопка "Начать тренировку"
            const startButton = document.createElement('button');
            startButton.textContent = 'Начать тренировку';
            startButton.classList.add('start-button');
            startButton.onclick = function () {
                event.preventDefault();
                startWorkout(workout.name, workout.videoUrl,
'programs');
            };

```

```

workoutItem.appendChild(startButton);

// Кнопка "Добавить/Удалить из избранного"
const favoriteButton = document.createElement('button');
const userId = localStorage.getItem('userId');
const favoriteWorkoutIds =
JSON.parse(localStorage.getItem('favoriteWorkoutIds') || '[]');

favoriteButton.classList.add('favorite-button');
favoriteButton.textContent =
favoriteWorkoutIds.includes(workout.id) ? '❤️' : '♡';

if (favoriteWorkoutIds.includes(workout.id)) {
    favoriteButton.classList.add('added');
}

favoriteButton.onclick = async () => {
    event.preventDefault();
    if (favoriteWorkoutIds.includes(workout.id)) {
        await removeFavoriteWorkout(userId, workout.id);

        favoriteWorkoutIds.splice(favoriteWorkoutIds.indexOf(workout.id),
1);

        favoriteButton.textContent = '♡';
        favoriteButton.classList.remove('added');
    } else {
        await addFavoriteWorkout(userId, workout.id);
        favoriteWorkoutIds.push(workout.id);
        favoriteButton.textContent = '❤️';
        favoriteButton.classList.add('added');
    }
    localStorage.setItem('favoriteWorkoutIds',
JSON.stringify(favoriteWorkoutIds));
};

workoutItem.appendChild(favoriteButton);
workoutList.appendChild(workoutItem);
});

programDiv.appendChild(programTitle);
programDiv.appendChild(programDescription);
programDiv.appendChild(workoutList);
programsList.appendChild(programDiv);
});
}

// Добавление тренировки в избранное
export async function addFavoriteWorkout(userId, workoutId) {
    const apiUrl = 'http://localhost:5047/api';
    const token = localStorage.getItem('authToken');
    const params = new URLSearchParams();
    params.append('workoutId', workoutId);

```

```

    try {
      const response = await
fetch(`${apiBaseUrl}/auth/${userId}/favorites/add?${params.toStrin
g()}`, {
      method: 'POST',
      headers: {
        'Content-Type': 'application/json',
        'Authorization': `Bearer ${token}`
      }
    });
    if (!response.ok) throw new Error('Ошибка добавления в
избранное.');
```

alert('Тренировка добавлена в избранное.');

```

  } catch (error) {
    alert(`Ошибка: ${error.message}`);
    console.error('Add Favorite Error:', error);
  }
}

// Удаление тренировки из избранного
export async function removeFavoriteWorkout(userId, workoutId)
{
  const apiBaseUrl = 'http://localhost:5047/api';
  const token = localStorage.getItem('authToken');
  const params = new URLSearchParams();
  params.append('workoutId', workoutId);

  try {
    const response = await
fetch(`${apiBaseUrl}/auth/${userId}/favorites/remove?${params.toSt
ring()}`, {
    method: 'POST',
    headers: {
      'Content-Type': 'application/json',
      'Authorization': `Bearer ${token}`
    }
  });
  if (!response.ok) throw new Error('Ошибка удаления из
избранного.');
```

alert('Тренировка удалена из избранного.');

```

  } catch (error) {
    alert(`Ошибка: ${error.message}`);
    console.error('Remove Favorite Error:', error);
  }
}

// Функция начала тренировки
export function startWorkout(workoutName, videoUrl, sectionId)
{
  alert(`Начинаем тренировку: ${workoutName}`);
  const videoId = extractRuTubeVideoId(videoUrl);

```

```

    if (!videoId) {
        alert('Некорректный URL видео.');
```

return;

```
    }

    let rutubeIframe, videoContainer;
    if (sectionId === 'favorites-section') {
        rutubeIframe = document.getElementById('favorite-rutube-
video');
        videoContainer = document.getElementById('favorite-video-
container');
    } else {
        rutubeIframe = document.getElementById('rutube-video');
        videoContainer = document.getElementById('video-
container');
    }
    rutubeIframe.src = `https://rutube.ru/embed/${videoId}`;
    videoContainer.style.display = 'flex';
}

function extractRuTubeVideoId(url) {
    const regex = /(?:embed\/|video\/)([a-zA-Z0-9_-]+)/;
    const match = url.match(regex);
    return match ? match[1] : null;
}

async function programsLogic(){
    const apiUrl = 'http://localhost:5047/api';
    const programsList = document.getElementById('programs-
list');
    const videoContainer = document.getElementById('video-
container');
    const closeButton = document.getElementById('close-
video');

    document.getElementById('apply-
filters').addEventListener('click', function () {
        event.preventDefault();
        const minWorkoutsInput = document.getElementById('min-
workouts')?.value.trim();
        const keywordInput =
document.getElementById('keyword')?.value.trim();
        const minDurationInput = document.getElementById('min-
duration')?.value.trim();

        const minWorkouts = minWorkoutsInput ?
parseInt(minWorkoutsInput) : null;
        const keyword = keywordInput || null;
        const minDuration = minDurationInput ?
parseInt(minDurationInput) : null;

```

```

        const isValidMinWorkouts = minWorkouts !== null &&
!isNaN(minWorkouts);
        const isValidKeyword = keyword !== null &&
keyword.length > 0;
        const isValidMinDuration = minDuration !== null &&
!isNaN(minDuration);

        if (isValidMinWorkouts || isValidKeyword) {
            filterWorkoutPrograms(
                isValidMinWorkouts ? minWorkouts : null,
                isValidKeyword ? keyword : null
            );
        }
        if (isValidMinDuration) {
            filterWorkouts(null, minDuration);
        }
    });

    document.getElementById('clear-
filters').addEventListener('click', function (event) {
        event.preventDefault();
        document.getElementById('min-duration').value = '';
        document.getElementById('min-workouts').value = '';
        document.getElementById('keyword').value = '';
        loadWorkoutPrograms();
    });

    // Функция закрытия видео
    closeButton.addEventListener('click', () => {
        event.preventDefault();
        videoContainer.style.display = 'none';
        const rutubeIframe = document.getElementById('rutube-
video');
        rutubeIframe.src = '';
    });
}

async function filterWorkouts(minDifficulty, minDuration) {
    const apiBaseUrl = 'http://localhost:5047/api';
    const token = localStorage.getItem('authToken');
    const programsList = document.getElementById('programs-
list');
    const params = new URLSearchParams();

    if (minDuration) {
        params.append('minDuration', minDuration);
    }
    try {
        const response = await
fetch(`${apiBaseUrl}/workouts/filter?${params.toString()}`, {
            headers: { 'Authorization': `Bearer ${token}` }
        });
    }

```

```

        if (!response.ok) throw new Error('Ошибка фильтрации
данных.');
```

```

        const data = await response.json();

        if (Array.isArray(data) && data.length > 0) {
            displayFilteredWorkouts(data);
        } else {
            programsList.innerHTML = '<p>Нет подходящих
тренировок.</p>';
        }
    } catch (error) {
        alert(`Ошибка: ${error.message}`);
        console.error('Filter Error:', error);
    }
}

async function filterWorkoutPrograms(minWorkouts, keyword) {
    const apiBaseUrl = 'http://localhost:5047/api';
    const token = localStorage.getItem('authToken');
    const programsList = document.getElementById('programs-
list');
    const params = new URLSearchParams();

    if (minWorkouts) {
        params.append('minWorkouts', minWorkouts);
    }
    if (keyword) {
        params.append('keyword', keyword);
    }
    try {
        const response = await
fetch(`${apiBaseUrl}/workoutprograms/filter?${params.toString()}`,
{
            headers: { 'Authorization': `Bearer ${token}` }
        });
        if (!response.ok) throw new Error('Ошибка фильтрации
данных.');
```

```

        const data = await response.json();

        if (Array.isArray(data) && data.length > 0) {
            displayPrograms(data);
        } else {
            programsList.innerHTML = '<p>Нет подходящих программ
тренировок.</p>';
        }
    } catch (error) {
        alert(`Ошибка: ${error.message}`);
        console.error('Filter Error:', error);
    }
}

```

```

function displayFilteredWorkouts(filteredWorkouts) {
  const programsList = document.getElementById('programs-
list');
  if (!Array.isArray(filteredWorkouts) ||
filteredWorkouts.length === 0) {
    programsList.innerHTML = '<p>Нет подходящих программ
тренировок.</p>';
    return;
  }

  programsList.innerHTML = '';

  filteredWorkouts.forEach(workout => {
    const workoutItem = document.createElement('li');
    workoutItem.classList.add('workout-item');
    workoutItem.innerHTML = `
      ${workout.name} <span>${workout.duration}
мин)</span>
      <br>
      <br>
      ${workout.description}
    `;

    // Кнопка "Начать тренировку"
    const startButton = document.createElement('button');
    startButton.textContent = 'Начать тренировку';
    startButton.classList.add('start-button');
    startButton.onclick = function () {
      event.preventDefault();
      startWorkout(workout.name, workout.videoUrl,
'programs');
    };
    workoutItem.appendChild(startButton);

    // Кнопка "Добавить/Удалить из избранного"
    const favoriteButton = document.createElement('button');
    const userId = localStorage.getItem('userId');
    const favoriteWorkoutIds =
JSON.parse(localStorage.getItem('favoriteWorkoutIds') || '[]');

    favoriteButton.classList.add('favorite-button');
    favoriteButton.textContent =
favoriteWorkoutIds.includes(workout.id) ? '❤️' : '♡';

    if (favoriteWorkoutIds.includes(workout.id)) {
      favoriteButton.classList.add('added');
    }

    favoriteButton.onclick = async () => {
      event.preventDefault();
      if (favoriteWorkoutIds.includes(workout.id)) {

```

```

        await removeFavoriteWorkout(userId, workout.id);

favoriteWorkoutIds.splice(favoriteWorkoutIds.indexOf(workout.id),
1);

        favoriteButton.textContent = '♥';
        favoriteButton.classList.remove('added');
    } else {
        await addFavoriteWorkout(userId, workout.id);
        favoriteWorkoutIds.push(workout.id);
        favoriteButton.textContent = '♥';
        favoriteButton.classList.add('added');
    }
    localStorage.setItem('favoriteWorkoutIds',
JSON.stringify(favoriteWorkoutIds));
    };

    workoutItem.appendChild(favoriteButton);
    programsList.appendChild(workoutItem);
});
}

```

#### Листинг Г.8 – Файл servPrograms.js

```

const apiBaseUrl = 'http://localhost:5047/api';
const token = localStorage.getItem('authToken');

// Загрузка списка программ
export async function loadPrograms() {
    try {
        const response = await
fetch(`${apiBaseUrl}/workoutprograms`, {
            method: 'GET',
            headers: {
                'Authorization': `Bearer ${token}`
            }
        });
    };
    if (!response.ok) throw new Error('Ошибка загрузки
программ.');
```

```

        const programs = await response.json();
        renderProgramList(programs);
        loadWorkouts();
        servProgramsLogic();
    } catch (error) {
        alert(`Ошибка: ${error.message}`);
        console.error('Ошибка при загрузке программ:', error);
    }
}

// Отображение списка программ
function renderProgramList(programs) {
    const tbody = document.querySelector('#program-list tbody');
    tbody.innerHTML = '';

```

```

    programs.forEach(program => {
        const row = document.createElement('tr');
        row.innerHTML = `
            <td>${program.id}</td>
            <td>${program.name}</td>
            <td>${program.description}</td>
            <td>
                <button class="edit-btn" data-
id="${program.id}">Редактировать</button>
                <button class="delete-btn" data-
id="${program.id}">Удалить</button>
            </td>
        `;
        tbody.appendChild(row);
    });
    document.querySelectorAll('.edit-btn').forEach(button => {
        button.addEventListener('click', handleEditProgram);
    });

    document.querySelectorAll('.delete-btn').forEach(button => {
        button.addEventListener('click', handleDeleteProgram);
    });
}

// Показать форму добавления программы
function showAddProgramForm() {
    document.getElementById('add-program-form').style.display =
'block';
    document.body.classList.add('modal-open');
    loadWorkoutsForForm();
}

// Скрыть форму добавления программы
function hideAddProgramForm() {
    document.getElementById('add-program-form').style.display =
'none';
    document.body.classList.remove('modal-open');
}

// Показать форму редактирования программы
function showEditProgramForm(programId) {
    document.getElementById('edit-program-form').style.display =
'block';
    document.getElementById('edit-program-id').value = programId;
    document.body.classList.add('modal-open');
}

// Скрыть форму редактирования программы
function hideEditProgramForm() {
    document.getElementById('edit-program-form').style.display =
'none';
    document.body.classList.remove('modal-open');
}

```

```

    }

    function displayErrors(errors, errorDiv) {
        if (!errorDiv) {
            return;
        }
        errorDiv.innerHTML = '';
        errorDiv.style.display = 'none';
        if (errors.length > 0) {
            console.log('Обнаружены ошибки:', errors);
            errors.forEach(error => {
                const errorElement = document.createElement('p');
                errorElement.textContent = error;
                errorDiv.appendChild(errorElement);
            });
            errorDiv.style.display = 'block';
        }
    }

    function servProgramsLogic(){
        document.getElementById('add-program-
btn').addEventListener('click', showAddProgramForm);
        document.getElementById('cancel-add-
program').addEventListener('click', hideAddProgramForm);
        document.getElementById('cancel-edit-
program').addEventListener('click', hideEditProgramForm);
        document.getElementById('edit-program-form-
content').addEventListener('submit', async function (event) {
            event.preventDefault();

            const programId = document.getElementById('edit-program-
id').value;
            const programName = document.getElementById('edit-
program-name').value.trim();
            const programDescription =
document.getElementById('edit-program-description').value.trim();
            const errorDiv = document.getElementById('edit-error');

            if (!programName || !programDescription) {
                alert('Пожалуйста, заполните все поля.');
```

до 255 символов.");

```

                return;
            }

            let errors = [];
            if (programName.length < 2 || programName.length > 255)
            {
                errors.push("Длина названия программы должна быть от 2
                до 255 символов.");
            }

            const firstCharRegex = /^[A-Яa-яЁё]/;
            if (!firstCharRegex.test(programName)) {

```

```

        errors.push("Название должно начинаться с буквы.");
    }

    const allowedSymbolsRegex = /^[А-Яа-яЁё0-9\s.-]+$/;
    if (!allowedSymbolsRegex.test(programName)) {
        errors.push("Недопустимые символы в названии.
Разрешены только кириллические буквы, цифры, точки и дефисы.");
    }

    if (errors.length > 0) {
        displayErrors(errors, errorDiv);
        return false;
    }

    const requestBody = {
        id: parseInt(programId, 10),
        name: programName,
        description: programDescription
    };

    try {
        const response = await
fetch(`${apiBaseUrl}/workoutprograms/${programId}`, {
            method: 'PUT',
            headers: {
                'Content-Type': 'application/json',
                'Authorization': `Bearer ${token}`
            },
            body: JSON.stringify(requestBody)
        });

        if (!response.ok) {
            const errorData = await response.json();
            throw new Error(errorData.message || 'Ошибка
редактирования программы.');
```

}

```

        alert('Программа успешно отредактирована!');
        hideEditProgramForm();
        loadPrograms();
    } catch (error) {
        alert(`Ошибка: ${error.message}`);
        console.error('Ошибка при редактировании программы:',
error);
    }
});

// Обработка отправки формы добавления программы
document.getElementById('program-
form').addEventListener('submit', async function (event) {
    event.preventDefault();

```

```

        const programName = document.getElementById('program-
name').value.trim();
        const programDescription =
document.getElementById('program-description').value.trim();
        const select = document.getElementById('workout-
select');
        const selectedOptions =
Array.from(select.selectedOptions);
        const errorDiv = document.getElementById('add-error');

        if (!programName || !programDescription ||
selectedOptions.length === 0) {
            alert('Пожалуйста, заполните все поля.');
```

return;

```

        }
        let errors = [];
        if (programName.length < 2 || programName.length > 255)
{
            errors.push("Длина названия программы должна быть от 2
до 255 символов.");
        }

        const firstCharRegex = /^[А-Яа-яЁё]/;
        if (!firstCharRegex.test(programName)) {
            errors.push("Название должно начинаться с буквы.");
        }

        const allowedSymbolsRegex = /^[А-Яа-яЁё0-9\s.-]+$/;
        if (!allowedSymbolsRegex.test(programName)) {
            errors.push("Недопустимые символы в названии.
Разрешены только кириллические буквы, цифры, точки и дефисы.");
        }

        if (errors.length > 0) {
            displayErrors(errors, errorDiv);
            return false;
        }
        const workoutIds = selectedOptions.map(option =>
parseInt(option.value, 10));

        const requestBody = {
            name: programName,
            description: programDescription,
            workoutIds: workoutIds
        };

        try {
            const response = await
fetch(`${apiBaseUrl}/workoutprograms`, {
                method: 'POST',
                headers: {
```

```

        'Content-Type': 'application/json',
        'Authorization': `Bearer ${token}`
    },
    body: JSON.stringify(requestBody)
  });

  if (!response.ok) {
    const errorData = await response.json();
    throw new Error(errorData.message || 'Ошибка
добавления программы.');
```

```

  }

  alert('Программа успешно добавлена!');
  hideAddProgramForm();
  loadPrograms();
} catch (error) {
  alert(`Ошибка: ${error.message}`);
  console.error('Ошибка при добавлении программы:',
error);
}
});

document.getElementById('add-workouts-
btn').addEventListener('click', async () => {
  const programId = document.getElementById('edit-program-
id').value;
  const select = document.getElementById('add-workout-
select');
  const selectedOptions =
Array.from(select.selectedOptions);

  if (selectedOptions.length === 0) {
    alert('Пожалуйста, выберите хотя бы одну
тренировку.');
```

```

    return;
  }

  const workoutIds = selectedOptions.map(option =>
parseInt(option.value, 10));

  await addWorkoutsToProgram(programId, workoutIds);
  loadProgramWorkouts(programId);
});

document.getElementById('remove-workouts-
btn').addEventListener('click', async () => {
  const programId = document.getElementById('edit-program-
id').value;
  const select = document.getElementById('remove-workout-
select');
  const selectedOptions =
Array.from(select.selectedOptions);
```

```

        if (selectedOptions.length === 0) {
            alert('Пожалуйста, выберите хотя бы одну
тренировку.');
```

return;

```
        }

        const workoutIds = selectedOptions.map(option =>
parseInt(option.value, 10));

        await removeWorkoutsFromProgram(programId, workoutIds);
        loadProgramWorkouts(programId);
    });
}

// Обработка редактирования программы
async function handleEditProgram(event) {
    const programId = event.target.dataset.id;
    const program = await fetchProgramById(programId);
    document.getElementById('edit-program-name').value =
program.name;
    document.getElementById('edit-program-description').value =
program.description;
    showEditProgramForm(programId);
}

// Получение программы по ID
async function fetchProgramById(programId) {
    const response = await
fetch(`${apiBaseUrl}/workoutprograms/${programId}`, {
        method: 'GET',
        headers: {
            'Authorization': `Bearer ${token}`
        }
    });
    if (!response.ok) throw new Error('Ошибка загрузки
программы.');
```

return await response.json();

```

}

// Обработка удаления программы
async function handleDeleteProgram(event) {
    const programId = event.target.dataset.id;
    if (confirm('Вы уверены, что хотите удалить эту программу?'))
    {
        try {
            const response = await
fetch(`${apiBaseUrl}/workoutprograms/${programId}`, {
                method: 'DELETE',
                headers: {
                    'Authorization': `Bearer ${token}`
                }
            });

```

```

    });
    if (!response.ok) throw new Error('Ошибка удаления
программы.');
```

программы.');

```

    alert('Программа успешно удалена!');
    loadPrograms();
  } catch (error) {
    alert(`Ошибка: ${error.message}`);
    console.error('Ошибка при удалении программы:', error);
  }
}

async function loadWorkouts() {
  const apiBaseUrl = 'http://localhost:5047/api';
  const token = localStorage.getItem('authToken');

  try {
    const response = await fetch(`${apiBaseUrl}/workouts`, {
      method: 'GET',
      headers: {
        'Authorization': `Bearer ${token}`
      }
    });

    if (!response.ok) throw new Error('Ошибка загрузки
тренировок.');
```

тренировок.');

```

    const workouts = await response.json();

    const addSelect = document.getElementById('add-workout-
select');
    const removeSelect = document.getElementById('remove-
workout-select');

    workouts.forEach(workout => {
      const option = document.createElement('option');
      option.value = workout.id;
      option.textContent = workout.name;

      addSelect.appendChild(option.cloneNode(true));
      removeSelect.appendChild(option);
    });
  } catch (error) {
    alert(`Ошибка: ${error.message}`);
    console.error('Ошибка при загрузке тренировок:', error);
  }
}

async function addWorkoutsToProgram(programId, workoutIds) {
  const apiBaseUrl = 'http://localhost:5047/api';
  const token = localStorage.getItem('authToken');
  const params = new URLSearchParams();

```

```

    try {
      for (const workoutId of workoutIds) {
        params.append('workoutId', workoutId);
        const response = await
fetch(`${apiBaseUrl}/workoutprograms/${programId}/add?${params.toS
tring()}`, {
          method: 'POST',
          headers: {
            'Content-Type': 'application/json',
            'Authorization': `Bearer ${token}`
          }
        });

        if (!response.ok) {
          const errorData = await response.json();
          throw new Error(errorData.message || `Ошибка
добавления тренировки с ID ${workoutId}.`);
        }
      }

      alert('Все тренировки успешно добавлены в программу.');
```

```

    } catch (error) {
      alert(`Ошибка: ${error.message}`);
      console.error('Add Workouts Error:', error);
    }
  }

  async function removeWorkoutsFromProgram(programId,
workoutIds) {
    const apiBaseUrl = 'http://localhost:5047/api';
    const token = localStorage.getItem('authToken');
    const params = new URLSearchParams();

    try {
      for (const workoutId of workoutIds) {
        params.append('workoutId', workoutId);
        const response = await
fetch(`${apiBaseUrl}/workoutprograms/${programId}/remove?${params.
toString()}`, {
          method: 'POST',
          headers: {
            'Content-Type': 'application/json',
            'Authorization': `Bearer ${token}`
          }
        });

        if (!response.ok) {
          const errorData = await response.json();
          throw new Error(errorData.message || `Ошибка удаления
тренировки с ID ${workoutId}.`);
        }
      }
    }
  }

```

```

    }

    alert('Все тренировки успешно удалены из программы.');
```

```

  } catch (error) {
    alert(`Ошибка: ${error.message}`);
    console.error('Remove Workouts Error:', error);
  }
}

async function loadWorkoutsForForm() {
  const apiUrl = 'http://localhost:5047/api';
  const token = localStorage.getItem('authToken');

  try {
    const response = await fetch(`${apiUrl}/workouts`, {
      method: 'GET',
      headers: {
        'Authorization': `Bearer ${token}`
      }
    });

    if (!response.ok) throw new Error('Ошибка загрузки тренировок.');
```

```

    const workouts = await response.json();

    const select = document.getElementById('workout-select');
    select.innerHTML = '';

    workouts.forEach(workout => {
      const option = document.createElement('option');
      option.value = workout.id;
      option.textContent = workout.name;
      select.appendChild(option);
    });
  } catch (error) {
    alert(`Ошибка: ${error.message}`);
    console.error('Ошибка при загрузке тренировок:', error);
  }
}

```

### Листинг Г.9 – Файл servWorkouts.js

```

const apiUrl = 'http://localhost:5047/api';
const token = localStorage.getItem('authToken');
// Функция для загрузки тренировок
export async function loadServWorkouts() {
  try {
    const response = await fetch(`${apiUrl}/workouts`, {
      headers: { 'Authorization': `Bearer ${token}` }
    });
    if (!response.ok) throw new Error('Ошибка загрузки данных.');
```

```

const workouts = await response.json();
const tbody = document.querySelector('#workouts-list
tbody');
tbody.innerHTML = '';

workouts.forEach(workout => {
  const row = document.createElement('tr');
  row.innerHTML = `
    <td>${workout.id}</td>
    <td>${workout.name}</td>
    <td>${workout.description}</td>
    <td>${workout.duration}</td>
    <td>${workout.metValue}</td>
    <td>
      <button class="edit-btn" data-id="${workout.id}"
data-name="${workout.name}" data-
description="${workout.description}" data-
duration="${workout.duration}" data-metValue="${workout.metValue}"
data-video-url="${workout.videoUrl}">Редактировать</button>
      <button class="delete-btn" data-
id="${workout.id}">Удалить</button>
    </td>
  `;
  tbody.appendChild(row);
});

document.querySelectorAll('.edit-btn').forEach(button => {
  button.addEventListener('click', () => {
    const id = button.dataset.id;
    const name = button.dataset.name;
    const description = button.dataset.description;
    const duration = button.dataset.duration;
    const metValue = button.dataset.metValue;
    const videoUrl = button.dataset.videoUrl;
    editWorkout(id, name, description, duration, metValue,
videoUrl);
  });
});

document.querySelectorAll('.delete-btn').forEach(button =>
{
  button.addEventListener('click', () => {
    const id = button.dataset.id;
    deleteWorkout(id);
  });
});
} catch (error) {
  alert(`Ошибка: ${error.message}`);
  console.error('Workouts Error:', error);
}
servWorkoutsLogic();
}

```

```

    async function editWorkout(id, name, description, duration,
metValue, videoUrl) {
        document.getElementById('workout-id').value = id;
        document.getElementById('workout-name').value = name;
        document.getElementById('workout-description').value =
description;
        document.getElementById('workout-duration').value =
duration;
        document.getElementById('workout-metValue').value =
metValue;
        document.getElementById('workout-video-url').value =
videoUrl;
        document.getElementById('edit-workout-form').style.display =
'block';
        document.body.classList.add('modal-open');
    }

    function servWorkoutsLogic(){
        const workoutsTable = document.getElementById('workouts-
list').querySelector('tbody');
        const addWorkoutForm = document.getElementById('edit-
workout-form');
        document.getElementById('add-workout-
button').addEventListener('click', () => {
            event.preventDefault();
            document.getElementById('workout-id').value = '';
            document.getElementById('workout-name').value = '';
            document.getElementById('workout-description').value =
'';
            document.getElementById('workout-duration').value = '';
            document.getElementById('workout-metValue').value = '';
            document.getElementById('workout-video-url').value = '';
            document.getElementById('edit-workout-
form').style.display = 'block';
            document.body.classList.add('modal-open');
        });

        document.getElementById('edit-workout-
form').addEventListener('submit', async function (event) {
            event.preventDefault();
            const id = document.getElementById('workout-id').value
|| null;
            const name = document.getElementById('workout-
name').value;
            const description = document.getElementById('workout-
description').value;
            const duration =
parseInt(document.getElementById('workout-duration').value);
            const metValue =
parseFloat(document.getElementById('workout-metValue').value);

```

```

        const videoUrl = document.getElementById('workout-video-
url').value;
        const token = localStorage.getItem('authToken');

        const requestBody = {
            name: name,
            description: description,
            duration: duration,
            metValue: metValue,
            videoUrl: videoUrl
        };
        if (id) {
            requestBody.id = id;
        }

        const url = id ? `${apiBaseUrl}/workouts/${id}` :
`${apiBaseUrl}/workouts`;
        const method = id ? 'PUT' : 'POST';

        try {
            const response = await fetch(url, {
                method: method,
                headers: {
                    'Content-Type': 'application/json',
                    'Authorization': `Bearer ${token}`
                },
                body: JSON.stringify(requestBody)
            });
            if (!response.ok) throw new Error('Ошибка сохранения
тренировки. ');
            alert(id ? 'Тренировка успешно обновлена!' :
'Тренировка успешно добавлена!');
            document.getElementById('edit-workout-
form').style.display = 'none';
            document.body.classList.remove('modal-open');
            loadServWorkouts();
        } catch (error) {
            alert(`Ошибка: ${error.message}`);
            console.error('Ошибка при сохранении тренировки:',
error);
        }
    });

    document.getElementById('cancel-
edit').addEventListener('click', () => {
        document.getElementById('edit-workout-
form').style.display = 'none';
        document.body.classList.remove('modal-open');
    });
}

// Функция для удаления тренировки

```

```

    async function deleteWorkout(workoutId) {
        const token = localStorage.getItem('authToken');
        try {
            const response = await
fetch(`${apiBaseUrl}/workouts/${workoutId}`, {
            method: 'DELETE',
            headers: {
                'Authorization': `Bearer ${token}`
            }
        });

            if (!response.ok) throw new Error('Ошибка удаления
тренировки.');
```

```

            alert('Тренировка успешно удалена!');
        } catch (error) {
            alert(`Ошибка: ${error.message}`);
            console.error('Ошибка при удалении тренировки:', error);
        }
        loadServWorkouts();
    }

```

**Приложение Д**  
**(рекомендуемое)**  
**Результаты функционального тестирования**

## Редактировать программу

Длина названия программы должна быть от 2 до 255 символов.

Название:

й

Рисунок Д.1 – Результаты тестирования класса эквивалентности «Длина меньше 2»

## Редактировать программу

Длина названия программы должна быть от 2 до 255 символов.

Название:

Йогадлнддънрьлилдвпдв Аропоыаподжопдтпдлоп Опвпарпвроп

Рисунок Д.2 – Результаты тестирования класса эквивалентности «Длина больше 255»

## Редактировать программу

Название должно начинаться с буквы.

Название:

.Теннис

Рисунок Д.3 – Результаты тестирования класса эквивалентности «Название начинается не с буквы»

## Редактировать программу

Недопустимые символы в названии. Разрешены только кириллические буквы, цифры, точки и дефисы.

Название:

Фитнес@

Рисунок Д.4 – Результаты тестирования класса эквивалентности «Название содержит недопустимые символы»

## Редактировать программу

Название должно начинаться с буквы.

Название:

3215

Рисунок Д.5 – Результаты тестирования класса эквивалентности «Название не содержит буквы»

**Приложение Е**  
**(рекомендуемое)**  
**Листинг модульных тестов**

```

using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Testing;
using Microsoft.EntityFrameworkCore;
using Moq;
using SportApi.Controllers;
using SportApi.Models;
using SportApi.Services;
using System.Net.Http.Json;
using System.Net;
using Xunit;
using Microsoft.AspNetCore.Identity.Data;
using Azure;

namespace SportApi.Tests
{
    public class AuthControllerTests :
    IClassFixture<WebApplicationFactory<Program>>
    {
        private readonly WebApplicationFactory<Program> _factory;

        public AuthControllerTests(WebApplicationFactory<Program>
factory)
        {
            _factory = factory;
        }

        private HttpClient CreateClientWithToken(string token)
        {
            var client = _factory.CreateClient();
            client.DefaultRequestHeaders.Add("Authorization",
"Bearer " + token);
            return client;
        }

        [Fact]
        public async Task RegisterOk()
        {
            var client = _factory.CreateClient();

            var request = new
            {
                username = "testuser127",
                password = "TestPassword123",
                role = "user"
            };

            var response = await
client.PostAsJsonAsync("/api/auth/register", request);
            var content = await
response.Content.ReadAsStringAsync();
            Console.WriteLine(content);
        }
    }
}

```

```

        Assert.Equal(HttpStatusCode.OK, response.StatusCode);
    }

    [Fact]
    public async Task LoginReturnToken()
    {
        var client = _factory.CreateClient();

        // Register the user first
        var loginRequest = new
SportApi.Controllers.AuthController.LoginRequest
        {
            username = "testuser123",
            password = "TestPassword123"
        };

        var loginResponse = await
client.PostAsJsonAsync("/api/auth/login", loginRequest);
        Assert.Equal(HttpStatusCode.OK,
loginResponse.StatusCode);

        var content = await
loginResponse.Content.ReadAsStringAsync();
        Assert.Contains("token", content);
    }

    [Fact]
    public async Task UpdateOk()
    {
        var client = _factory.CreateClient();

        var loginRequest = new
        {
            username = "testuser123",
            password = "TestPassword123"
        };

        var loginResponse = await
client.PostAsJsonAsync("/api/auth/login", loginRequest);
        Assert.Equal(HttpStatusCode.OK,
loginResponse.StatusCode);

        var content = await
loginResponse.Content.ReadAsStringAsync();
        var token = content.Split(' ')[3];

        // Now update the profile
        var clientWithToken = CreateClientWithToken(token);

        var userId = 7;

        var updateRequest = new

```

```

        {
            dateOfBirth = "1990-01-01",
            weight = 70,
            height = 180
        };

        var updateResponse = await
clientWithToken.PostAsJsonAsync($"/api/auth/{userId}/update-
profile", updateRequest);
        Assert.Equal(HttpStatusCode.OK,
updateResponse.StatusCode);
    }

    [Fact]
    public async Task GetProfile_ReturnsOk()
    {
        var client = _factory.CreateClient();

        var loginRequest = new
        {
            username = "testuser123",
            password = "TestPassword123"
        };

        var loginResponse = await
client.PostAsJsonAsync("/api/auth/login", loginRequest);
        var loginContent = await
loginResponse.Content.ReadAsStringAsync();
        var token = loginContent.Split(' ')[3];

        var clientWithToken = CreateClientWithToken(token);

        var userId = 7;

        var response = await
clientWithToken.GetAsync($"/api/auth/{userId}");

        Assert.Equal(HttpStatusCode.OK, response.StatusCode);

        var content = await
response.Content.ReadAsStringAsync();

        Assert.Contains("username", content);
        Assert.Contains("testuser123", content);
    }

    [Fact]
    public async Task AddFavoriteWorkout_ReturnsOk()
    {
        // Arrange: Регистрация пользователя и получение
токена
        var client = _factory.CreateClient();

```

```

        var loginRequest = new
        {
            username = "testuser123",
            password = "TestPassword123"
        };

        var loginResponse = await
client.PostAsJsonAsync("/api/auth/login", loginRequest);
        var loginContent = await
loginResponse.Content.ReadAsStringAsync();
        var token = loginContent.Split(' ')[3];

        var clientWithToken = CreateClientWithToken(token);

        var userId = 7;
        var workoutId = 1;

        var response = await
clientWithToken.PostAsync($"/api/auth/{userId}/favorites/add?workoutId={workoutId}", null);

        Assert.Equal(HttpStatusCode.OK, response.StatusCode);
    }

    [Fact]
    public async Task RemoveFavoriteWorkout_ReturnsOk()
    {
        var client = _factory.CreateClient();

        var loginRequest = new
        {
            username = "testuser123",
            password = "TestPassword123"
        };

        var loginResponse = await
client.PostAsJsonAsync("/api/auth/login", loginRequest);
        var loginContent = await
loginResponse.Content.ReadAsStringAsync();
        var token = loginContent.Split(' ')[3];

        var clientWithToken = CreateClientWithToken(token);

        var userId = 7;
        var workoutId = 1;

        var response = await
clientWithToken.PostAsync($"/api/auth/{userId}/favorites/remove?workoutId={workoutId}", null);

        Assert.Equal(HttpStatusCode.OK, response.StatusCode);
    }

```

}  
}  
}