**Реферат**

Курсовая работа содержит в себе 23 страниц, 7 рисунков, 8 источников, 1 приложение.

ПРИЛОЖЕНИЕ, ДИАГРАММЫ ВАРИАНТОВ ИСПОЛЬЗОВАНИЯ, ДИАГРАММЫ КЛАССОВ, МЕТОДЫ, КЛАССЫ, ТЕХНИЧЕСКОЕ ЗАДАНИЕ, ТЕСТИРОВАНИЕ ПРОДУКТА, АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ, ПРОГРАММНАЯ ИНЖЕНЕРИЯ.

Цель курсовой работы – спроектировать и разработать эффективное, масштабируемое и удобное для пользователя приложение для оператора такси, применяя современные методологии и инструменты программной инженерии.

Объектами исследования курсовой работы являются методологии разработки программного обеспечения, применимые к созданию приложении, архитектурные паттерны и принципы проектирования приложений, технологии и фреймворки для разработки кроссплатформенных приложений, методы проектирования и реализации пользовательских интерфейсов мобильных приложений и системы управления базами данных и методы работы с данными в приложениях.

**Содержание**

[Введение 3](#_Toc175502020)

[1 Нормативные ссылки 4](#_Toc175502021)

[2 Постановка и формулировка задачи 5](#_Toc175502022)

[3 Анализ предметной области 5](#_Toc175502023)

[3.1 Назначение приложения и предметная область 5](#_Toc175502024)

[3.2 Форма реализации приложения 6](#_Toc175502025)

[3.3 Диаграмма вариантов использования приложения 8](#_Toc175502026)

[3.4 Диаграмма компонентов приложения 10](#_Toc175502027)

[3.5. Логическая модель данных для приложения 11](#_Toc175502028)

[4 Разработка технического задания 12](#_Toc175502029)

[5 Разработка пользовательского интерфейса 14](#_Toc175502030)

[5.1 Вход и регистрация пользователя 15](#_Toc175502031)

[5.2 Начальная страница приложения 16](#_Toc175502032)

[6 Проектирование классов 17](#_Toc175502033)

[7 Результаты тестирования программы 19](#_Toc175502034)

[Заключение 21](#_Toc175502035)

[Список использованных источников 22](#_Toc175502036)

[Приложение А Тест на анти плагиат 23](#_Toc175502037)

# **Введение**

**Целью курсовой работы является**, спроектировать и разработать эффективное, масштабируемое и удобное для пользователя приложение для тренажерного зала, применяя современные методологии и инструменты программной инженерии.

**Объектами исследования** настоящей курсовой работы являются:

* существующие приложения для тренажерных залов
* методологии разработки программного обеспечения
* современные инструменты программной инженерии
* требования и потребности для пользователей тренажерного зала

**Предметами исследования** настоящей курсовой работы являются:

* архитектура эффективного и масштабируемого приложения для тренажерных залов
* методы обеспечения удобства использования (user-friendly interface) в приложениях
* процессы проектирования и разработки приложения с учетом современных методологий
* способы интеграции инструментов программной инженерии в процесс создания приложения

**Основными задачи курсовой работы являются:**

* разработать диаграмму вариантов использования приложения;
* разработать диаграмму классов;
* разработать логическую модель данных;
* разработать диаграмму компонентов;

# **Нормативные ссылки**

В настоящей курсовой работе были использованы ссылки на следующие руководящие документы:

– ГОСТ Р 7.0.5-2008 СИБИД. Библиографическая ссылка. Общие требования и правила составления;

– ГОСТ Р 7.0.108-2022 СИБИД. Библиографические ссылки на электронные документы, размещенные в информационно-телекоммуникационных сетях. Общие требования к составлению и оформлению;

– ГОСТ Р ИСО 9241-210-2016 Эргономика взаимодействия человек – система;

– ГОСТ 7.82-2001 СИБИД. Библиографическая запись. Библиографическое описание электронных ресурсов. Общие требования и правила составления;

– ГОСТ 19.101-77 Единая система программной документации. Общие положения;

– ГОСТ 19.401-78 ЕСПД. Текст программы. Требования к содержанию и оформлению;

– ГОСТ 19.103-77 ЕСПД. Обозначение программ и программных продуктов;

– ГОСТ 34.602-89 «Техническое задание на создание автоматизированной системы»

# **Постановка и формулировка задачи**

Для изучения предмета «Программная Инженерия» была взята следующая задача для реализации: «Разработка приложения для тренажерного залов».

# **Анализ предметной области**

Анализ предметной области - это важный этап в процессе разработки программного обеспечения, который предшествует непосредственному проектированию и реализации системы. Этот процесс включает в себя изучение и описание той сферы деятельности, для которой создается программное обеспечение.

## **3.1 Назначение приложения и предметная область**

Приложение для твоего тренажерного зала - это ключ к эффективному управлению вашим бизнесом и максимальному удобству для клиентов:

* простая и удобная регистрация с возможностью заполнения профиля, выбора предпочитаемых типов тренировок;
* доступ к расписанию групповых занятий, тренажерного зала, персональных тренировок;
* быстрая и удобная онлайн-запись на понравившиеся занятия;
* доступ к подробному расписанию всех занятий с фильтрацией;

**Назначение и цель разработки** приложения для оператора такси заключаются в создании эффективного, удобного и современного инструмента, который оптимизирует записи на занятие, и отслеживание текущих занятий пользователям тренажерного зала, повысит качество обслуживания пользователей и предоставит дополнительные возможности как для пользователи, так и для сотрудников тренажерного зала.

Основные аспекты разрабатываемого приложения:

* удобство для клиентов - простой и интуитивный интерфейс, возможность онлайн-записи на занятия, отмены записи, выбора типа тренировок, просмотра расписания с фильтрацией, получения push-уведомлений.
* эффективное управление - автоматизация процессов, упрощение работы администраторов, возможность отслеживать посещаемость, популярность занятий и других метрик, что позволяет принимать взвешенные решения по оптимизации работы тренажерного зала.
* повышение эффективности - снижение времени, затраченного на ручные операции, что позволяет персоналу сосредоточиться на улучшении качества обслуживания клиентов.).

## **3.2 Форма реализации приложения**

Приложение будет реализовано в виде web-приложения (на React.js) для доступа через браузер на различных устройства, что позволит обеспечит максимальный охват аудитории и удобство использования.

База данных приложения реализована на PostgreSQL, так она обеспечивает:

* надежность и целостность данных - PostgreSQL обеспечивает полное соответствие ACID (Атомарность, Согласованность, Изолированность, Долговечность). Поддерживает транзакции, что критично для финансовых операций при продаже билетов.
* масштабируемость - хорошо справляется с большими объемами данных, что важно для растущего бизнеса по продаже билетов. Поддерживает горизонтальное масштабирование через репликацию и шардинг.
* производительность - эффективно обрабатывает сложные запросы, что полезно при поиске и фильтрации мероприятий. Имеет продвинутый планировщик запросов для оптимизации производительности.
* поддержка JSON - Позволяет хранить и обрабатывать JSON-данные, что удобно для хранения динамической информации о мероприятиях и билетах.

Его возможности хорошо соответствуют требованиям такого типа приложений, обеспечивая надежную работу и возможности для будущего роста и развития системы.

На рынке существует несколько аналогичных систем, которые можно рассмотреть:

* uber dispatchery - глобальная система управления для партнеров Uber, предоставляет инструменты для управления автопарком и водителями
* mindbody - Популярная платформа, предлагающая комплексную систему управления фитнес-клубом, включающую онлайн-запись, управление расписанием, обработку платежей, marketing инструменты и аналитику.
* glofox - Флагманская система управления фитнес-клубом, фокусирующаяся на мощных маркетинговых функциях, CRM, онлайн-записи и аналитике.
* zen Planner - Простой в использовании, но мощный инструмент для управления фитнес-клубами, предлагающий платежные функции, онлайн-запись, управление расписанием и отчеты.
* booker - Платформа, ориентированная на управление расписанием, онлайн-записью и автоматизацией процессов, позволяющая клиентам бронировать занятия, оплачивать услуги и получать push-уведомления.

Перечень потребностей пользователя, удовлетворяемых приложением:

* удобный просмотр информации о всех возможных тренеровок;
* удобный просмотр информации о расписаниях тренировок;

Это приложение для тренажерного зала призвано оптимизировать процесс управления записи и отмена тренировкам, сделать его более эффективным и удобным как для сотрудников тренажерного зала, так и для пользователей, предоставляя широкий спектр функций для удовлетворения потребностей всех участников процесса тренеровок.

## **3.3 Диаграмма вариантов использования приложения**

Диаграмма вариантов использования (use case diagram) является важным инструментом в разработке программного обеспечения, особенно на этапе анализа и проектирования.

Диаграмма вариантов использования - это диаграмма, используемая в UML (Unified Modeling Language), которая показывает функциональность системы с точки зрения пользователей. Она визуально представляет различные сценарии использования системы и взаимодействия между пользователями (акторами) и системой.

Ключевые элементы диаграммы вариантов использования:

* актер (Actor) - Представляет внешнюю сущность, взаимодействующую с системой, может быть человеком, другой системой или внешним устройство и изображается в виде стилизованной фигурки человека;
* вариант использования (Use Case) - представляет конкретную функцию или сценарий использования системы. Изображается в виде эллипса с названием внутри;
* система (System) - представляет границы разрабатываемой системы, обычно изображается в виде прямоугольника, содержащего варианты использования;
* связи – ассоциация это связь между актером и вариантом использования, включение (include) это один вариант использования включает функциональность другого. Расширение (extend) это один вариант использования может быть расширен другим при определенных условиях. Обобщение это отношение наследования между актерами или вариантами использования.

На рисунке 1 показана диаграмма использования для приложения тренажерного зала.

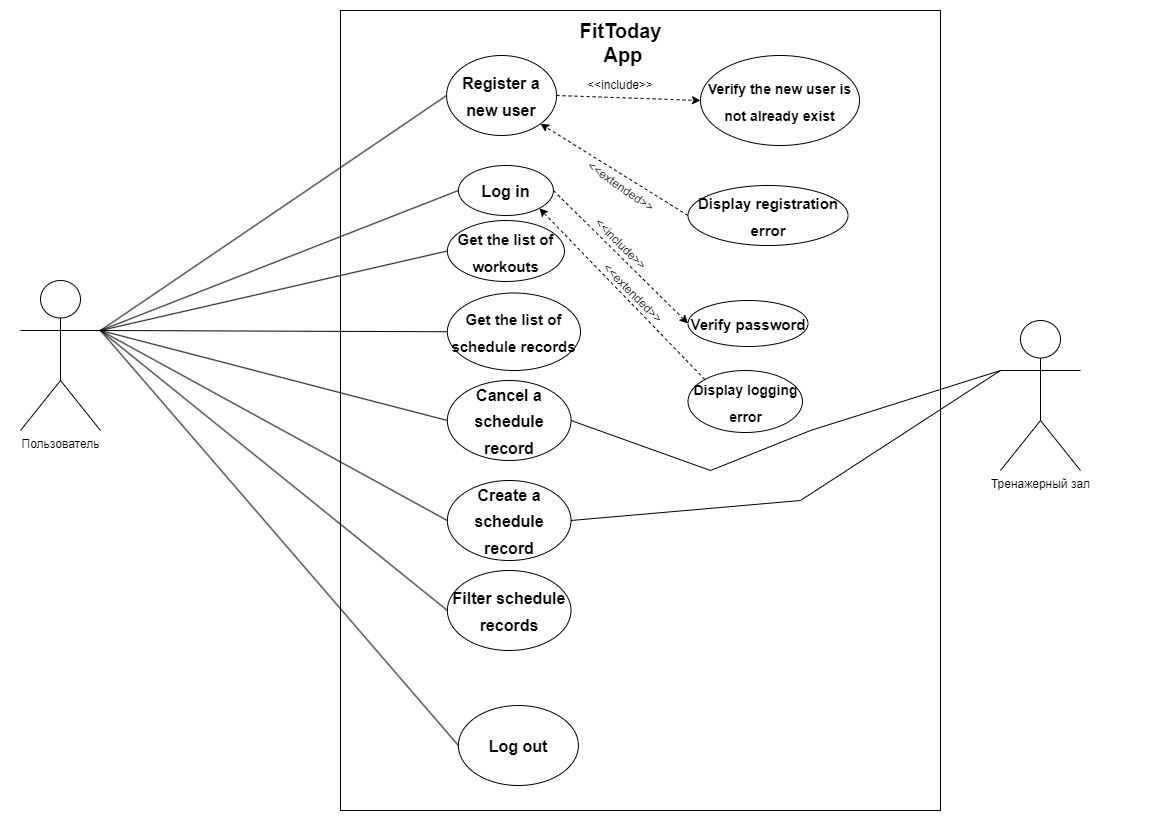


Рисунок 1 - диаграмма использования для приложения тренажерного зала

## **3.4 Диаграмма компонентов приложения**

Диаграмма компонентов - это структурная диаграмма в UML (Unified Modeling Language), которая показывает организацию и взаимосвязи между компонентами в системе.

Ключевые элементы диаграммы компонентов:

* компонент представляет модульную часть системы с четко определенным интерфейсом.
* интерфейс определяет набор операций, которые компонент предоставляет или требует.
* порт точка взаимодействия компонента с внешним миром.
* коннектор представляет связь между компонентами.

На рисунке 2 показана диаграмма компонентов для приложения для приложения тренажерного зала.

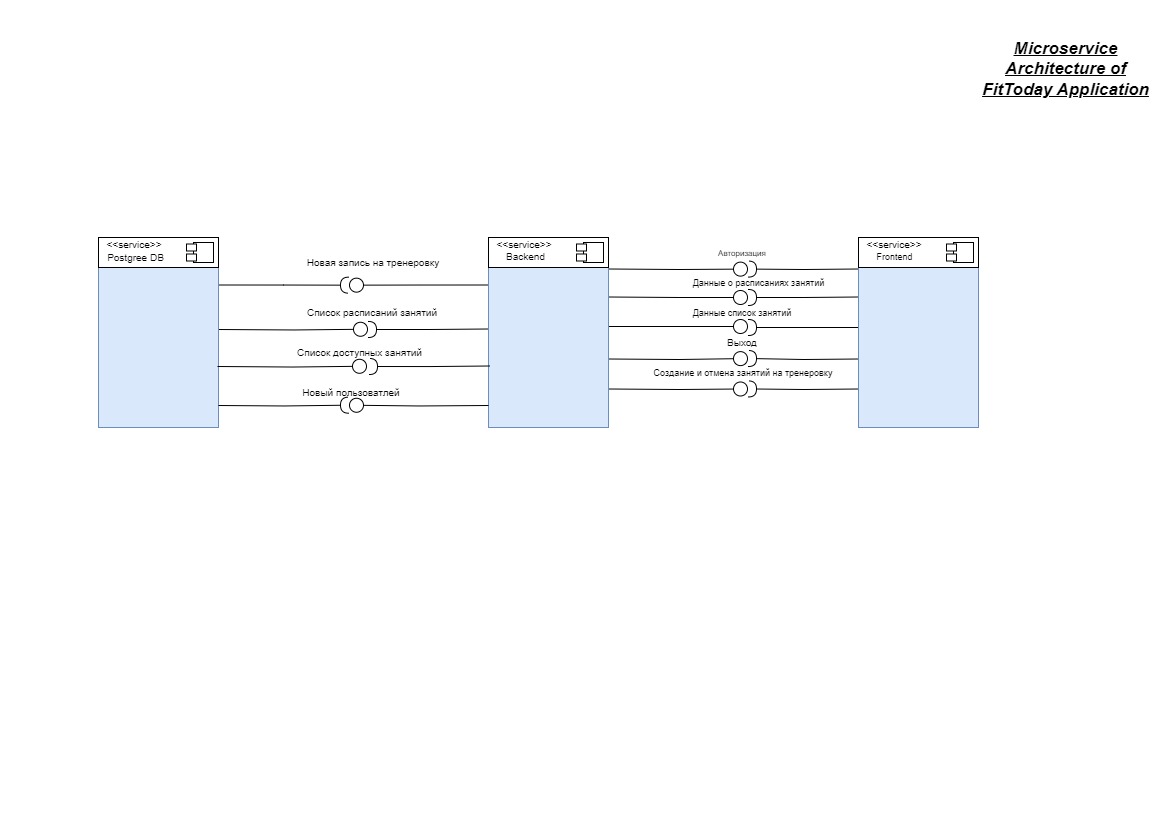


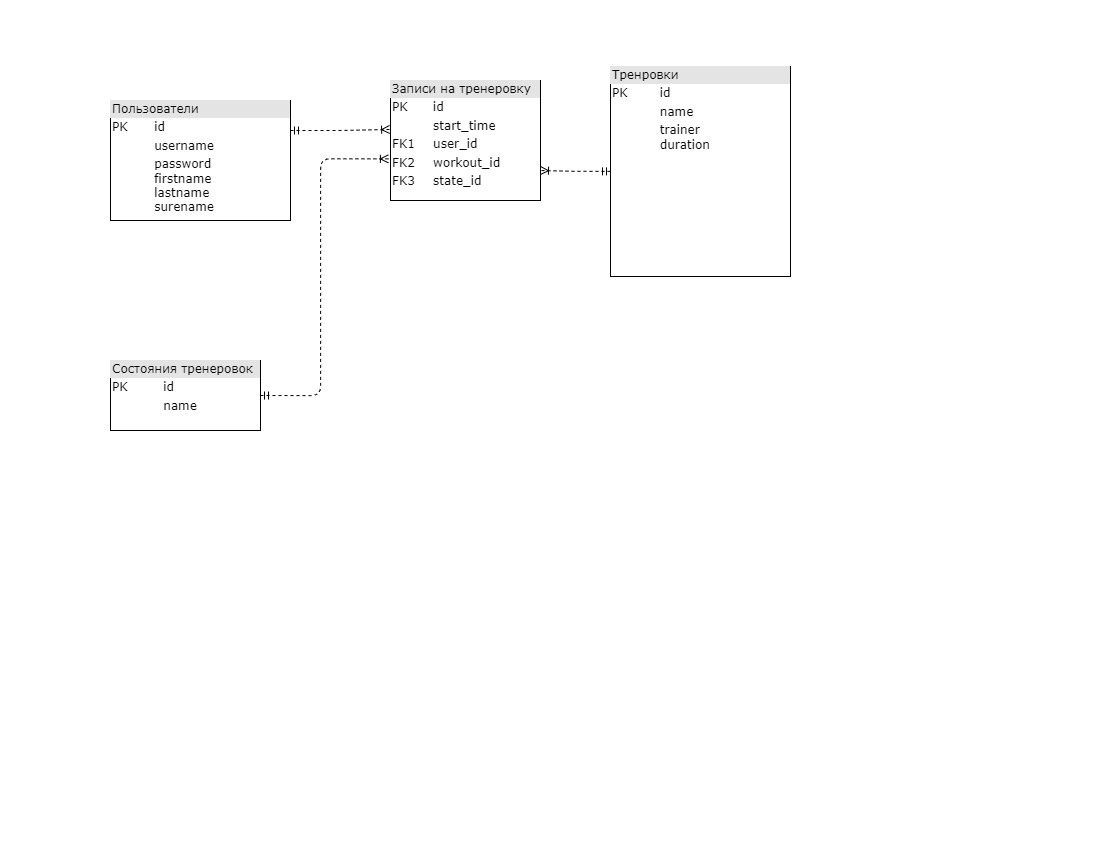
Рисунок 2 - диаграмма компонентов для приложения тренажерного зала

## **Логическая модель данных для приложения**

Логическая модель данных - это абстрактное представление структуры данных в информационной системе, которое не зависит от конкретной системы управления базами данных (СУБД) или физической реализации. Это важный этап в процессе проектирования баз данных, который следует за концептуальным моделированием и предшествует физическому моделированию.

Логическая модель данных является критически важным этапом в процессе проектирования базы данных, обеспечивая мост между бизнес-требованиями и техническим реализацией системы хранения данных.

На рисунке 3 представлена логическая структура данных БД для приложения тренажерного зала.



Рисунке 3 - Логическая структура данных БД для приложения тренажерного зала.

# **Разработка технического задания**

Техническое задание - документ в котором сформулированы основные цели разработки, требования к по, определены сроки и этапы разработки, регламентирован процесс приемо-сдаточных испытаний.

В разработке технического задания участвуют как представители заказчика, так и представители исполнителя. В основе этого документа лежат исходные требования заказчика, анализ передовых достижений техники, результаты выполнения научно-исследовательских работ, предпроектных исследований, научного прогнозирования и т. п.

Разработка технического задания выполняется в следующей последовательности:

1) устанавливают набор выполняемых функций, а также перечень и характеристики исходных данных;

2) определяют перечень результатов, их характеристики и способы представления;

3) уточняют среду функционирования программного обеспечения: конкретную комплектацию и параметры технических средств, версию используемой опера­ционной системы и, возможно, версии и параметры другого установленного программного обеспечения, с которым предстоит взаимодействовать будущему программному продукту;

4) в случаях, когда разрабатываемое программное обеспечение собирает и хранит некоторую информацию или включается в управление каким-либо техническим процессом, необходимо также четко регламентировать действия программы в случае сбоев оборудования и энергоснабжения.

**Наименование системы -**  "FitToday" приложение для пользователе тренажерного зала

**Основание для разработки,** потребность рынка в современном удобном сервисе тренажерного зала.

**Назначение приложения,** автоматизация процесса обработки записей на тренировки от пользователей.

**Цели:**

* упростить процесс обработки записи на различные виды тренировок;
* упростить просмотр тренировок по их состояниям ведение автопарка;
* упростить процесс отмена активных тренировок.

**Функциональные требования:**

* регистрация и авторизация пользователей;
* отображения всех расписание у пользователя;
* отображения всех возможных записей на тренировки;
* фильтрация расписания по состояниям;
* отмена активных расписаний на тренировки.

**Нефункциональные требования:**

* поддержка iOS и Android платформ
* веб-версия приложения
* высокая производительность (время отклика не более 2 секунд)
* масштабируемость (поддержка до 100 000 одновременных пользователей)
* безопасность данных пользователей и транзакций

# **Разработка пользовательского интерфейса**

Пользовательский интерфейс – это совокупность правил взаимодействия пользователя с программой или вычислительной системой и средств, реализующих это взаимодействие.

Элемент взаимодействия — это элемент пользовательского интерфейса, с помощью которого пользователь непосредственно взаимодействует с программой или вычислительной системой.

Пассивный элемент взаимодействия — это элемент пользовательского интерфейса, через который пользователь не имеет прямого доступа к системным или программным ресурсам, т.е. не может управлять или изменять эти ресурсы напрямую и непосредственно.

Активный элемент взаимодействия - это элемент пользовательского интерфейса, через который пользователь имеет прямой доступ к системным и программным ресурсам с возможностью непосредственного управления и изменения их.

## **5.1 Вход и регистрация пользователя**

На странице на главной странице пользователя представлено 2 кнопки, кнопка «Вход» открывает диалоговое окно, где пользователь производит авторизацию, путем ввода данных в поля ввода, такие как Логин и Пароль, и нажатия на кнопку «На тренировку».

Кнопка «Регистрация» пользователь производить регистрацию вводит всю необходимую информацию.

После успешной авторизации или регистрации пользователь переходит на домашнюю страницу приложения, которая описана в пункте 5.4.

На рисунке 4 показана диалоговое окно входа пользователя.

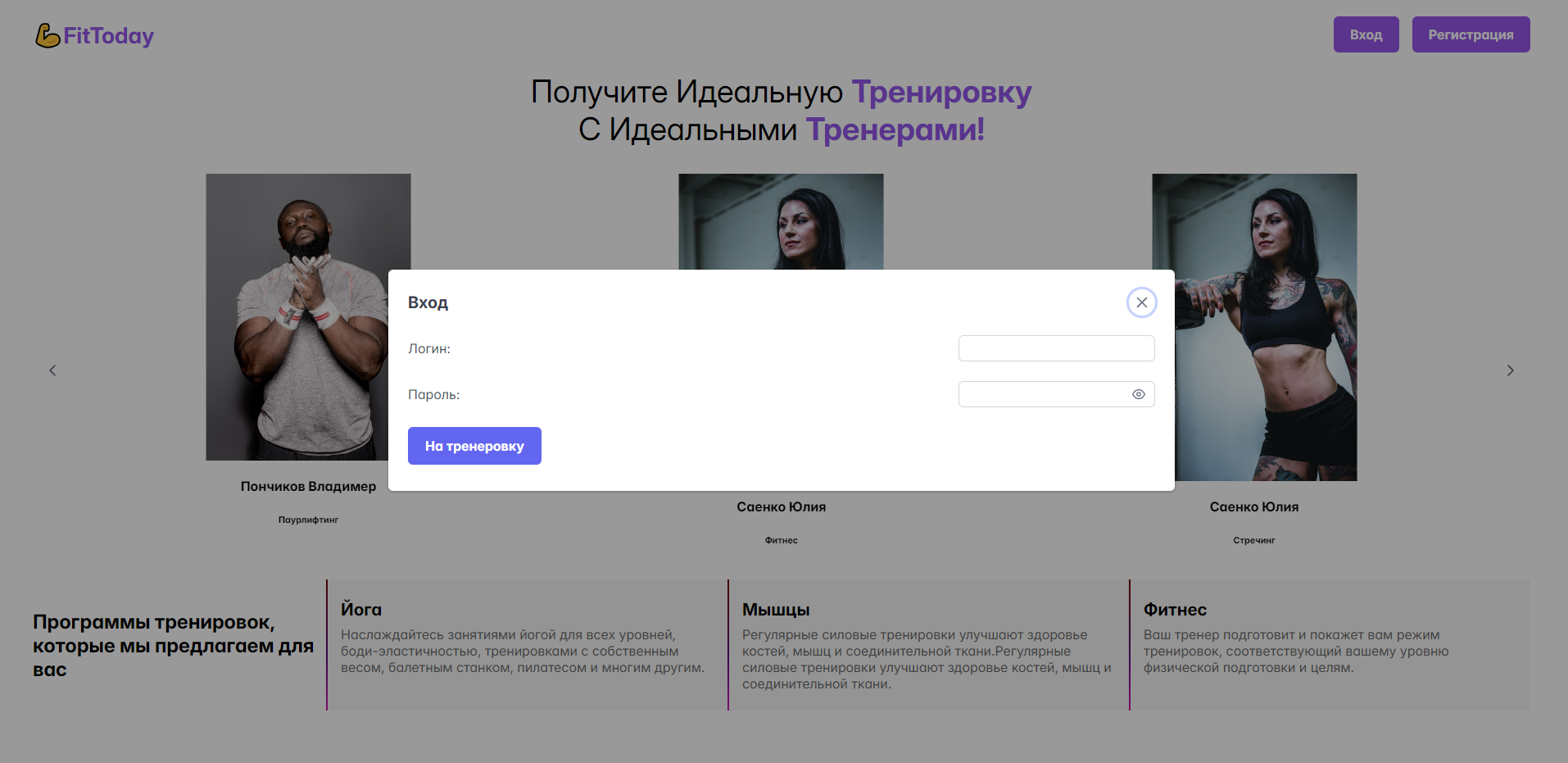


Рисунок 4 – Диалоговое окно входа пользователя.

На рисунке 2 показана страница регистрации нового пользователя.

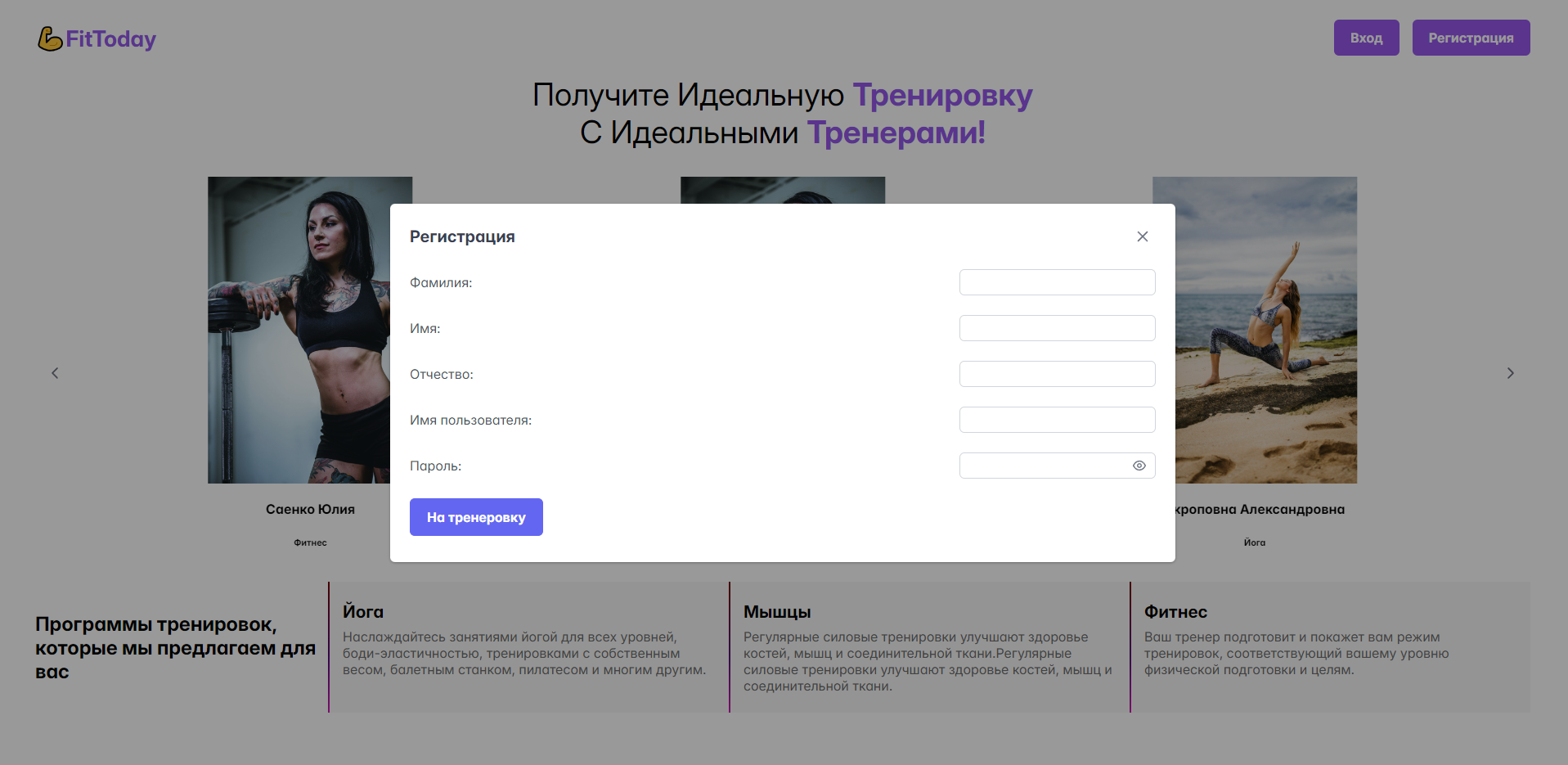


Рисунок 5 - Диалоговое окно регистрации нового пользователя.

## **5.2 Начальная страница приложения**

На начальной странице приложения пользователю представлено на рисунке 6:

* навигационная область приложения - пользователю представляется логотип приложения, список выбора состояние фильтрации тренировок, кнопка выхода, которая производит выход из профиля пользователя;
* область приветствия пользователя – приложение получает информацию об пользователе, и приветствует его по его имени, и показывает текущею дату и время в реально времени, изменение времени происходит каждую секунду;
* область просмотра расписания тренировок – отображение всех, или по состояниям тренировок (все, активные, отменённые), где красный цвет означает, что тренировка отмена, а фиолетовый цвет означает, что тренировка активна;
* область записи на занятия - пользователю представляется список тренировок в тренажёрном зале, где он может произвести запись на тренировку. Запись на тренировку происходит путем нажатия на кнопку «Запись», после чего создается запись в базе данных, и сотруднику отображается его запись под состоянием активно.

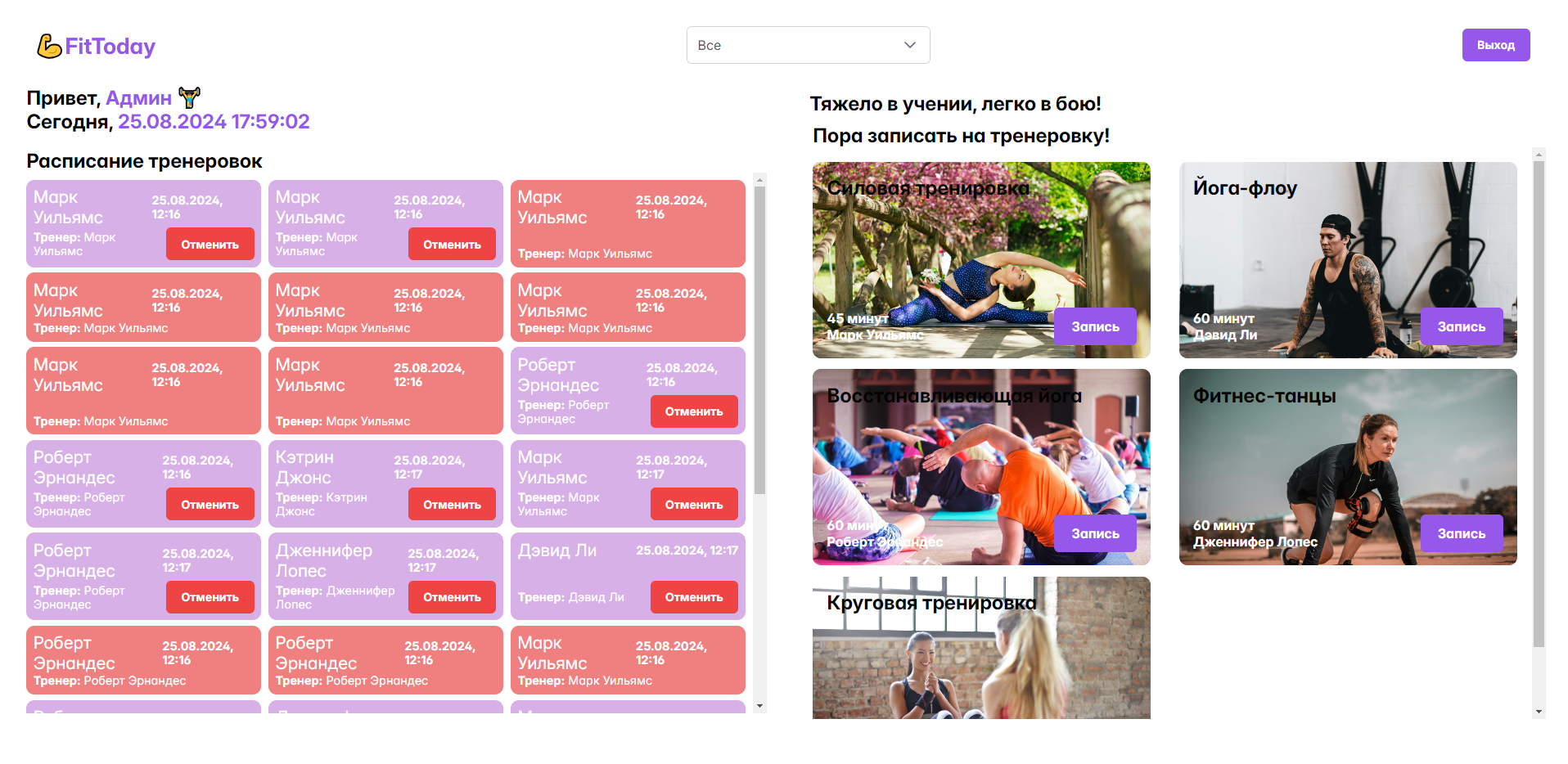


Рисунок 6 – Начальная страница приложения

# **Проектирование классов**

Диаграмма классов определяет типы классов системы и различного рода статические связи, которые существуют между ними. На диаграммах классов изображаются также атрибуты классов, операции классов и ограничения, которые накладываются на связи между классами.

Диаграмма классов UML - это граф, узлами которого являются элементы статической структуры проекта (классы, интерфейсы), а дугами - отношения между узлами (ассоциации, наследование, зависимости).

Стереотипы классов – это механизм, позволяющий разделять классы на категории.

В языке UML определены три основных стереотипа классов:

* Boundary (граница);
* Entity (сущность);
* Control (управление).

На диаграмме классов изображаются следующие элементы:

* Пакет (package) - набор элементов модели, логически связанных между собой;
* Класс (class) - описание общих свойств группы сходных объектов;
* Интерфейс (interface) - абстрактный класс, задающий набор операций, которые объект произвольного класса, связанного с данным интерфейсом, предоставляет другим объектам.

Атрибут – это элемент информации, связанный с классом. Атрибуты хранят инкапсулированные данные класса.

Так как атрибуты содержатся внутри класса, они скрыты от других классов. В связи с этим может понадобиться указать, какие классы имеют право читать и изменять атрибуты. Это свойство называется видимостью атрибута (attribute visibility).

У атрибута можно определить четыре возможных значения этого параметра:

* Public (общий, открытый). Это значение видимости предполагает, что атрибут будет виден всеми остальными классами. Любой класс может просмотреть или изменить значение атрибута. В соответствии с нотацией UML общему атрибуту предшествует знак « + ».
* Private (закрытый, секретный). Соответствующий атрибут не виден никаким другим классом. Закрытый атрибут обозначается знаком « – » в соответствии с нотацией UML.
* Protected (защищенный). Такой атрибут доступен только самому классу и его потомкам. Нотация UML для защищенного атрибута – это знак « # ».
* Package or Implementation (пакетный). Предполагает, что данный атрибут является общим, но только в пределах его пакета. Этот тип видимости не обозначается никаким специальным значком.

Класс - это группа сущностей (объектов), обладающих сходными свойствами, а именно, данными и поведением. Отдельный представитель некоторого класса называется объектом класса или просто объектом.

Под поведением объекта в UML понимаются любые правила взаимодействия объекта с внешним миром и с данными самого объекта.

# **Результаты тестирования программы**

Тестирование — это процесс анализа ПО, направленный на выявление отличий между его реально существующими и требуемыми свойствами и на оценку свойств ПО.

Никакое тестирование не может доказать отсутствие ошибок в ПО.

На начальных этапах контроль осуществляют в основном вручную или с использованием CASE-средств, на последнем этапе – он принимает форму тестирования.

Процесс разработки ПО предполагает три стадии тестирования:

* автономное тестирование - это процесс тестирования отдельных модулей, компонентов или единиц программного обеспечения в изолированной среде. Цель заключается в проверке функциональности и корректности работы каждого отдельного элемента системы независимо от остальных частей.
* комплексное тестирование (интеграционное) - это процесс объединения отдельных модулей или компонентов программного обеспечения и тестирования их взаимодействия друг с другом. Цель заключается в проверке корректной совместной работы всех элементов системы.
* системное (оценочное) тестирование - это процесс тестирования всей интегрированной системы в целом, чтобы проверить соответствие заданным требованиям и критериям качества. Оно проводится после успешного завершения комплексного тестирования.

Различают два подхода к формированию тестов:

* структурный
* функциональный.

Основные принципы:

* избегать тестирования программы самим автором,
* предполагаемые результаты должны быть известны до тестирования,
* необходимо изучать результаты каждого теста,
* необходимо проверять действие программы на неверных данных.

Вероятность наличия необнаруженных ошибок пропорциональна количеству уже найденных ошибок в этой программе.

Jest - это популярный фреймворк для тестирования JavaScript, созданный Facebook. Вот краткое описание Jest в примерно 100 словах:

Jest предоставляет простой и эффективный способ написания и запуска тестов для JavaScript-кода. Он поддерживает автоматическое мокирование зависимостей, что упрощает изоляцию тестируемого кода. Jest включает встроенную поддержку покрытия кода, что помогает разработчикам отслеживать, какие части кода протестированы.

Фреймворк использует понятный синтаксис для описания тестов, включая функции describe() для группировки и it() для отдельных тестовых случаев. Jest также предоставляет богатый набор матчеров для проверки различных условий.

Одно из главных преимуществ Jest - его скорость. Он может запускать тесты параллельно и имеет умную систему запуска, которая приоритизирует провалившиеся тесты.

Jest хорошо интегрируется с популярными инструментами разработки и поддерживает различные типы тестирования, включая модульное, интеграционное и end-to-end тестирование.

На рисунке 15 показана результат успешного тестирование приложения

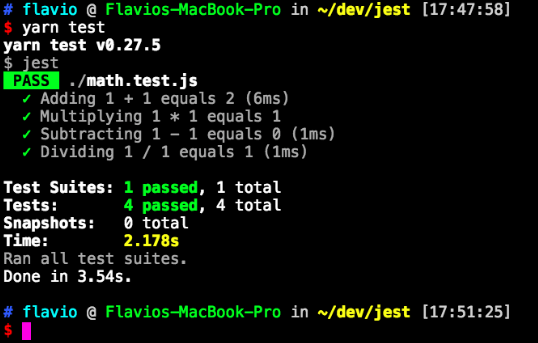


Рисунок 15 – Тестирование приложения

# **Заключение**

В ходе курсовой работы было выполнено главная цель спроектирована и разработана эффективное, масштабируемое и удобное для пользователя приложение для тренажерного зала, применяя современные методологии и инструменты программной инженерии.

Так же были выполнены главные задачи проекта:

* разработка приложения
* разработана диаграмма вариантов использования приложения;
* разработана диаграмма классов;
* разработана логическая модель данных;
* разработана диаграмма компонентов;

# **Список использованных источников**

1. Введение в программную инженерию: Учебник / В.А. Антипов, А.А. Бубнов, А.Н. Пылькин, В.К. Столчнев. — М.: КУРС: ИНФРА-М, 2019. — 336 с. - Режим доступа: http://znanium.com/bookread2.php?book=1035160
2. Гвоздева В.А. Введение в специальность программиста [Электронный ресурс]: учебник / В.А. Гвоздева. - 2-e изд., испр. и доп. - М.: ИД ФОРУМ: НИЦ ИНФРА-М, 2017. - 208 с. http://znanium.com/bookread2.php?book=552523
3. Объектно-ориентированное программирование с примерами на С# : учеб. пособие / П.Б. Хорев. М. : ФОРУМ : ИНФРА-М, 2019. –200 с. - Режим доступа: http://znanium.com/catalog/product/1018034
4. Объектно-ориентированное программирование. Хорошая книга для Хороших Людей / Комлев Н.Ю. – М.:СОЛОН-Пр., 2015. – 298 с.: Режим доступа: http://znanium.com/catalog/product/884394
5. Использование диаграммы вариантов использования UML при проектировании программного обеспечения. URL: https://habr.com/ru/articles/566218/. Дата посещения: 01.08.2024
6. Диаграммы классов. URL: <https://studfile.net/preview/2806643/page:4/>. Дата посещения: 01.08.2024
7. Пользовательский интерфейс и его виды. URL: <https://studfile.net/preview/2484532/page:15/>. Дата посещения: 01.07.2024
8. Фундаментальная теория тестирования. <URL:https://habr.com/ru/articles/549054/>. Дата посещения: 21.08.2024

# **Приложение А Тест на анти плагиат**

Данная работа была проверена на сайте www.antiplagiat.ru, о результат проверки предоставлен ниже на рисунке 7.

Рисунок 7 – Результат проверки работы на сайте www.antiplagiat.ru