

Проверка на антиплагиат (09.02.07)

Государственная итоговая аттестация (ГИА)

Общий балл: 17 % Средний риск

Расул Галеев

UUID отправки: 0c1ac9b0-e800-f584-9e1a-07f07e25f9c2

Общее количество отчетов 1	Максимальное совпадение 17 % DIPLOM_v2.docx	Средний процент совпадений 17 %	Дата отправки: 04.06.25 23:56 GMT+3	Среднее число слов 9 632 Максимальное: DIPLOM_v2.docx
-------------------------------	---	------------------------------------	---	---

Вложение 1 17 % Число слов: 9 632
Источник: DIPLOM_v2.docx

Глобальная база данных (15) 14 %

<div>24</div> Работа учащегося	<div>15</div> Работа учащегося	<div>19</div> Работа учащегося
<div>18</div> Работа учащегося	<div>14</div> Работа учащегося	<div>13</div> Работа учащегося
<div>16</div> Работа учащегося	<div>20</div> Работа учащегося	<div>17</div> Работа учащегося
<div>23</div> Работа учащегося	<div>22</div> Работа учащегося	<div>10</div> Работа учащегося
<div>25</div> Работа учащегося	<div>21</div> Работа учащегося	<div>11</div> Работа учащегося

Institutional Database (4) 2 %

<div>9</div> Документ пользователя	<div>8</div> Документ пользователя	<div>7</div> Документ пользователя
<div>2</div> Работа учащегося		

Интернет (6) 1 %

<div>1</div> pup-zemli	<div>12</div> referat911	<div>5</div> referat911
<div>4</div> turboreferat	<div>6</div> referat911	<div>3</div> lib-i

Основные источники (3)

<div>24</div> Работа учащегося 3 %	<div>15</div> Работа учащегося 3 %	<div>19</div> Работа учащегося 2 %
------------------------------------	------------------------------------	------------------------------------

1

 федеральное государственное бюджетное образовательное учреждение высшего образования «Казанский национальный исследовательский технический университет им. А.Н.

2

Туполева-КАИ» (КНИТУ-КАИ)

ОТДЕЛЕНИЕ СРЕДНЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

ДИПЛОМНЫЙ ПРОЕКТ

Тема: «Веб-сайт для фитнес-центра с использованием ИИ для консультации и подбора персональных тренировок, а также составлением плана питания»

Руководитель

Преподаватель

(должность) (подпись)

(И.О. Фамилия)

Обучающийся

4433

3 Галеев Р.А.

(группа) (подпись)

(И.О. Фамилия)

Специальность 09.02.07 «Информационные системы и программирование»

(шифр и наименование специальности)

Казань 2025 г. ЛИСТ ЗАДАНИЯ

ЛИСТ ЗАДАНИЯ

АННОТАЦИЯ

Дипломный проект на тему «Веб-сайт для фитнес-центра с использованием ИИ для консультации и подбора персональных тренировок, а также составлением плана питания» посвящён созданию информационной системы, направленной на автоматизацию предоставления фитнес-услуг и поддержку пользователей в достижении их целей в области здоровья и физической активности. Проект состоит из * страниц, содержит * рисунка и включает в себя 20 источников использованной литературы. Основной задачей проекта является разработка веб-приложения, которое с использованием технологий искусственного интеллекта формирует индивидуальные тренировочные программы и планы питания, а также предоставляет консультации по физической активности. Приложение учитывает персональные данные пользователя, уровень физической подготовки, цели тренировок и предпочтения в питании. Веб-сервис разработан с использованием современных технологий: Django в качестве backend-фреймворка, React — для frontend-части, PostgreSQL — в качестве системы управления базами данных. Для разработки и отладки проекта использовалась среда PyCharm. Коммуникация между клиентской и серверной частью осуществляется через REST API. Проект ориентирован на фитнес-центры и индивидуальных пользователей, стремящихся получать персонализированные рекомендации и эффективно управлять своим здоровьем и физической формой. Использование искусственного интеллекта позволяет повысить качество предоставляемых рекомендаций, вовлечённость пользователей, а также упростить взаимодействие с сервисом благодаря интеллектуальной поддержке.

4 СОДЕРЖАНИЕ ВВЕДЕНИЕ 6

ГЛАВА 1 ПОСТАНОВКА ЗАДАЧИ 8

1.1 Цели и задачи 9

1.2 Требования к функциональности системы 10

1.3 Список требований к программному продукту 12

1.4 Требования к интерфейсу приложения 13

5 ГЛАВА 2 ПРОЕКТИРОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ 15

6 2.1 Выбор средств реализации 16

2.1.1 Языки программирования 16

7 2.1.2 Среда разработки 17

2.1.3 Система управления базами данных 17

2.1.4 Фреймворки и библиотеки 18

2.2 Проектирование диаграммы вариантов использования 18

5 2.3 Проектирование диаграммы последовательности 21

2.4 Проектирование диаграммы деятельности 23

2.5 Проектирование диаграммы состояний 27

2.6 Проектирование диаграммы сущность-связь 30

8 2.7 Проектирование базы данных 34

ГЛАВА 3 ПОЛЬЗОВАТЕЛЬСКАЯ ДОКУМЕНТАЦИЯ 37

3.1 Руководство программиста 37

3.2 Руководство пользователя 39

3.2.1 Руководство роли «Пользователь» 40

3.2.2 Руководство роли «Администратор» 44

ГЛАВА 4 ТЕСТИРОВАНИЕ 47

ЗАКЛЮЧЕНИЕ 49

9 СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 51

ПРИЛОЖЕНИЕ А СЛОВАРЬ ДАННЫХ 53

ПРИЛОЖЕНИЕ Б ЛИСТИНГ ПРОГРАММЫ 55

ПРИЛОЖЕНИЕ В ПРЕЗЕНТАЦИЯ 55

ВВЕДЕНИЕ

В последние годы внедрение технологий в фитнес-индустрию стало важным шагом на пути к персонализации тренировок и улучшению качества услуг. В современном мире, где здоровый образ жизни и индивидуальный подход к физической активности приобретают всё большее значение, разработка инновационных решений, способных учитывать уникальные особенности пользователей, становится необходимостью. Однако многие существующие приложения для фитнеса ограничиваются стандартными рекомендациями, не учитывающими личные цели, физическую подготовку и образ жизни пользователей, что снижает их эффективность и привлекательность. Использование искусственного интеллекта (ИИ) в разработке персональных фитнес-программ играет ключевую роль в решении этих задач. Технологии ИИ позволяют учитывать широкий спектр индивидуальных данных, включая уровень физической подготовки, состояние здоровья, предпочтения в тренировках, доступное время и цели пользователя. Анализируя эти параметры, ИИ способен создавать уникальные планы тренировок и питания, адаптируемые в режиме реального времени. Цель дипломного проекта заключается в разработке веб-приложения, которое с использованием искусственного интеллекта предоставляет персонализированные рекомендации для занятий фитнесом. 10 Для достижения поставленной цели необходимо решить

следующие задачи: 1. Проанализировать существующие решения в области фитнеса и определить их недостатки. 2. Выбрать архитектурный подход и стек технологий для реализации проекта. 3. Разработать структуру базы данных для хранения информации о пользователях, тренировках и питании. 4. Реализовать backend-систему на основе Django. 5. Создать клиентскую часть приложения на React с удобным пользовательским интерфейсом. 6. Разработать и интегрировать модуль искусственного интеллекта для генерации персонализированных рекомендаций. 7. Протестировать систему и оценить корректность её работы. 8. В процессе реализации проекта будут применяться современные подходы к проектированию и разработке веб-приложений, а также методы интеллектуальной обработки данных. Особое внимание будет уделено обеспечению гибкости, масштабируемости и точности генерируемых рекомендаций. Таким образом, создание веб-приложения с использованием искусственного интеллекта, выполняющего функции персонального фитнес-тренера, является актуальной задачей. Оно не только упростит процесс тренировки и повысит её эффективность, но и сделает фитнес более доступным и персонализированным, что положительно скажется на уровне физической активности и поддержании здорового образа жизни.

ГЛАВА 1 ПОСТАНОВКА ЗАДАЧИ

Разработка веб-приложения для персонализированного подбора тренировок и планов питания на основе технологий искусственного интеллекта требует детального предварительного анализа, чёткого понимания целей и формулировки требований, на которых будет основываться дальнейшая реализация проекта. Настоящая глава посвящена постановке задачи и определению концепции программного продукта, который направлен на улучшение взаимодействия пользователей с фитнес-сферой посредством цифровых технологий. В рамках данной главы проводится анализ ключевых аспектов проекта: обосновывается актуальность решения, формулируются основные цели и задачи разработки, а также определяются требования к функциональности и техническим характеристикам системы. Такой подход позволяет сформировать обоснованную и структурированную модель веб-сервиса, обеспечивающего индивидуальный подход к построению тренировочного процесса. Особое внимание уделяется функциональному назначению системы. Предполагается, что разрабатываемый продукт будет не просто предоставлять шаблонные тренировочные планы, а адаптировать рекомендации на основе индивидуальных параметров пользователя, включая физические характеристики, цели, уровень подготовки и предпочтения. Это достигается за счёт применения алгоритмов машинного обучения и анализа данных, что существенно повышает эффективность и релевантность предоставляемой информации. Кроме того, в данной главе рассматриваются вопросы

безопасности, производительности, надёжности хранения и передачи пользовательских данных, а также удобства взаимодействия с интерфейсом. Немаловажным аспектом является адаптивность пользовательского интерфейса, обеспечивающая полноценную работу на устройствах с различными характеристиками, включая смартфоны и планшеты. Предложенное решение направлено на расширение возможностей пользователей в достижении фитнес-целей и оптимизацию тренировочного процесса. Это делает необходимость разработки подобного продукта особенно актуальной в условиях современной тенденции к цифровизации повседневной жизни и повышения интереса к здоровому образу жизни. ⑪ 1.1

Цели и задачи

Целью настоящего проекта является разработка веб-приложения для фитнес-центра, которое с использованием технологий искусственного интеллекта (ИИ) позволит формировать персонализированные тренировочные планы и рекомендации по питанию. Приложение должно учитывать индивидуальные характеристики пользователей и автоматически подбирать наиболее эффективные решения, способствующие достижению поставленных целей: снижение веса, набор мышечной массы, улучшение выносливости или поддержание общего тонуса организма. Построение подобного программного продукта обусловлено необходимостью повышения эффективности тренировочного процесса за счёт автоматизации рутинных задач и предоставления пользователям интеллектуальной поддержки. В современных условиях высокая конкуренция на рынке фитнес-услуг требует от организаций внедрения цифровых решений, повышающих вовлечённость клиентов и обеспечивающих высокий уровень персонализации. ⑫ Для реализации поставленной цели необходимо решить следующие задачи:

1. Разработать архитектуру веб-приложения с использованием современных технологий (React, Django, PostgreSQL), обеспечивающих масштабируемость и удобство в сопровождении проекта. 2. Реализовать модуль регистрации и авторизации пользователей с защитой персональных данных и возможностью сохранения истории взаимодействий. 3. Обеспечить сбор и хранение индивидуальных параметров пользователей (возраст, рост, вес, цели тренировок, уровень подготовки и др.) с последующим использованием этих данных для генерации рекомендаций. 4. Разработать интеллектуальный модуль, основанный на алгоритмах ИИ и машинного обучения, который будет анализировать пользовательские данные и формировать персонализированные тренировочные программы и планы питания. 5. Создать функциональность для отслеживания прогресса: фиксации выполненных тренировок, визуализации изменений и генерации отчётов. 6. Реализовать механизм адаптивной новостной ленты с подборкой статей, видеоконтента и советов, соответствующих интересам пользователя. 7. Реализовать современный адаптивный пользовательский интерфейс, соответствующий требованиям доступности и удобства использования. 8. Провести тестирование всех функциональных компонентов приложения, включая проверку надёжности, безопасности и производительности. Реализация указанных задач позволит создать эффективный инструмент, обеспечивающий индивидуальный подход к организации тренировочного процесса, что будет способствовать повышению мотивации и достижению стабильных результатов пользователями. ⑨ 1.2 Требования к функциональности системы

Разрабатываемое приложение должно быть многофункциональным и обеспечивать высокое качество взаимодействия с пользователем, предоставляя широкий спектр ключевых возможностей. Одной из главных функций является предоставление персонализированных тренировочных программ, которые адаптируются на основе уникальных данных пользователя, таких как физическая подготовка, цели тренировок, предпочтительные виды активности и график. Программа будет учитывать такие параметры, как уровень нагрузки, возраст, пол и состояние здоровья, чтобы предложить наиболее эффективное решение для достижения поставленных целей. Для повышения вовлеченности и мотивации пользователей необходимо реализовать функцию отслеживания прогресса. Она будет фиксировать выполненные упражнения, достигнутые результаты и динамику изменений, предоставляя пользователю возможность анализировать свои достижения через графики и отчеты. Это позволит не только отслеживать улучшения, но и мотивировать продолжать занятия. Кроме того, приложение должно поддерживать интеграцию с популярными социальными платформами, такими как Instagram, Facebook, Twitter, а также с мессенджерами. Эта возможность позволит пользователям делиться своими достижениями, результатами тренировок и индивидуальными планами, создавая сообщества для поддержки и обмена опытом. Еще одной важной функцией является новостная лента, где будут публиковаться актуальные статьи, тренды в области фитнеса, рекомендации экспертов и обучающие видеоролики. Лента будет персонализированной, чтобы отображать только ту информацию, которая соответствует интересам и целям пользователя, делая приложение полезным не только для тренировки, но и для обучения. Также приложение должно предоставлять возможность выбора или создания индивидуального тренировочного плана. Пользователи смогут сохранять свои планы, редактировать их или возвращаться к уже выполненным. Для обеспечения максимальной персонализации ИИ будет анализировать физические параметры пользователя (рост, вес, уровень активности и текущую форму), предлагая наиболее эффективные упражнения и оптимальные нагрузки, что сделает процесс тренировки более адаптированным и эффективным. Система будет разделена на две роли: «Клиент» и «Администратор». Роль клиента включает возможность персонализированного выбора и отслеживания тренировок и питания, в то время как администратор будет заниматься управлением контентом и пользователями, обеспечивая функциональность платформы и корректную работу всех сервисов. Реализация всех этих функций обеспечит пользователю не только удобство и мотивацию, но и индивидуальный подход, что делает приложение незаменимым помощником в достижении фитнес-целей. 1.3 Список требований к программному продукту

Приложение должно обеспечивать стабильную и надежную работу, исключая потерю пользовательских данных, даже при нестабильном интернет-соединении. Ключевым требованием является надежное хранение данных, включая историю тренировок, параметры прогресса и настройки, которые должны сохраняться в базе данных с возможностью синхронизации между устройствами. Для этого будет использована база данных PostgreSQL, что гарантирует высокую производительность и надежность при обработке больших объемов информации. Для обеспечения конфиденциальности всех пользовательских данных необходимо их передача исключительно через защищенные каналы связи (например, HTTPS), а также их хранение с использованием современных методов шифрования, чтобы минимизировать риски несанкционированного доступа. Система должна обеспечивать обработку данных пользователя в многозадачной среде с использованием асинхронных операций. Это позволит поддерживать высокую производительность и минимизировать задержки при выполнении таких операций, как расчет параметров тренировок или синхронизация данных с сервером. Особое внимание следует уделить функционалу для ведения истории тренировок. Пользователи должны иметь возможность сохранять, просматривать, редактировать и удалять свои тренировочные программы. Для повышения удобства взаимодействия с историей данных, необходимо реализовать функции фильтрации и поиска по ключевым параметрам (например, по дате или типу тренировки).

Кроме того, приложение должно использовать современные стандарты безопасности, включая поддержку протоколов TLS для безопасной передачи данных и защиты информации от несанкционированного доступа. 1.4 Требования к интерфейсу приложения

Интерфейс приложения должен быть разработан с учетом современных стандартов пользовательского опыта и удобства. Вдохновленный принципами Material Design, интерфейс будет интуитивно понятным и визуально привлекательным. Это обеспечит легкость в использовании для всех категорий пользователей, независимо от их уровня технической подготовки. Главным приоритетом является удобство навигации, чтобы каждый пользователь мог без труда найти нужный функционал, будь то подбор тренировок, отслеживание прогресса или взаимодействие с тренером. Для обеспечения комфортного взаимодействия с системой интерфейс должен быть четким и логичным, с продуманной иерархией элементов. Цветовая схема будет подбираться с учетом психологии восприятия, где акценты на важнейших функциях будут выполнены контрастными цветами, что позволит быстро ориентироваться в приложении. Все элементы интерфейса, такие как кнопки, формы и меню, будут иметь четкие и легко различимые границы, а текст — оптимальный размер для чтения. Для обеспечения высококачественного пользовательского опыта важным аспектом будет адаптивность интерфейса. Он будет автоматически подстраиваться под размеры экрана устройства, будь то мобильный телефон, планшет или компьютер, обеспечивая таким образом удобный доступ к функционалу без потери качества взаимодействия. Это также подразумевает поддержку как вертикальной, так и горизонтальной ориентации экрана, что является важным для пользователей с различными предпочтениями и устройствами. Помимо адаптивности, особое внимание следует уделить быстродействию интерфейса. Каждый элемент будет оптимизирован для минимизации времени загрузки страниц и обработки запросов. Приложение должно обеспечивать мгновенную обратную связь с пользователем, например, при регистрации или создании тренировочного плана, что обеспечит высокий уровень удовлетворенности. Интерфейс должен быть не только удобным, но и современным, что позволит создать приятное визуальное восприятие и повысить доверие пользователей. Применение последних достижений в области веб-разработки обеспечит качественное отображение контента и эффектное анимационное оформление, не перегружая систему.

9 ГЛАВА 2 ПРОЕКТИРОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ

В этой главе описан процесс проектирования информационной системы для веб-приложения с искусственным интеллектом, предназначенного для создания персонализированных тренировочных программ и рекомендаций по питанию. Приводится обоснование выбора используемых технологий и инструментов, а также рассматриваются ключевые этапы проектирования, включая разработку архитектуры системы и создание диаграмм для визуализации функциональности и структуры приложения. Проектирование информационной системы имеет решающее значение, поскольку оно закладывает основу для надежности, эффективности и масштабируемости всего приложения. Важным моментом является выбор архитектурных решений, которые обеспечат гибкость системы и ее способность адаптироваться к меняющимся требованиям. Также особое внимание уделяется обеспечению безопасности данных, что критически важно для создания доверия со стороны пользователей и защиты их личной информации. В разделе рассматриваются ключевые аспекты выбора инструментов и технологий, которые обеспечат высокую производительность, безопасность и удобство для пользователей. Это включает выбор языков программирования, фреймворков, систем управления базами данных, а также библиотек и инструментов для реализации функционала приложения. Главной задачей является создание системы, способной эффективно работать с большими объемами данных, обеспечивать стабильную работу при повышенных нагрузках и предоставлять персонализированные рекомендации на основе физических данных и целей пользователей. Учитываются и возможности дальнейшего расширения функционала. Глава охватывает как технологические, так и проектировочные аспекты, которые помогут создать удобный, надежный и эффективный инструмент для формирования и отслеживания персонализированных тренировок и рекомендаций по питанию. 2.1 Выбор средств реализации

В данном разделе рассматриваются ключевые технологии и инструменты, которые будут использованы для разработки веб-приложения с искусственным интеллектом для создания персонализированных тренировочных программ и рекомендаций по питанию. Для реализации функционала, обеспечения надежности и безопасности системы было принято решение использовать ряд проверенных технологий, которые обеспечат высокую производительность и удобство разработки. 7 2.1.1 Языки программирования

Для серверной части приложения был выбран Python. Этот язык является одним из наиболее распространенных и эффективных в области разработки систем с элементами искусственного интеллекта и машинного обучения. Python широко используется для создания нейросетевых моделей благодаря множеству специализированных библиотек, таких как TensorFlow, PyTorch, Keras, и Scikit-learn. Эти инструменты позволяют эффективно строить и обучать нейросети, что является основой для формирования персонализированных рекомендаций по тренировкам и питанию. Кроме того, Python хорошо поддерживает разработку веб-приложений благодаря фреймворку Django. Django предлагает высокоуровневые средства для организации серверной логики, работы с базой данных и управления безопасностью. Важной особенностью Django является встроенная система аутентификации и авторизации, что позволит легко внедрить защиту данных и контроль доступа в приложение. Для клиентской части веб-приложения был выбран JavaScript, в частности, его библиотека React. JavaScript позволяет создавать динамичные, отзывчивые и интерактивные интерфейсы, что особенно важно для приложения, которое должно обеспечивать удобный и интуитивно понятный пользовательский опыт. React использует компонентный подход, который позволяет создавать переиспользуемые элементы интерфейса, что ускоряет разработку и упрощает поддержку кода. React также позволяет строить интерфейсы, которые эффективно обновляются только при изменении данных, что значительно улучшает производительность приложения. 7 2.1.2 Среда разработки

Для разработки серверной части приложения предпочтительнее использовать PyCharm — мощную среду разработки для Python. PyCharm идеально подходит для работы с фреймворками Django и TensorFlow. Она предоставляет множество встроенных инструментов для работы с кодом, таких как автодополнение, рефакторинг, отладка и анализ качества кода. Важной особенностью PyCharm является поддержка виртуальных окружений, что помогает изолировать зависимости проекта и предотвращать конфликты между различными библиотеками. Для разработки фронтенда будет использована среда WebStorm, которая является одним из лучших редакторов для работы с JavaScript и React. WebStorm предлагает широкие

возможности для быстрой разработки, включая поддержку всех современных стандартов веб-разработки, удобное автодополнение, инструменты для отладки и тестирования, а также возможность интеграции с системами сборки и управления зависимостями, такими как Webpack и NPM. Это позволяет ускорить процесс разработки и повысить стабильность работы приложения. 2.1.3 Система управления базами данных Для хранения данных приложения была выбрана PostgreSQL — одна из самых мощных и надежных реляционных систем управления базами данных. PostgreSQL идеально подходит для работы с большими объемами данных и поддерживает сложные запросы, что необходимо для хранения информации о пользователях, их тренировках и прогрессе. PostgreSQL также известна своей производительностью и масштабируемостью, что позволяет эффективно обрабатывать запросы даже при высокой нагрузке. Система предоставляет широкие возможности для защиты данных, включая шифрование, управление правами доступа и поддержку транзакций, что делает её идеальной для хранения чувствительных данных пользователей. PostgreSQL также поддерживает создание индексов и оптимизацию запросов, что значительно повышает скорость работы приложения. 9 2.1.4

Фреймворки и библиотеки

Для ускорения разработки и улучшения функциональности веб-приложения будут использованы несколько ключевых фреймворков и библиотек:

- Django Rest Framework (DRF): это библиотека для разработки RESTful API на основе фреймворка Django. Она упрощает процесс создания API, предоставляя удобные средства для работы с запросами, сериализации данных и обработки ошибок. DRF обеспечивает высокую безопасность и гибкость при создании взаимодействий между сервером и клиентом.
- React Router: эта библиотека используется для организации маршрутизации в приложении на React. React Router позволяет создавать динамические переходы между различными компонентами приложения, обеспечивая плавную навигацию без необходимости перезагружать страницу. Это особенно важно для веб-приложений, где скорость и отзывчивость интерфейса играют ключевую роль.
- Axios: это популярная библиотека для работы с HTTP-запросами в JavaScript. Axios позволяет взаимодействовать с сервером для отправки данных и получения ответов, что важно для работы с API. Она поддерживает промисы и имеет встроенную обработку ошибок.

2.2 Проектирование диаграммы вариантов использования

Диаграмма вариантов использования (Use Case Diagram) является важным инструментом для визуализации взаимодействий между пользователями (актерами) и системой. Она помогает на концептуальном уровне понять, какие функции система предоставляет своим пользователям и как эти функции будут использоваться в реальных сценариях. В данном проекте диаграмма вариантов использования позволит подробно описать взаимодействие пользователей с системой персонализированных тренировок и рекомендаций по питанию с использованием искусственного интеллекта. В процессе разработки системы были выделены два основных актера, каждый из которых будет взаимодействовать с системой:

- Пользователь.
- Администратор.

Описание актеров разрабатываемой информационной системы представлено в таблице 1. Таблица 1 – Выявление актеров Актер Краткое описание

Пользователь Это человек, который использует систему для получения персонализированных рекомендаций по тренировкам и питанию. Он взаимодействует с системой для выбора тренировок, получения советов и отслеживания прогресса.

Администратор Это пользователь с расширенными правами, который управляет контентом, настройками системы и пользователями. Администратор следит за правильностью данных и их модерацией.

Выявленные варианты использования для ранее указанных актеров представлено в таблицах 2 и 3. Таблица 2 – Варианты использования актера «Администратор» Актер – «Администратор»

Наименование сценария	Формулировка
-----------------------	--------------

Модерация контента	Администратор проверяет и утверждает пользовательский контент, такой как отзывы или результаты тренировок.
--------------------	--

Управление пользователями	Администратор может создавать, удалять и редактировать учетные записи пользователей.
---------------------------	--

Таблица 3 – Варианты использования актера «Пользователь» Актер – «Пользователь»

Наименование сценария	Формулировка
-----------------------	--------------

Пройти опрос	Пользователь может пройти опрос, чтобы получить персонализированные рекомендации по тренировкам и питанию.
--------------	--

Выбор тренировки	Пользователь может выбрать тип тренировки (например, кардио, силовые, растяжка) для создания персонализированного плана тренировок.
------------------	---

Поделиться тренировками в соцсетях	Пользователь может делиться своими тренировочными планами и результатами через социальные сети.
------------------------------------	---

Оценка тренировки	Пользователь может оценить предложенную тренировку (например, поставить лайк или оставить отзыв).
-------------------	---

Получать рекомендации	ИИ анализирует данные о предпочтениях пользователя (интенсивность тренировок, цели, физическое состояние) и предоставляет рекомендации по тренировкам и питанию.
-----------------------	--

Авторизация	Пользователь может войти в систему, используя свою учетную запись или зарегистрировавшись в приложении.
-------------	---

Просмотр личных данных	Пользователь может просматривать свои личные данные, включая историю тренировок и прогресс.
------------------------	---

Изменение личных данных Пользователь может изменять свои личные данные, такие как имя, email и прочее.

На рисунке 1 предсталвлена спроектированная диаграмма вариантов использования системы.

Рисунок 1 – Диаграмма вариантов использования 2.3 Проектирование диаграммы последовательности

Диаграмма последовательности в UML — это диаграмма, которая показывает порядок взаимодействия объектов системы во времени, а именно, как сообщения передаются между объектами для выполнения определённого сценария. На диаграмме отображаются объекты (в виде прямоугольников), а также последовательность их взаимодействий через сообщения, что позволяет понять, как объект выполняет задачи на протяжении времени. Диаграмма последовательности является полезным инструментом для визуализации времени взаимодействий между компонентами системы. Она отображает не только последовательность вызовов функций и методов, но и взаимозависимость объектов, что помогает моделировать сложные сценарии, где важно учитывать порядок и время выполнения операций. Для разрабатываемой системы была создана диаграмма последовательности для прецедента «Регистрация и получение персонализированных рекомендаций», которая представляет собой пошаговое взаимодействие между клиентом и системой. Она описывает процесс от захода пользователя на веб-сайт до получения персонализированных рекомендаций на основе введенных данных. Описание шагов диаграммы последовательности: 1. Клиент открывает сайт: пользователь инициирует процесс, заходя на веб-сайт, что запускает взаимодействие с сервером. 2. Сервер запрашивает данные: сервер обрабатывает запрос клиента и подготавливает необходимые данные для отображения соответствующей страницы, например, страницы входа или регистрации. 3. Клиент вводит данные (регистрация/авторизация): пользователь заполняет форму, предоставляя данные для регистрации или авторизации (например, логин и пароль). 4. Сервер получает страницу: сервер предоставляет клиенту запрашиваемую страницу — это может быть форма авторизации или личный кабинет пользователя. 5. Сервер отправляет данные: данные, введенные пользователем (например, логин и пароль), отправляются на сервер для дальнейшей обработки. 6. База данных проверяет данные: сервер проверяет введенные данные, сверяя их с сохраненными в базе данных (например, логин и пароль пользователя). 7. AI отвечает токеном авторизации: если данные корректны, система (или искусственный интеллект) генерирует токен авторизации и отправляет его пользователю, подтверждая успешный вход в систему. 8. Клиент вводит параметры тела: пользователь вводит дополнительные данные, связанные с его физическими характеристиками (например, вес, рост, уровень физической активности), для получения персонализированных рекомендаций. 9. Сервер отправляет данные: введенные параметры отправляются на сервер для дальнейшей обработки. 10. Сервер делает запрос к ai: сервер передает собранные данные искусственному интеллекту для анализа и генерации рекомендаций, которые могут быть связаны с тренировками или диетой. 11. AI предоставляет рекомендации: ai анализирует данные и на основе полученных параметров возвращает пользователю персонализированные рекомендации, например, план тренировок или диеты. 12. База данных извлекает данные прогресса: в случае необходимости сервер может также запрашивать историю прогресса пользователя из базы данных для уточнения рекомендаций или анализа динамики. 13. Сервер отображает рекомендации: полученные рекомендации и данные прогресса отображаются пользователю на веб-странице, завершив процесс взаимодействия. На рисунке 2 представлена спроектированная диаграмма последовательности прецедента «Регистрация и получение персонализированных рекомендаций».

Рисунок 2 – Диаграмма последовательности

2.4 Проектирование диаграммы деятельности

9 **Диаграмма деятельности — это один из видов диаграмм UML, который описывает последовательность действий и бизнес-процессов в системе.**

Она фокусируется на том, как выполняются процессы в системе, и на переходах между различными состояниями объектов. Диаграмма деятельности помогает понять логику выполнения действий, которые могут быть как линейными, так и ветвящимися, с возможностью параллельного выполнения процессов. Каждое действие или операция в диаграмме деятельности представлено в виде прямоугольника, а переходы между ними — стрелками, которые показывают последовательность. Диаграммы деятельности полезны для описания сложных процессов, где важно четко отследить последовательность действий и условия, при которых происходят изменения в системе. В контексте данной системы для веб-приложения с ИИ, диаграмма деятельности будет использоваться для отображения таких процессов, как: 1. Прохождение опроса для определения телосложения и цели тренировок. Эта диаграмма отображает все шаги от авторизации пользователя до получения персонализированных рекомендаций на основе его ответов. 2. Публикация поста в системе. Диаграмма описывает процесс от отправки поста пользователем до его модерации и уведомления о результате. 3. Управление пользователями администратором. Эта диаграмма демонстрирует процесс, в котором администратор просматривает и управляет учетными записями пользователей, включая возможность их изменения или ограничения доступа. Эти диаграммы позволяют ясно и понятно изобразить порядок действий и взаимоотношения между субъектами системы, что помогает улучшить понимание процессов и их эффективное моделирование. Прецедент №1. «Прохождение опроса для определения телосложения и цели тренировок»: 1. Пользователь открывает приложение и авторизуется: пользователь заходит в приложение и проходит авторизацию, чтобы начать взаимодействие с системой. 2. Пользователь начинает опрос: пользователь инициирует опрос, чтобы система могла собрать данные для персонализированных рекомендаций. 3. Система оценивает ответы пользователя: на основе введенных данных система анализирует ответы и оценивает параметры, такие как телосложение и цели тренировок. 4. Система отображает результаты опроса: система выводит пользователю результаты опроса, включая определение его телосложения и рекомендации по целям тренировок. 5. Пользователь сохраняет результаты и просматривает рекомендации: пользователь может сохранить результаты опроса и просмотреть персонализированные рекомендации по тренировкам и питанию. На рисунке 3 представлена спроектированная диаграмма деятельности прецедента «Прохождение опроса для определения телосложения и цели тренировок».

Рисунок 3 – Диаграмма деятельности прецедента №1

Прецедент №2. «Публикация поста»: 1. Пользователь отправляет пост: пользователь создаёт и отправляет пост в приложение. 2. Система сохраняет

пост и передает на проверку: пост сохраняется в системе и передается на проверку модератору. 3. Администратор проверяет пост: администратор проверяет отправленный пост на соответствие правилам модерации. 4. Система уведомляет пользователя о результате модерации: после проверки система уведомляет пользователя о результатах модерации (одобрен или отклонен пост). На рисунке 4 представлена спроектированная диаграмма деятельности прецедента «Публикация поста».

Рисунок 4 – Диаграмма деятельности прецедента №2

Прецедент №3. «Управление пользователями»: 1. Администратор авторизуется в системе: администратор заходит в систему, чтобы начать процесс управления пользователями. 2. Администратор просматривает список пользователей: администратор просматривает список всех зарегистрированных пользователей в системе. 3. Администратор выбирает пользователя для управления: из списка пользователей администратор выбирает того, с кем он хочет взаимодействовать. 4. Администратор изменяет информацию о пользователе или ограничивает доступ: администратор может изменить информацию о пользователе, а также ограничить его доступ к системе при необходимости. 5. Система обновляет данные пользователя и уведомляет его: система обновляет информацию в базе данных и уведомляет пользователя о внесенных изменениях. На рисунке 5 представлена спроектированная диаграмма деятельности прецедента «Управление пользователями».

Рисунок 5 – Диаграмма деятельности прецедента №3

2.5 Проектирование диаграммы состояний

Диаграмма состояний — это один из типов диаграмм UML, который используется для моделирования изменений состояния объектов системы в ответ на события. Эта диаграмма позволяет наглядно представить, как объект переходит из одного состояния в другое, а также какие действия или события вызывают эти переходы. Диаграммы состояний часто применяются для описания динамики объектов, которые могут находиться в нескольких состояниях в процессе своей жизни. ⑨ Основные элементы диаграммы состояний: · Состояния — различные этапы жизни объекта, которые он может проходить. · Переходы — связи между состояниями, указывающие на возможность перехода из одного состояния в другое при наступлении определенного события. · События — действия или условия, которые вызывают переходы между состояниями. · Начальное и конечное состояние — указывают на начальную и конечную точку жизненного цикла объекта. Диаграммы состояний часто используются для: · Моделирования жизненного цикла объекта. · Отображения поведения системы в ответ на внешние или внутренние события. · Описания процессов, которые требуют изменений состояния в течение времени. В рамках разработки информационных систем, диаграммы состояний помогают четко определить, в каком состоянии может находиться объект в процессе взаимодействия с другими объектами системы. Рассмотрим состояния для следующих объектов: 1. Пользователь. 2. Пост. Объект №1. «Пользователь»: 1. Не зарегистрирован: пользователь не имеет аккаунта в приложении и не может выполнять действия, требующие авторизации. 2. Зарегистрирован: пользователь создал аккаунт, но еще не авторизовался в системе. Может инициировать процесс авторизации. 3. Авторизован: пользователь вошел в систему и имеет доступ ко всем функциям приложения, включая тестирование уровня и просмотр таблицы лидеров. 4. Проходит опрос: пользователь находится в процессе прохождения опроса. 5. Редактирует профиль: пользователь изменяет информацию о своем аккаунте. 6. Просматривает посты: пользователь просматривает свои посты или посты других пользователей. ⑨ Разработанная диаграмма состояний для объекта «Пользователь» представлена на рисунке 6.

Рисунок 6 – Диаграмма состояний для объекта №1

Объект №2. «Пост»: 1. Создан: пост был написан пользователем, но еще не отправлен на модерацию. 2. Ожидает модерации: пост отправлен на модерацию и ждет проверки администратором. 3. Опубликован: пост одобрен администратором и виден другим пользователям. 4. Удален: пост был удален пользователем или администратором. ⑨ Разработанная диаграмма состояний для объекта «Одежда» представлена на рисунке 7.

Рисунок 7 – Диаграмма состояний для объекта №2

2.6 Проектирование диаграммы сущность-связь

Диаграмма сущностей-связей (ERD — Entity-Relationship Diagram) является инструментом для проектирования базы данных, который описывает взаимосвязи между сущностями, их атрибутами и типами данных. В рамках разработки системы для платформы NeuroFit, ERD используется для представления структуры базы данных, включая сущности, их атрибуты и связи, что позволяет оптимально организовать хранение данных и их взаимодействие. Рассмотрим каждую сущность и детализируем информацию о ней: 1. Сущность `users_customuser` (Пользователь) представляет информацию о пользователях системы. Она содержит следующие атрибуты: · `id` — уникальный идентификатор пользователя, который является первичным ключом; · `password` — хэшированный пароль пользователя; · `last_login` — дата последнего входа пользователя; · `is_superuser` — флаг суперпользователя; · `username` — имя пользователя; · `email` — электронная почта пользователя; · `date_of_birth` — дата рождения пользователя; · `gender` — пол пользователя; · `height` — рост пользователя; · `weight` — вес пользователя; · `goal` — цель пользователя пользователя; · `has_equipment` — наличие оборудования пользователя; · `avatar` — путь к аватару пользователя; · `is_active` — активность аккаунта пользователя; · `is_staff` — флаг сотрудника; · `created_at` — дата создания аккаунта; · `updated_at` — дата обновления аккаунта. 2. Сущность `blog_post` (Публикация) представляет информацию о публикациях пользователей в системе. Она включает в себя следующие атрибуты: · `id` — уникальный идентификатор поста, который является первичным ключом; · `user_id` — идентификатор пользователя, который создал пост; · `content` — содержимое публикации; · `image` — путь к изображению, связанному с публикацией (если таковое имеется); · `is_approved` — статус одобрения публикации администратором; · `created_at` — дата и время создания публикации; · `updated_at` — дата и время последнего обновления публикации. 3. Сущность `blog_comment` (Комментарии) представляет информацию о комментариях к публикациям. Она содержит следующие атрибуты: · `id` — уникальный идентификатор комментария, который является первичным ключом; · `post_id` — идентификатор поста, к которому относится комментарий; · `user_id` — идентификатор пользователя, который оставил комментарий; · `content` — текст комментария; · `created_at` — дата и время создания комментария. 4. Сущность

progress_progresschart (Прогресс) представляет информацию о прогрессе пользователей в системе. Она включает следующие атрибуты: · id — уникальный идентификатор записи о прогрессе, который является первичным ключом; · user_id — идентификатор пользователя, который оставил запись о прогрессе; · date — дата, на которую сделана запись о прогрессе; · weight — вес пользователя в килограммах на указанную дату; · height — рост пользователя в сантиметрах на указанную дату; · photo — фотография прогресса пользователя; · notes — заметки пользователя о своем прогрессе; · created_at — дата и время создания записи о прогрессе. 5. Сущность nutrition_nutrition (Рацион питания) представляет информацию питания пользователя. Она включает следующие атрибуты: · id — уникальный идентификатор записи о питании, который является первичным ключом; · user_id — идентификатор пользователя, который оставил запись о питании; · date — дата, на которую сделана запись о питании; · meals — приемы пищи пользователя на указанную дату; · calories — количество калорий приема пищи; · created_at — дата и время создания записи о приеме пищи. 6. Сущность workouts_workout (Тренировки) представляет информацию тренировках пользователя. Она включает следующие атрибуты: · id — уникальный идентификатор тренировки, который является первичным ключом; · user_id — идентификатор пользователя, который оставил запись о тренировке; · date — дата тренировки; · plan — план тренировки пользователя в формате JSON; · completed — статус выполнения тренировки; · created_at — дата и время создания записи о тренировке. 7. Сущность blog_like (Лайки) представляет информацию лайках к постам пользователя. Она включает следующие атрибуты: · id — уникальный идентификатор оценки, который является первичным ключом; · user_id — идентификатор пользователя, который оставил оценку; · post_id — идентификатор поста, который был оценен; · created_at — дата и время оценки. ⑨ На рисунке 8 представлена спроектированная диаграмма сущность-связь системы.

Рисунок 8 – Диаграмма сущность-связь

2.7 Проектирование базы данных

Проектирование базы данных — это процесс создания структуры базы данных, которая будет эффективно хранить и обрабатывать информацию, а также обеспечивать интеграцию с приложениями и другими системами. Процесс проектирования базы данных включает несколько этапов, каждый из которых направлен на создание удобной и производительной системы для хранения данных. Важнейшие этапы проектирования базы данных: 1. Анализ требований. На этом этапе необходимо определить, какие данные будут храниться в базе данных и какие процессы или функции они будут поддерживать. Это требует понимания целей системы и того, как данные будут использоваться. Например, в системе для фитнес-платформы важными данными являются информация о пользователях, их прогрессе, публикациях и комментариях, а также соответствующие связи между ними. 2. Концептуальное проектирование. Этот этап включает в себя создание концептуальной модели данных, которая представляет собой описание всех сущностей (например, пользователь, публикация, комментарий) и их отношений. Концептуальная модель описывает, какие данные должны быть собраны и как они связаны. На основе этого создается диаграмма сущностей-связей (ERD), которая визуализирует взаимодействие между сущностями и их атрибутами. 3. Логическое проектирование. ⑧ На этом этапе концептуальная модель преобразуется в логическую модель базы данных.

Логическое проектирование описывает, как данные будут организованы и какие типы данных будут использоваться для каждого атрибута. В отличие от концептуальной модели, логическая модель уже учитывает особенности выбранной системы управления базами данных (СУБД), такие как типы данных, ограничения целостности и индексы. Логическое проектирование также включает определение связей между таблицами, таких как первичные и внешние ключи. 4. Физическое проектирование. Физическое проектирование фокусируется на том, как данные будут физически храниться в базе данных. Это включает в себя выбор конкретных методов хранения данных, оптимизацию производительности запросов и индексацию для ускорения поиска и доступа к данным. Также важно учитывать вопросы безопасности, резервного копирования и восстановления данных. 5. ⑨ Нормализация базы данных. Нормализация — это процесс организации данных в базе таким образом, чтобы минимизировать избыточность и исключить аномалии обновления. Нормализация включает в себя разделение данных на таблицы и определение отношений между ними. На разных этапах нормализации (от 1NF до 5NF) устраняются дублирующиеся данные, а связи между таблицами упорядочиваются. В результате получается структура базы данных, которая гарантирует её целостность и уменьшает вероятность ошибок при изменении данных. 6. ⑨

Денормализация базы данных. В некоторых случаях для улучшения производительности работы с базой данных может быть полезно денормализовать её. Это означает объединение таблиц или повторное использование данных для ускорения чтения данных. Однако денормализация может привести к увеличению избыточности и возникновению аномалий при обновлениях, поэтому её следует использовать осторожно, в зависимости от потребностей системы. 7. Определение индексов и оптимизация запросов. На этапе физического проектирования особое внимание уделяется созданию индексов для ускорения выполнения запросов, особенно в больших базах данных. Индексы помогают быстро находить записи в таблицах по определенным полям. Важно выбирать поля для индексации, которые используются в условиях WHERE или JOIN в SQL-запросах. Оптимизация запросов помогает улучшить общую производительность системы. 8. Обеспечение целостности данных. Целостность данных гарантирует, что база данных будет содержать точную и актуальную информацию. Это достигается через применение различных типов ограничений: · Ограничения уникальности — для обеспечения уникальности значений в столбцах. · Ограничения внешних ключей — для поддержания корректных связей между таблицами. · Ограничения NOT NULL — для гарантии наличия данных в обязательных полях. · Ограничения CHECK — для проверки корректности данных при их вставке или обновлении. 9. Резервное копирование и восстановление. Важно предусмотреть процесс резервного копирования базы данных, чтобы защитить данные от потерь в случае сбоя системы. Также следует продумать процедуры восстановления данных и тестировать их регулярность, чтобы в случае необходимости можно было быстро восстановить работоспособность системы.

ГЛАВА 3 ПОЛЬЗОВАТЕЛЬСКАЯ ДОКУМЕНТАЦИЯ

В данной главе представлена документация, предназначенная для пользователей и разработчиков информационной системы. Пользовательская документация охватывает инструкции по использованию системы для конечных пользователей, а руководство программиста предоставляет технические подробности и описание интерфейсов системы, которые могут быть полезны для разработки, интеграции и поддержки системы. 3.1 Руководство программиста

Проект реализован с использованием фреймворка Django. Он состоит из двух основных приложений (users, coaches) и конфигурационной папки main. Ниже приведено подробное описание всех директорий и файлов backend-части проекта. 1. Корневые файлы проекта · manage.py – основной управляющий скрипт Django-проекта. Используется для выполнения административных команд: запуск сервера, выполнение миграций, создание суперпользователя и т.д. · requirements.txt – файл со списком зависимостей проекта. Позволяет быстро установить все нужные библиотеки с помощью команды `pip install -r requirements.txt`. 2. Папка main – конфигурация проекта

Используется для настройки и управления общими параметрами Django-проекта. · main/init.py – делает папку main Python-модулем. Обычно содержит базовую инициализацию (может быть пустым). · main/asgi.py – точка входа для ASGI-приложения. Используется при запуске проекта с использованием асинхронных серверов, например, Uvicorn или Daphne. · main/wsgi.py – точка входа для WSGI-приложения. Применяется для деплоя на сервер, например, через Gunicorn или uWSGI. · main/settings.py – основной конфигурационный файл. Здесь задаются: ⑦ **параметры подключения к базе данных**; список установленных приложений; middleware; настройки локали и времени; пути к статическим и медиа-файлам; ключи безопасности и разрешённые хосты. · main/urls.py – корневой маршрутизатор проекта. Здесь подключаются URL-адреса из приложений (users, coaches и т.д.), а также административная панель. 3. Папка users – приложение, отвечающее за пользователей

Содержит логику для регистрации, аутентификации, управления пользователями. · users/init.py – инициализация пакета users. · users/admin.py – регистрация моделей пользователей в админ-панели Django. Позволяет управлять пользователями из интерфейса администратора. · users/apps.py – конфигурация приложения users, указывается имя приложения и путь для автоподключения. · users/models.py – описывает модели базы данных, связанные с пользователями. Например: User, Profile и другие. · users/serializers.py – содержит сериализаторы для преобразования моделей в формат JSON и обратно. Используются в API. · users/views.py – содержит представления (views), реализующие бизнес-логику: регистрация, логин, редактирование профиля и прочее. · users/urls.py – локальный маршрутизатор приложения. Определяет эндпоинты, доступные в users. · users/tests.py – модуль для написания автоматических тестов. Проверяет корректность работы пользовательских функций. · users/migrations/ – папка с миграциями для моделей приложения. Хранит файлы миграций, автоматически создаваемые командой makemigrations. 4. Папка coaches – приложение для работы с тренерами и их услугами

Реализует логику, связанную с тренерами, расписаниями, услугами, отзывами и т.д. · coaches/init.py – инициализация пакета coaches. · coaches/admin.py – регистрация моделей тренеров, расписаний и других сущностей в административной панели. · coaches/apps.py – конфигурация приложения coaches. · coaches/models.py – определяет модели, такие как Coach, Service, Schedule, которые отображают структуру базы данных. · coaches/views.py – обработчики запросов для отображения, создания и редактирования данных, связанных с тренерами. · coaches/urls.py – маршруты, относящиеся к функционалу тренеров. · coaches/tests.py – содержит тесты для проверки логики, связанной с тренерами и их записями. · coaches/migrations/ – файлы миграций для моделей coaches. 3.2 Руководство пользователя Настоящее руководство предназначено для пользователей и администраторов веб-приложения фитнес-центра, разработанного для индивидуального подбора тренировок и питания с помощью ИИ. В зависимости от роли, интерфейс системы предлагает различный набор возможностей и функций. В приложении реализованы две основные роли: 1. Пользователь — клиент фитнес-центра, желающий воспользоваться сервисом для подбора персонального плана питания и тренировок. После регистрации и авторизации пользователь получает доступ к чату с ИИ-тренером, персональному меню и блоку с тренировками. Также он может редактировать личные данные и цели в профиле. 2. Администратор — сотрудник, ответственный за управление системой через админ-панель. Администратор имеет возможность просматривать, редактировать и удалять информацию о пользователях, а также добавлять новых. Админ-панель предоставляет доступ ко всем основным сущностям системы для эффективного контроля и поддержки актуальности данных. В данном разделе подробно описан пользовательский интерфейс, включая главные страницы, функциональные возможности и порядок взаимодействия с системой. Все действия сопровождаются пояснениями и иллюстрациями для более удобного понимания. 3.2.1 Руководство роли «Пользователь» На главной странице представлена основная информация о сервисе, его целях и возможностях. В верхней части экрана размещена навигационная панель, в которой доступны кнопки «Вход» и «Регистрация». После авторизации они заменяются на кнопки «ИИ-тренер», «Тренировки», «Прогресс». На рисунке 9 представлен интерфейс главной страницы.

Рисунок 9 – Главная страница

Чтобы получить доступ ко всем функциям системы, пользователю необходимо авторизоваться. Это позволяет пройти опрос и получить персональные рекомендации по тренировкам и питанию. На странице авторизации пользователь заполняет поля «Логин» и «Пароль». При вводе некорректных данных появляется сообщение об ошибке. После успешного ввода данных открывается доступ к основному функционалу. На рисунке 10 представлено окно авторизации.

Рисунок 10 – Авторизация

Регистрация необходима для создания личного кабинета. Пользователь указывает логин, email и пароль. После регистрации потребуется заполнить анкету для подбора индивидуальных рекомендаций. На рисунке 11 показана страница регистрации.

Рисунок 11 – Регистрация

После регистрации пользователь может перейти к настройке профиля. Здесь он указывает параметры: пол, вес, рост и цель тренировок. Эти данные используются для персонализации рекомендаций. На рисунке 12 показано окно настройки профиля.

Рисунок 12 – Настройка профиля

После авторизации пользователь может перейти в раздел своего профиля, где отображаются все личные данные, введённые при регистрации или

настроенные позже. В этом разделе можно увидеть параметры пользователя, такие как имя, пол, вес, рост и цель тренировок. Эти данные являются основой для подбора персонализированных рекомендаций и тренировок. На рисунке 13 представлен интерфейс профиля пользователя.

Рисунок 13 – Окно с профилем

В разделе «ИИ-тренер» пользователь может пообщаться с интеллектуальным тренером, который предоставляет рекомендации по тренировкам, питанию и образу жизни, основываясь на индивидуальных данных пользователя. ИИ-тренер анализирует информацию о пользователе, такую как вес, рост, цель тренировок, и предлагает наиболее подходящие программы и советы. На рисунке 14 показано окно с чатом ИИ-тренера.

Рисунок 14 – Окно с чатом ИИ-тренера

В разделе «Питание» пользователь может получить персонализированное меню, составленное на основе его физических данных и целей. Программа автоматически анализирует введенные параметры, такие как пол, возраст, вес, рост и цель (например, похудение, набор массы, поддержание формы), и генерирует рекомендации по питанию. На рисунке 15 представлено окно с разделом «Питание», где пользователи могут ознакомиться с рекомендованным меню.

Рисунок 15 – Окно с питанием

В разделе «Тренировки» отображаются программы тренировок, подобранные для пользователя на основе его параметров и целей. Программа автоматически составляется с учётом данных, введенных пользователем, и может включать различные типы тренировок, такие как кардио, силовые упражнения, растяжка и другие. На рисунке 16 показано окно с тренировки, где пользователь может выбрать и начать выполнение упражнений, следуя расписанию.

Рисунок 16 – Окно с тренировками

3.2.2 Руководство роли «Администратор» На главной странице админ-панели администратора встречается окно авторизации. В нем необходимо ввести имя пользователя и пароль для доступа к функционалу панели управления. После успешного ввода данных администратор попадает в панель, где доступны все функции для управления пользователями и контентом. На рисунке 17 представлено окно авторизации администратора.

Рисунок 17 – Авторизация для администратора

После авторизации администратор попадает на главную страницу админ-панели. Здесь отображаются различные сущности, которые администратор может изменять или удалять. На рисунке 18 показана главная страница админ-панели.

Рисунок 18 – Главная страница админ-панели

На странице управления пользователями администратор видит список всех зарегистрированных пользователей. Для изменения данных конкретного пользователя необходимо выбрать его из списка. Также на этой странице доступна кнопка для добавления нового пользователя в систему. На рисунке 19 показан список пользователей в админ-панели.

Рисунок 19 – Список пользователей в админ-панели

После выбора конкретного пользователя, администратор попадает на страницу с его личными данными. Здесь можно изменять параметры пользователя, такие как имя, email, логин, а также цели, вес и рост. Эти данные могут быть обновлены в зависимости от изменений, сделанных пользователем. На рисунке 20 изображена страница с изменением данных пользователя.

Рисунок 20 – Изменение данных пользователя

Для добавления нового пользователя администратор использует специальную форму, где нужно указать логин, email и пароль. Также может потребоваться заполнение дополнительной информации о пользователе, такой как его параметры для персонализации рекомендаций. На рисунке 21 показана форма для добавления нового пользователя.

Рисунок 21 – Изменение данных пользователя

ГЛАВА 4 ТЕСТИРОВАНИЕ

Тестирование является критически важным этапом разработки программного обеспечения, так как позволяет убедиться в корректности работы системы, выявить возможные ошибки и обеспечить соответствие функциональным требованиям. В рамках данного проекта тестирование проводилось для модуля работы с пользователями, включая регистрацию, аутентификацию и управление профилем. Для тестирования использовался фреймворк Django REST Framework, который предоставляет инструменты для написания модульных и интеграционных тестов. В частности, были написаны тесты для проверки следующих сценариев: 1. Регистрация пользователя: ⑨ проверка корректности создания нового пользователя в системе. 2. Аутентификация пользователя и получение профиля: проверка входа в систему и доступа к данным профиля. 3. Обновление профиля пользователя: проверка возможности изменения данных профиля, таких как email. Тесты выполнялись автоматически, и их результаты показали успешное прохождение всех проверок, что подтверждает корректность реализации функционала. Время выполнения тестов

составило 890 мс, что свидетельствует об эффективности кода. 1. Тест №1. test_user_registration. Этот тест проверяет корректность регистрации нового пользователя. Он отправляет POST-запрос на эндпоинт регистрации с данными пользователя (username, email, password) и проверяет: · Статус ответа (должен быть 201 Created). · Наличие пользователя в базе данных после регистрации. 2. Тест №2. test_user_login_and_get_profile. Тест проверяет процесс аутентификации пользователя и доступ к его профилю. Сначала создается пользователь через регистрацию, затем: · Отправляется POST-запрос на эндпоинт получения токена (JWT) с данными для входа. · Проверяется статус ответа (должен быть 200 OK) и наличие токена доступа. · Используя полученный токен, отправляется GET-запрос на эндпоинт профиля и проверяется: статус ответа (должен быть 200 OK), совпадение username в ответе с данными зарегистрированного пользователя. 3. Тест №3. test_profile_update. Тест проверяет возможность обновления данных профиля пользователя. После регистрации и аутентификации: · Отправляется PATCH-запрос на эндпоинт профиля с новым email. · Проверяется статус ответа (должен быть 200 OK). · Проверяется, что email в ответе соответствует обновленному значению. Все тесты успешно проходят, что подтверждает работоспособность и надежность реализованного функционала. На рисунке 22 представлен результат тестирования сценариев.

Рисунок 22 – Unit тестирование

ЗАКЛЮЧЕНИЕ

В рамках выполнения дипломного проекта была разработана информационная система для фитнес-центра с использованием искусственного интеллекта для консультаций и подбора персональных тренировок, а также составления плана питания. В ходе работы над проектом были рассмотрены ключевые аспекты потребностей пользователей, а также требования к функционалу и интерфейсу системы. Основной целью проекта было создание удобного и эффективного веб-приложения, которое бы обеспечивало персонализированный подход для пользователей при выборе тренировок и питания. В ходе работы были достигнуты все поставленные цели, и реализованы основные функциональные требования. В процессе разработки системы были выполнены следующие задачи: 1. Анализированы потребности целевой аудитории, что позволило учесть предпочтения пользователей при создании функционала системы. 2. Определены требования к информационной системе, включая функционал для пользователей и администраторов. 3. Разработана архитектура приложения, обеспечивающая стабильную работу системы и эффективное взаимодействие всех её компонентов. 4. Создана структура базы данных, которая позволяет эффективно хранить информацию о пользователях, тренировках, рекомендациях и прогрессе. 5. Реализован интуитивно понятный интерфейс, который обеспечивает удобство взаимодействия с системой как для пользователей, так и для администраторов. 6. Внедрены ключевые функции системы, такие как регистрация и авторизация пользователей, подбор тренировок и планов питания, а также мониторинг прогресса. Таким образом, разработанная информационная система существенно улучшает процессы планирования и организации тренировок для пользователей, предлагая им персонализированные рекомендации. Также система позволяет администраторам эффективно управлять данными пользователей и контролировать актуальность информации. В конечном счете, это приложение значительно повышает эффективность и удобство работы в фитнес-центре, а также способствует улучшению качества обслуживания клиентов.

9 СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Кузнецов В. Python для начинающих. Изд. Вильямс, 2020 – 416 с. 2. Шилдт Х. Python. Самоучитель для начинающих. Изд. Питер, 2021 – 400 с. 3. Гаскойн А. Python. Разработка приложений. Изд. О'Рейли, 2021 – 496 с. 4. Райан С. Практическое руководство по Python. Изд. Эксмо, 2021 – 544 с. 5. Сваровский В. Программирование на Python для научных вычислений. Изд. Вильямс, 2021 – 460 с. 6. Фримен Э. Программирование на Python. Изд. Питер, 2022 – 528 с. 7. Хейтерс Д. Python для анализа данных. Изд. ДМК Пресс, 2022 – 380 с. 8. Лоуренс Г. Секреты Python. Современные техники разработки. Изд. Эксмо, 2022 – 512 с. 9. Книги Р. Основы Python. Изд. Вильямс, 2021 – 320 с. 10. Ван Х. Использование Python для решения инженерных задач. Изд. Питер, 2023 – 462 с. 11. «Python Documentation» [Электронный ресурс]: официальная документация Python. URL: <https://docs.python.org/3/> (дата обращения: 05.04.2025) Режим доступа: свободный
12. «Flask Documentation» [Электронный ресурс]: документация по Flask для создания веб-приложений. URL: <https://flask.palletsprojects.com/> (дата обращения: 05.04.2025) Режим доступа: свободный
13. «Django Documentation» [Электронный ресурс]: официальная документация Django. URL: <https://docs.djangoproject.com/> (дата обращения: 05.04.2025) Режим доступа: свободный
14. «SQLAlchemy Documentation» [Электронный ресурс]: документация по SQLAlchemy для работы с базами данных. URL: <https://www.sqlalchemy.org/> (дата обращения: 05.04.2025) Режим доступа: свободный
15. «PostgreSQL Documentation» [Электронный ресурс]: официальная документация PostgreSQL. URL: <https://www.postgresql.org/docs/> (дата обращения: 05.04.2025) Режим доступа: свободный
16. «RESTful API Design» [Электронный ресурс]: руководство по проектированию RESTful API. URL: <https://restfulapi.net/> (дата обращения: 05.04.2025) Режим доступа: свободный
17. Зайцев Н. Разработка REST API с использованием Python. Изд. Вильямс, 2022 – 400 с. 18. Рейнольдс Р. Разработка веб-приложений на Python. Изд. Питер, 2022 – 522 с. 19. Нилссон Д. Архитектура Python. Изд. Вильямс, 2021 – 350 с. 20. Смит Дж. Эффективное использование Python в веб-разработке. Изд. Эксмо, 2022 – 380 с.

ПРИЛОЖЕНИЕ А СЛОВАРЬ ДАННЫХ

Таблица 1 – Определение характеристик атрибутов

Таблица	Поле	Тип данных	Ключ
users_customuser	id	bigint	PK

password varchar(128)

last_login datetime

is_superuser boolean

username varchar(50)

email varchar(254)

date_of_birth date

gender varchar(10)

height integer

weight numeric(5,2)

goal varchar(20)

has_equipment boolean

avatar varchar(100)

is_active boolean

is_staff boolean

created_at datetime

updated_at datetime

nutrition_nutrition	id	bigint	PK
---------------------	----	--------	----

date date

meals jsonb

calories integer

created_at datetime

updated_at datetime

user_id bigint FK (users_customuser)

workouts_workout	id	bigint	PK
------------------	----	--------	----

date date

plan jsonb

completed boolean

created_at datetime

user_id bigint FK (users_customuser)

Продолжение таблицы 1 progress_progresschart id bigint PK

date date

weight numeric(5,2)

height integer

photo varchar(100)

notes text

user_id bigint FK (users_customuser)

blog_post id bigint PK

content text

image varchar(100)

created_at datetime

user_id bigint FK (users_customuser)

is_approved boolean

blog_like id bigint PK

created_at datetime

post_id bigint FK (blog_post.id)

user_id bigint FK (users_customuser)

blog_comment id bigint PK

content text

created_at datetime

post_id bigint FK (blog_post.id)

user_id bigint FK (users_customuser)

ПРИЛОЖЕНИЕ Б ЛИСТИНГ ПРОГРАММЫ

manage.py

import os

import sys

```
def main(): ⑬ os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'main.settings') try: ⑭ from django.core.management import execute_from_command_line
except ImportError as exc: raise ImportError( "Couldn't import Django. Are you sure it's installed and " "available on your PYTHONPATH environment variable? Did you " "forget to activate a virtual environment?" ) from exc execute_from_command_line(sys.argv)
```

```
if __name__ == '__main__': main()
```

requirements.txt

⑮ asgiref==3.8.1

Django==4.2

django-cors-headers==4.7.0

django-environ==0.12.0

⑮ `djangoframework==3.15.2`

`djangoframework_simplejwt==5.4.0`

`psycpg2==2.9.10`

⑤ `PyJWT==2.10.1`

⑮ `sqlparse==0.5.3`

`typing_extensions==4.12.2`

`tzdata==2025.1`

`admin.py`

⑮ `from django.contrib import admin`

⑰ `# Register your models here.`

`apps.py`

⑮ `from django.apps import AppConfig`

`class CoachesConfig(AppConfig):` ⑲ `default_auto_field = 'django.db.models.BigAutoField'` ⑭ `name = 'coaches'`

`models.py`

⑳ `from django.db import models`

⑮ `# Create your models here.`

`tests.py`

⑳ `from django.test import TestCase`

⑰ `# Create your tests here.`

`urls.py`

⑳ `from django.urls import path`

⑳ `from .views import GenerateView`

`urlpatterns = [path('generate/', GenerateView.as_view(), name='generate'),]`

`views.py`

`import openai`

`from openai import OpenAI`

⑮ `from rest_framework.permissions import IsAuthenticated`

`from rest_framework.response import Response`

`from rest_framework.views import APIView`

`from backend.main import settings`

`client = OpenAI(api_key=settings.OPENAI_API_KEY)`

`class GenerateView(APIView):` ⑮ `permission_classes = [IsAuthenticated]`

`def post(self, request):` `messages = request.data["messages"]` `try:` `completion = client.chat.completions.create(model="gpt-4o-mini", messages=messages,)`

```
content = completion.choices[0].message.content return Response({"answer": content})
```

```
except openai.OpenAIError as e: ⑮return Response({"error": f"Ошибка OpenAI: ⑮{str(e)}", status=500)
```

asgi.py

```
import os
```

```
⑲from django.core.asgi import get_asgi_application
```

```
⑲os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'main.settings')
```

```
⑲application = get_asgi_application()
```

settings.py

```
import environ
```

```
from datetime import timedelta
```

```
⑲from pathlib import Path
```

```
BASE_DIR = Path(__file__).resolve().parent.parent
```

```
ENV_FILE = BASE_DIR / '.env'
```

```
env = environ.Env() environ.Env.read_env(ENV_FILE)
```

```
SECRET_KEY = env('SECRET_KEY') DEBUG = env.bool('DEBUG', default=True) ALLOWED_HOSTS = env.list('ALLOWED_HOSTS', default=[])
```

```
⑲INSTALLED_APPS = [ 'django.contrib.admin', 'django.contrib.auth', 'django.contrib.contenttypes', 'django.contrib.sessions', 'django.contrib.messages',  
'django.contrib.staticfiles',
```

```
'rest_framework', 'rest_framework_simplejwt', 'corsheaders', 'drf_yasg',
```

```
'users', ]
```

```
REST_FRAMEWORK = { 'DEFAULT_AUTHENTICATION_CLASSES': [ 'rest_framework_simplejwt.authentication.JWTAuthentication', ], }
```

```
SIMPLE_JWT = { 'ACCESS_TOKEN_LIFETIME': timedelta(minutes=15), 'REFRESH_TOKEN_LIFETIME': timedelta(days=14), 'ROTATE_REFRESH_TOKENS': False,  
'BLACKLIST_AFTER_ROTATION': True, }
```

```
⑲MIDDLEWARE = [ 'django.middleware.security.SecurityMiddleware', 'django.contrib.sessions.middleware.SessionMiddleware',  
'django.middleware.common.CommonMiddleware', 'django.middleware.csrf.CsrfViewMiddleware', 'django.contrib.auth.middleware.AuthenticationMiddleware',  
'django.contrib.messages.middleware.MessageMiddleware', 'django.middleware.clickjacking.XFrameOptionsMiddleware',  
'corsheaders.middleware.CorsMiddleware', ]
```

```
ROOT_URLCONF = 'main.urls'
```

```
TEMPLATES = [ { 'BACKEND': 'django.template.backends.django.DjangoTemplates', 'DIRS': [], 'APP_DIRS': True, 'OPTIONS': { 'context_processors': ⑲[  
'django.template.context_processors.debug', 'django.template.context_processors.request', 'django.contrib.auth.context_processors.auth',  
'django.contrib.messages.context_processors.messages', ], }, }, ]
```

```
WSGI_APPLICATION = 'main.wsgi.application'
```

```
DATABASES = { 'default': { 'ENGINE': ⑲'django.db.backends.postgresql', 'NAME': env('DB_NAME'), 'USER': env('DB_USER'), 'PASSWORD': env('DB_PASSWORD'),  
'HOST': env('DB_HOST', default='localhost'), 'PORT': env('DB_PORT', default='5432'), }, }
```

```
⑲AUTH_PASSWORD_VALIDATORS = [ { 'NAME': ⑲'django.contrib.auth.password_validation.UserAttributeSimilarityValidator', }, { 'NAME':  
'django.contrib.auth.password_validation.MinimumLengthValidator', }, { 'NAME': ⑲'django.contrib.auth.password_validation.CommonPasswordValidator', }, {  
'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator', }, ]
```

```
LANGUAGE_CODE = 'ru-ru' ⑲TIME_ZONE = 'UTC' ⑲USE_I18N = True
```

②⑤ USE_TZ = True

STATIC_URL = 'static/'

①⑤ CORS_ALLOW_ALL_ORIGINS = True

①⑨ DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'

①⑤ AUTH_USER_MODEL = 'users.CustomUser'

OPENAPI_API_KEY = env('OPENAPI_API_KEY')

urls.py

①⑥ from django.contrib import admin

from django.urls import path, include

from drf_yasg import openapi

from drf_yasg.views import get_schema_view

①⑤ from rest_framework import permissions

from rest_framework_simplejwt.views import TokenObtainPairView, TokenRefreshView

schema_view = get_schema_view(openapi.Info(title="NeuroFit API", default_version='v1',), public=True, permission_classes=[permissions.AllowAny],)

urlpatterns = [path('admin/', admin.site.urls), path('api/users/', include('users.urls')), path('api/coaches/', include('coaches.urls')), path('api/token/', TokenObtainPairView.as_view(), name='token_obtain_pair'), path('api/token/refresh/', TokenRefreshView.as_view(), name='token_refresh'), path('swagger/', schema_view.with_ui('swagger', cache_timeout=0), name='schema-swagger-ui'),]

wsgi.py

import os

①③ from django.core.wsgi import get_wsgi_application

os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'main.settings')

application = get_wsgi_application()

__init__.py

admin.py

①⑥ from django.contrib import admin

from .models import Custom User

@admin.register(CustomUser) class CustomUserAdmin(admin.ModelAdmin): list_display = ('id', 'username', 'email', 'is_admin', 'created_at', 'updated_at') list_filter = ('is_admin', 'gender') search_fields = ('username', 'email')

apps.py

①⑧ from django.apps import AppConfig

①⑤ class UsersConfig(AppConfig): ①⑨ default_auto_field = 'django.db.models.BigAutoField' ①④ name = 'users'

models.py

①⑦ from django.contrib.auth.models import AbstractBaseUser, BaseUserManager, PermissionsMixin

②⑦ from django.db import models

```
from django.utils.timezone import now
```

```
class CustomUserManager(BaseUserManager): def create_user(self, username, email, password=None, **extra_fields): if not email: raise ValueError("The Email field must be set") email = self.normalize_email(email) extra_fields.setdefault('is_admin', False) user = self.model(username=username, email=email, **extra_fields) user.set_password(password) user.save(using=self._db) return user
```

```
def create_superuser(self, username, email, password=None, **extra_fields): extra_fields.setdefault('is_admin', True) extra_fields.setdefault('is_staff', True) extra_fields.setdefault('is_superuser', True)
```

```
return self.create_user(username, email, password, **extra_fields)
```

```
class CustomUser(AbstractBaseUser, PermissionsMixin): id = models.AutoField(primary_key=True) username = models.CharField(max_length=50, unique=True) email = models.EmailField(max_length=100, unique=True) password_hash = models.CharField(max_length=255) date_of_birth = models.DateField(blank=True, null=True) gender = models.CharField(max_length=10, blank=True, null=True) height = models.IntegerField(blank=True, null=True) weight = models.DecimalField(max_digits=5, decimal_places=2, blank=True, null=True) goal = models.CharField(max_length=255, blank=True, null=True) is_admin = models.BooleanField(default=False) is_staff = models.BooleanField(default=False) is_active = models.BooleanField(default=True) created_at = models.DateTimeField(default=now) updated_at = models.DateTimeField(auto_now=True)
```

```
objects = CustomUserManager()
```

```
15 USERNAME_FIELD = 'username' REQUIRED_FIELDS = ['email', 'password_hash']
```

```
class Meta: verbose_name = "Пользователь" verbose_name_plural = "Пользователи"
```

```
22 def __str__(self): return self.username
```

```
serializers.py
```

```
15 from rest_framework import serializers
```

```
20 from .models import CustomUser
```

```
15 class RegisterSerializer(serializers.ModelSerializer): password = serializers.CharField(write_only=True)
```

```
class Meta: 15 model = CustomUser fields = ('username', 'email', 'password', )
```

```
def validate_password(self, value): if len(value) < 8: raise serializers.ValidationError("Пароль должен содержать минимум 8 символов.") return value
```

```
15 def create(self, validated_data): user = CustomUser.objects.create_user( username=validated_data['username'], email=validated_data['email'], password=validated_data['password'], ) return user
```

```
15 class UserSerializer(serializers.ModelSerializer):
```

```
class Meta: model = CustomUser fields = ('id', 'username', 'email', 'date_of_birth', 'gender', 'height', 'weight', 'goal', 'created_at', 'updated_at', ) read_only_fields = ('id', 'created_at', 'updated_at')
```

```
tests.py
```

```
21 from django.test import TestCase
```

```
17 # Create your tests here.
```

```
urls.py
```

```
22 from django.urls import path
```

```
from .views import RegisterView, ProfileView
```

```
urlpatterns = [ path('register/', RegisterView.as_view(), name='register'), path('profile/', ProfileView.as_view(), name='profile'), ]
```

```
views.py
```

```
15 from rest_framework import generics
```

```
from rest_framework.permissions import AllowAny
```

```
from rest_framework.permissions import IsAuthenticated
```

```
from .serializers import RegisterSerializer, UserSerializer
```

```
class RegisterView(generics.CreateAPIView): serializer_class = RegisterSerializer permission_classes = [AllowAny]
```

```
class ProfileView(generics.RetrieveUpdateAPIView): serializer_class = UserSerializer permission_classes = [IsAuthenticated]
```

```
def get_object(self): return self.request.user
```

0001_initial.py # Generated by Django 4.2 on 2025-02-11 15:14

```
18 from django.db import migrations, models
```

```
import django.utils.timezone
```

```
18 class Migration(migrations.Migration):
```

```
initial = True
```

```
dependencies = [ ('auth', '0012_alter_user_first_name_max_length'), ]
```

```
operations = [ migrations.CreateModel( name='CustomUser', fields=[ ('password', models.CharField(max_length=128, verbose_name='password')), ('last_login', models.DateTimeField(blank=True, null=True, verbose_name='last login')), ('is_superuser', models.BooleanField(default=False, help_text='Designates that this user has all permissions without explicitly assigning them.', verbose_name='superuser status')), ('id', models.AutoField(primary_key=True, serialize=False)), ('username', models.CharField(max_length=50, unique=True)), ('email', models.EmailField(max_length=100, unique=True)), ('password_hash', models.CharField(max_length=255)), ('date_of_birth', models.DateField()), ('gender', models.CharField(blank=True, max_length=10, null=True)), ('height_cm', models.IntegerField(blank=True, null=True)), ('weight_kg', models.DecimalField(blank=True, decimal_places=2, max_digits=5, null=True)), ('goal', models.CharField(blank=True, max_length=255, null=True)), ('is_admin', models.BooleanField(default=False)), ('is_staff', models.BooleanField(default=False)), ('is_active', models.BooleanField(default=True)), ('created_at', models.DateTimeField(default=django.utils.timezone.now)), ('updated_at', models.DateTimeField(auto_now=True)), ('groups', models.ManyToManyField(blank=True, help_text='The groups this user belongs to. A user will get all permissions granted to each of their groups.', related_name='user_set', related_query_name='user', to='auth.group', verbose_name='groups')), ('user_permissions', models.ManyToManyField(blank=True, help_text='Specific permissions for this user.', related_name='user_set', related_query_name='user', to='auth.permission', verbose_name='user permissions')), ], options={ 'abstract': False, }, ), ]
```

0002_alter_customuser_options.py # Generated by Django 4.2 on 2025-02-11 21:13

```
18 from django.db import migrations
```

```
class Migration(migrations.Migration):
```

```
14 dependencies = [ ('users', '0001_initial'), ]
```

```
operations = [ migrations.AlterModelOptions( name='customuser', options={ 'verbose_name': 'Пользователь', 'verbose_name_plural': 'Пользователи', }, ), ]
```

0003_alter_customuser_date_of_birth.py # Generated by Django 4.2 on 2025-02-11 21:33

```
18 from django.db import migrations, models
```

```
class Migration(migrations.Migration):
```

```
dependencies = [ ('users', '0002_alter_customuser_options'), ]
```

```
operations = [ migrations.AlterField( model_name='customuser', name='date_of_birth', field=models.DateField(blank=True, null=True), ), ]
```

0004_rename_height_cm_customuser_height_and_more.py # Generated by Django 4.2 on 2025-02-11 22:57

```
18 from django.db import migrations
```

```
class Migration(migrations.Migration):
```

```
dependencies = [ ('users', '0003_alter_customuser_date_of_birth'), ]
```

```
operations = [ migrations.RenameField( model_name='customuser', old_name='height_cm', new_name='height', ), migrations.RenameField( model_name='customuser', old_name='weight_kg', new_name='weight', ), ]
```

ПРИЛОЖЕНИЕ В ПРЕЗЕНТАЦИЯ

Рисунок 1 – Слайд №1

Рисунок 2 – Слайд №2

Рисунок 3 – Слайд №3

Рисунок 4 – Слайд №4

Рисунок 5 – Слайд №5

Рисунок 6 – Слайд №6

Рисунок 7 – Слайд №7

Рисунок 8 – Слайд №8

Рисунок 9 – Слайд №9




Рисунок 10 – Слайд №10

Рисунок 11 – Слайд №11

Рисунок 12 – Слайд №12

Рисунок 13 – Слайд №13


330

Совпадения с источником (98)		
①	https://pup-zemli.ru/sveden/common.html 	100 %
Отправленная работа федеральное государственное бюджетное образовательное учреждение высшего образования «Казанский национальный исследовательский технический университет им.		Источник федеральное государственное бюджетное образовательное учреждение высшего образования «Казанский национальный исследовательский технический университет им
②	Работа учащегося	84 %
Отправленная работа Туполева-КАИ» (КНИТУ-КАИ)		Источник Туполева (КНИТУ-КАИ)
③	http://book.lib-i.ru/25tehnicheskie/index91.php 	63 %
Отправленная работа Галеев Р.А.		Источник Изотов, Р.А
④	https://www.turboreferat.ru/tax/jelementy-nalogooblozheniya/212972-1060205-page1.html 	63 %
Отправленная работа СОДЕРЖАНИЕ ВВЕДЕНИЕ 6		Источник Содержание работы ВВЕДЕНИЕ

5	https://www.referat911.ru/Municipalnoe-pravo/tektologiya-aa-bogdanova/314268.html	65 %
Отправленная работа	Источник	
ГЛАВА 2 ПРОЕКТИРОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ 15	..15 Глава 2	
6	https://www.referat911.ru/Investicii/gosudarstvennaya-investicionnaya-politika-neftyanogo-kompleksa/486291.html	65 %
Отправленная работа	Источник	
2.1 Выбор средств реализации 16 2.1.1 Языки программирования 16	16 2.1 16 2.1	
7	Документ пользователя	65 %
Отправленная работа	Источник	
2.1.2 Среда разработки 17	2.2.1	
5	https://www.referat911.ru/Municipalnoe-pravo/tektologiya-aa-bogdanova/314268.html	65 %
Отправленная работа	Источник	
2.3 Проектирование диаграммы последовательности 21	..21 2.3	
8	Документ пользователя	66 %
Отправленная работа	Источник	
2.7 Проектирование базы данных 34	2.4 Концептуальное проектирование базы данных	
9	Документ пользователя	73 %
Отправленная работа	Источник	
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 51	СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 110	
10	Работа учащегося	100 %
Отправленная работа	Источник	
Для достижения поставленной цели необходимо решить следующие задачи:	Для достижения поставленной цели необходимо решить следующие задачи	
11	Работа учащегося	65 %
Отправленная работа	Источник	
1.1 Цели и задачи	Актуальность, цели и задачи	
12	https://www.referat911.ru/Statistika/istoilya-perepisi-naseleniya/44310.html	100 %
Отправленная работа	Источник	
Для реализации поставленной цели необходимо решить следующие задачи:	Для реализации поставленной цели необходимо решить следующие задачи	
9	Документ пользователя	66 %
Отправленная работа	Источник	
1.2 Требования к функциональности системы	1.2 Требования к программному обеспечению	

9	Документ пользователя	73 %
Отправленная работа	Источник	
ГЛАВА 2 ПРОЕКТИРОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ	Проектирование информационной системы	
7	Документ пользователя	65 %
Отправленная работа	Источник	
2.1.1 Языки программирования	1.4.2.1	
7	Документ пользователя	73 %
Отправленная работа	Источник	
2.1.2 Среда разработки	2.2.1	
9	Документ пользователя	65 %
Отправленная работа	Источник	
2.1.4 Фреймворки и библиотеки	1.4.2	
9	Документ пользователя	78 %
Отправленная работа	Источник	
<p>Диаграмма деятельности — это один из видов диаграмм UML, который описывает последовательность действий и бизнес-процессов в системе.</p>	<p>Диаграмма деятельности — это один из видов UML-диаграмм, который описывает потоки управления или последовательность действий в процессе</p>	
9	Документ пользователя	65 %
Отправленная работа	Источник	
Основные элементы диаграммы состояний:	Основные элементы диаграммы сущность-связь	
9	Документ пользователя	68 %
Отправленная работа	Источник	
Разработанная диаграмма состояний для объекта «Пользователь» представлена на рисунке 6.	Разработанная диаграмма состояний для прецедента «Регистрация сотрудника» представлена на рисунке 8	
9	Документ пользователя	68 %
Отправленная работа	Источник	
Разработанная диаграмма состояний для объекта «Одежда» представлена на рисунке 7.	Разработанная диаграмма состояний для прецедента «Регистрация сотрудника» представлена на рисунке 8	
9	Документ пользователя	69 %
Отправленная работа	Источник	
<p>На рисунке 8 представлена спроектированная диаграмма сущность-связь системы. Рисунок 8 – Диаграмма сущность-связь 2.7</p> <p>Проектирование базы данных</p>	<p>Построенная диаграмма сущность-связь представлена на рис</p> <p>Диаграмма сущность-связь Проектирование базы данных</p>	

8	Документ пользователя	79 %
Отправленная работа	Источник	
На этом этапе концептуальная модель преобразуется в логическую модель базы данных.	Концептуальная модель базы данных, созданная на предыдущем этапе, переводится в логическую модель	
9	Документ пользователя	63 %
Отправленная работа	Источник	
Нормализация базы данных.	Проектирование базы данных	
9	Документ пользователя	63 %
Отправленная работа	Источник	
Денормализация базы данных.	Проектирование базы данных	
7	Документ пользователя	65 %
Отправленная работа	Источник	
параметры подключения к базе данных;	· подключение к базе данных	
9	Документ пользователя	71 %
Отправленная работа	Источник	
проверка корректности создания нового пользователя в системе.	Заголовок Проверка корректности авторизации пользователя в системе	
9	Документ пользователя	84 %
Отправленная работа	Источник	
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 110	
13	Работа учащегося	83 %
Отправленная работа	Источник	
os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'main.settings') try:	os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'HealthCare.settings')	
14	Работа учащегося	100 %
Отправленная работа	Источник	
from django.core.management import execute_from_command_line except ImportError as exc: raise ImportError("Couldn't import Django. Are you sure it's installed and " "available on your PYTHONPATH environment variable?	from django.core.management import execute_from_command_line except ImportError as exc raise ImportError("Couldn't import Django Are you sure it's installed and " "available on your PYTHONPATH environment variable	
14	Работа учащегося	92 %
Отправленная работа	Источник	
Did you " "forget to activate a virtual environment?") from exc execute_from_command_line(sys.argv) if __name__ == '__main__':	Did you " "forget to activate a virtual environment?") from exc execute_from_command_line(sys.argv) if __name__ == "__main__"	

15	Работа учащегося	65 %
Отправленная работа	Источник	
asgiref==3.8.1	Table 8-1	
15	Работа учащегося	66 %
Отправленная работа	Источник	
djangorestframework==3.15.2	Figure 2-15	
5	https://www.referat911.ru/Municipalnoe-pravo/tektologiya-aa-bogdanova/314268.html 	63 %
Отправленная работа	Источник	
PyJWT==2.10.1	10 1.4	
15	Работа учащегося	66 %
Отправленная работа	Источник	
sqlparse==0.5.3	Figure 5-3	
16	Работа учащегося	100 %
Отправленная работа	Источник	
from django.contrib import admin	from django.contrib import admin	
17	Работа учащегося	100 %
Отправленная работа	Источник	
# Register your models here.	# Register your models here	
18	Работа учащегося	100 %
Отправленная работа	Источник	
from django.apps import AppConfig	from django.apps import AppConfig	
19	Работа учащегося	100 %
Отправленная работа	Источник	
default_auto_field = 'django.db.models.BigAutoField'	DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'	
14	Работа учащегося	71 %
Отправленная работа	Источник	
name = 'coaches'	name = 'Reporter'	
20	Работа учащегося	100 %
Отправленная работа	Источник	
from django.db import models	from django.db import models	

18	Работа учащегося	100 %
Отправленная работа	Источник	
# Create your models here.	# Create your models here	
21	Работа учащегося	64 %
Отправленная работа	Источник	
from django.test import TestCase	from django import forms	
17	Работа учащегося	70 %
Отправленная работа	Источник	
# Create your tests here.	# Create your views here	
22	Работа учащегося	100 %
Отправленная работа	Источник	
from django.urls import path	from django.urls import path	
20	Работа учащегося	67 %
Отправленная работа	Источник	
from .views import GenerateView	from .views import HomePageView	
15	Работа учащегося	97 %
Отправленная работа	Источник	
from rest_framework.permissions import IsAuthenticated from rest_framework.response import Response from rest_framework.views import APIView	from rest_framework.permissions import AllowAny, IsAuthenticated from rest_framework.response import Response from rest_framework.views import APIView	
15	Работа учащегося	86 %
Отправленная работа	Источник	
permission_classes = [IsAuthenticated] def post(self, request):	permission_classes = [AllowAny] def post(self, request)	
15	Работа учащегося	100 %
Отправленная работа	Источник	
return Response({"error":	return Response({"error"	
15	Работа учащегося	100 %
Отправленная работа	Источник	
{str(e))", status=500)	str(e)), status=500)	
23	Работа учащегося	100 %
Отправленная работа	Источник	
from django.core.asgi import get_asgi_application	from django.core.asgi import get_asgi_application	

13	Работа учащегося	87 %
Отправленная работа		Источник
os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'main.settings')		os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'HealthCare.settings')
23	Работа учащегося	100 %
Отправленная работа		Источник
application = get_asgi_application()		application = get_asgi_application()
24	Работа учащегося	100 %
Отправленная работа		Источник
from pathlib import Path BASE_DIR = Path(__file__).resolve().parent.parent		from pathlib import Path BASE_DIR = Path(__file__).resolve().parent.parent
24	Работа учащегося	98 %
Отправленная работа		Источник
INSTALLED_APPS = ['django.contrib.admin', 'django.contrib.auth', 'django.contrib.contenttypes', 'django.contrib.sessions', 'django.contrib.messages', 'django.contrib.staticfiles',		INSTALLED_APPS = ['django.contrib.admin', 'django.contrib.auth', 'django.contrib.contenttypes', 'django.contrib.sessions', 'django.contrib.messages', 'django.contrib.staticfiles', 'HealthCareApp',]
24	Работа учащегося	95 %
Отправленная работа		Источник
MIDDLEWARE = ['django.middleware.security.SecurityMiddleware', 'django.contrib.sessions.middleware.SessionMiddleware', 'django.middleware.common.CommonMiddleware', 'django.middleware.csrf.CsrfViewMiddleware', 'django.contrib.auth.middleware.AuthenticationMiddleware', 'django.contrib.messages.middleware.MessageMiddleware', 'django.middleware.clickjacking.XFrameOptionsMiddleware', 'corsheaders.middleware.CorsMiddleware',] ROOT_URLCONF = 'main.urls' TEMPLATES = [{ 'BACKEND': 'django.template.backends.django.DjangoTemplates', 'DIRS':		MIDDLEWARE = ['django.middleware.security.SecurityMiddleware', 'django.contrib.sessions.middleware.SessionMiddleware', 'django.middleware.common.CommonMiddleware', 'django.middleware.csrf.CsrfViewMiddleware', 'django.contrib.auth.middleware.AuthenticationMiddleware', 'django.contrib.messages.middleware.MessageMiddleware', 'django.middleware.clickjacking.XFrameOptionsMiddleware',] ROOT_URLCONF = 'HealthCare.urls' TEMPLATES = [{ 'BACKEND': 'django.template.backends.django.DjangoTemplates', 'DIRS':
24	Работа учащегося	100 %
Отправленная работа		Источник
[], 'APP_DIRS': True, 'OPTIONS': { 'context_processors':		[], 'APP_DIRS': True, 'OPTIONS': { 'context_processors':
19	Работа учащегося	95 %
Отправленная работа		Источник
['django.template.context_processors.debug', 'django.template.context_processors.request', 'django.contrib.auth.context_processors.auth', 'django.contrib.messages.context_processors.messages',], },], WSGI_APPLICATION = 'main.wsgi.application' DATABASES = { 'default':		['django.template.context_processors.debug', 'django.template.context_processors.request', 'django.contrib.auth.context_processors.auth', 'django.contrib.messages.context_processors.messages',], },], WSGI_APPLICATION = 'HealthCare.wsgi.application' DATABASES = { 'default':
24	Работа учащегося	82 %
Отправленная работа		Источник
'django.db.backends.postgresql', 'NAME':		'django.db.backends.sqlite3', 'NAME'

24	Работа учащегося	100 %
Отправленная работа		
Источник		
AUTH_PASSWORD_VALIDATORS = [{ 'NAME':		
AUTH_PASSWORD_VALIDATORS = [{ 'NAME'		
19	Работа учащегося	100 %
Отправленная работа		
Источник		
'django.contrib.auth.password_validation.UserAttributeSimilarityValidator', , { 'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator', }, { 'NAME':		
'django.contrib.auth.password_validation.UserAttributeSimilarityValidator', , { 'NAME' 'django.contrib.auth.password_validation.MinimumLengthValidator', }, { 'NAME'		
24	Работа учащегося	88 %
Отправленная работа		
Источник		
'django.contrib.auth.password_validation.CommonPasswordValidator', }, { 'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator', },] LANGUAGE_CODE = 'ru-ru'		
'django.contrib.auth.password_validation.CommonPasswordValidator', }, { 'NAME' 'django.contrib.auth.password_validation.NumericPasswordValidator', },] LANGUAGE_CODE = 'en-us'		
25	Работа учащегося	100 %
Отправленная работа		
Источник		
TIME_ZONE = 'UTC'		
TIME_ZONE = 'UTC'		
24	Работа учащегося	100 %
Отправленная работа		
Источник		
USE_I18N = True		
USE_I18N = True		
25	Работа учащегося	100 %
Отправленная работа		
Источник		
USE_TZ = True STATIC_URL = 'static/'		
USE_TZ = True STATIC_URL = 'static/'		
15	Работа учащегося	66 %
Отправленная работа		
Источник		
CORS_ALLOW_ALL_ORIGINS = True		
CORS_ALLOWED_ORIGINS = [
19	Работа учащегося	100 %
Отправленная работа		
Источник		
DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'		
DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'		
15	Работа учащегося	72 %
Отправленная работа		
Источник		
AUTH_USER_MODEL = 'users.CustomUser'		
AUTH_USER_MODEL = "users.CustomUser"		

16	Работа учащегося	100 %
Отправленная работа		
from django.contrib import admin from django.urls import path, include		
Источник		
from django.contrib import admin from django.urls import path, include		
15	Работа учащегося	73 %
Отправленная работа		
from rest_framework import permissions from rest_framework_simplejwt.views import TokenObtainPairView, TokenRefreshView		
Источник		
from rest_framework.permissions import AllowAny, IsAuthenticated from rest_framework.views import APIView		
13	Работа учащегося	96 %
Отправленная работа		
from django.core.wsgi import get_wsgi_application os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'main.settings') application = get_wsgi_application()		
Источник		
from django.core.wsgi import get_wsgi_application os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'HealthCare.settings') application = get_wsgi_application()		
16	Работа учащегося	100 %
Отправленная работа		
from django.contrib import admin		
Источник		
from django.contrib import admin		
18	Работа учащегося	100 %
Отправленная работа		
from django.apps import AppConfig		
Источник		
from django.apps import AppConfig		
15	Работа учащегося	100 %
Отправленная работа		
class UsersConfig(AppConfig):		
Источник		
class UsersConfig(AppConfig)		
19	Работа учащегося	100 %
Отправленная работа		
default_auto_field = 'django.db.models.BigAutoField'		
Источник		
DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'		
14	Работа учащегося	71 %
Отправленная работа		
name = 'users'		
Источник		
name = 'Reporter'		
17	Работа учащегося	73 %
Отправленная работа		
from django.contrib.auth.models import AbstractBaseUser, BaseUserManager, PermissionsMixin		
Источник		
from django.contrib.auth.models import User		

<p>20 Работа учащегося</p> <p>100 %</p> <p>Отправленная работа</p> <p>from django.db import models</p>	<p>Источник</p> <p>from django.db import models</p>
<p>15 Работа учащегося</p> <p>100 %</p> <p>Отправленная работа</p> <p>USERNAME_FIELD = 'username'</p>	<p>Источник</p> <p>USERNAME_FIELD = 'username'</p>
<p>22 Работа учащегося</p> <p>100 %</p> <p>Отправленная работа</p> <p>def __str__(self):</p>	<p>Источник</p> <p>def str (self)</p>
<p>15 Работа учащегося</p> <p>100 %</p> <p>Отправленная работа</p> <p>from rest_framework import serializers</p>	<p>Источник</p> <p>from rest_framework import serializers</p>
<p>20 Работа учащегося</p> <p>69 %</p> <p>Отправленная работа</p> <p>from .models import CustomUser</p>	<p>Источник</p> <p>from .models import Post</p>
<p>15 Работа учащегося</p> <p>100 %</p> <p>Отправленная работа</p> <p>class RegisterSerializer(serializers.ModelSerializer): password = serializers.CharField(write_only=True)</p>	<p>Источник</p> <p>class RegisterSerializer(serializers.ModelSerializer) password = serializers.CharField(write_only=True)</p>
<p>15 Работа учащегося</p> <p>66 %</p> <p>Отправленная работа</p> <p>model = CustomUser fields = ('username', 'email', 'password',) def validate_password(self, value):</p>	<p>Источник</p> <p>fields = ['username', 'password', 'email', 'employee_id', def validate(self, data)</p>
<p>15 Работа учащегося</p> <p>100 %</p> <p>Отправленная работа</p> <p>def create(self, validated_data):</p>	<p>Источник</p> <p>def create(self, validated_data)</p>
<p>15 Работа учащегося</p> <p>100 %</p> <p>Отправленная работа</p> <p>class UserSerializer(serializers.ModelSerializer):</p>	<p>Источник</p> <p>class UserSerializer(serializers.ModelSerializer)</p>
<p>21 Работа учащегося</p> <p>64 %</p> <p>Отправленная работа</p> <p>from django.test import TestCase</p>	<p>Источник</p> <p>from django import forms</p>

17	Работа учащегося	70 %
Отправленная работа	Источник	
# Create your tests here.	# Create your views here	
22	Работа учащегося	100 %
Отправленная работа	Источник	
from django.urls import path	from django.urls import path	
15	Работа учащегося	87 %
Отправленная работа	Источник	
from rest_framework import generics from rest_framework.permissions import AllowAny from rest_framework.permissions import IsAuthenticated from .serializers import RegisterSerializer, UserSerializer	from rest_framework import generics, status from rest_framework.permissions import AllowAny, IsAuthenticated from rest_framework.permissions import AllowAny, IsAuthenticated from .serializers import RegisterSerializer, LoginSerializer	
15	Работа учащегося	83 %
Отправленная работа	Источник	
class RegisterView(generics.CreateAPIView): serializer_class = RegisterSerializer permission_classes = [AllowAny]	class RegisterView(generics.CreateAPIView) permission_classes = [AllowAny]	
18	Работа учащегося	100 %
Отправленная работа	Источник	
from django.db import migrations, models	from django.db import migrations, models	
18	Работа учащегося	100 %
Отправленная работа	Источник	
class Migration(migrations.Migration): initial = True	class Migration(migrations.Migration) initial = True	
18	Работа учащегося	95 %
Отправленная работа	Источник	
from django.db import migrations class Migration(migrations.Migration):	from django.db import migrations, models class Migration(migrations.Migration)	
14	Работа учащегося	78 %
Отправленная работа	Источник	
dependencies = [('users', '0001_initial'),]	dependencies = [('Reporter', '0001_initial'),]	
18	Работа учащегося	100 %
Отправленная работа	Источник	
from django.db import migrations, models class Migration(migrations.Migration):	from django.db import migrations, models class Migration(migrations.Migration)	

Отправленная работа

Источник

```
from django.db import migrations class Migration(migrations.Migration):
```

```
from django.db import migrations, models class  
Migration(migrations.Migration)
```