СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Приложение под названием GymTracker: Умное напоминание и анализ тренировок разрабатывается как мобильный инструмент для поддержки регулярных занятий спортом и отслеживания силового прогресса. В условиях современного ритма жизни пользователи часто теряют мотивацию или забывают о тренировках, а также не имеют наглядной обратной связи по результатам. GymTracker решает эту проблему, предоставляя удобный интерфейс для планирования тренировок, автоматические напоминания, фиксацию данных о выполненных упражнениях и еженедельную аналитику. Приложение будет отправлять уведомления в заданное время, отслеживать выполнение тренировок, собирать информацию о весах, повторениях и самочувствии, а также формировать отчёт с динамикой силовых показателей и поздравлением, если неделя закрыта успешно. При этом продукт не будет выполнять функции персонального тренера, не будет синхронизироваться с внешними устройствами (фитнес-браслеты, часы) и не будет включать социальные функции или хранить медицинские данные. GymTracker — это персональный помощник, ориентированный на дисциплину, результат и удобство, без лишней теории и перегрузки.

1 ТРЕБОВАНИЯ ПОЛЬЗОВАТЕЛЯ

В рамках данного проекта ставится важная задача разработки специализированного мобильного приложения, предназначенного для планирования и отслеживания тренировочного процесса. Современные пользователи всё чаще обращаются к цифровым инструментам для контроля физической активности, постановки целей и анализа прогресса. Наличие удобного и функционального приложения позволяет систематизировать тренировки, отслеживать результаты и повышать мотивацию, что играет ключевую роль в достижении спортивных целей.

При создании подобного приложения крайне важно разработать удобный и интуитивно понятный интерфейс, который обеспечит быстрый доступ к основным функциям: выбору тренировочной программы, добавлению собственных упражнений, ведению истории занятий и контролю прогресса. В ходе функционирования программы предполагается работа с данными, вводимыми пользователем, а также их структурированное отображение в виде списков, карточек и графиков. Это позволит получить всесторонний анализ тренировочного процесса и повысить эффективность занятий.

Разрабатываемое приложение будет охватывать широкий спектр данных, предоставляя развернутую информацию по следующим ключевым категориям:

- Профиль пользователя — имя, цель тренировок, уровень подготовки, язык интерфейса, аватар.
- Программы тренировок — готовые планы по целям и уровням (похудение, набор массы, сила, выносливость).
- История — список выполненных тренировок с датами, упражнениями и результатами.
- Прогресс — графики изменения веса, самочувствия и силовых показателей.
- Настройки — смена имени, аватарки, языка, выход из профиля.

Для успешной реализации проекта был изучен ряд фундаментальных источников, предоставляющих необходимую теоретическую и практическую основу.

- Джош Кларк – “Designing Mobile Interfaces”. В книге подробно рассматриваются принципы проектирования интерфейсов для мобильных приложений, включая удобство навигации, визуальную иерархию и адаптацию под разные устройства.
- Стив Круг – “Не заставляйте меня думать”. Классическое руководство по юзабилити, в котором акцент сделан на простоте и интуитивности интерфейсов, что особенно важно для приложений, ориентированных на широкий круг пользователей.
- Джонатан Роуз – “Fitness App Development Guide”. Практическое пособие по созданию приложений для фитнеса, включающее примеры

структурирования данных о тренировках, построения графиков прогресса и внедрения мотивационных элементов.

Все эти источники сыграли ключевую роль в формировании концепции проекта. Они позволили глубже разобраться в тонкостях проектирования мобильных интерфейсов, определить основные направления разработки, выбрать оптимальные методы реализации, а также заложить надежную теоретическую и практическую базу для успешного выполнения поставленной задачи.

1.1 Постановка задачи

В рамках данного проекта необходимо разработать мобильное приложение, предназначенное для планирования и отслеживания тренировок. В современных условиях пользователю важно иметь доступ к инструменту, который позволит фиксировать результаты занятий, анализировать прогресс и корректировать программу тренировок. Задача приложения — предоставить удобный интерфейс для регистрации и настройки профиля, выбора готовой программы или добавления собственной тренировки, ведения истории занятий и отображения прогресса в виде графиков и статистики.

Программа должна реализовывать сбор и отображение информации по следующим категориям: профиль (имя, цель, уровень, язык, аватар), тренировки (тип — силовая, кардио, растяжка, упражнения, параметры — вес, повторы, подходы, длительность), история (список выполненных занятий с результатами), прогресс (динамика веса, самочувствия, силовых показателей), а также настройки (смена имени, аватарки, языка, выход из профиля).

1.2 Программные интерфейсы

Для реализации мобильного приложения, предназначенного для планирования и отслеживания тренировочного процесса, необходимо рассмотреть современные методы и алгоритмы, применяемые в области разработки пользовательских интерфейсов, обработки данных и визуализации прогресса. Основная задача заключается в том, чтобы обеспечить простоту использования, гибкость и адаптивность приложения при минимальной нагрузке на ресурсы устройства.

Одним из ключевых методов является использование модульной архитектуры приложения, при которой каждая функциональная часть (регистрация, добавление тренировок, история, прогресс, настройки) реализуется как отдельный модуль. Такой подход упрощает масштабирование и дальнейшее развитие проекта, а также позволяет изолировать возможные ошибки в отдельных компонентах.

Для хранения и обработки данных целесообразно применять реляционные базы данных (например, SQLite), встроенные в мобильные платформы.

Это обеспечивает быстрый доступ к информации о тренировках, истории и профиле пользователя. Алгоритмы выборки и фильтрации данных

позволяют эффективно отображать только необходимую информацию, например, тренировки за определённый период или динамику изменения веса.

Важным направлением является использование алгоритмов визуализации данных, которые позволяют преобразовывать числовые показатели в наглядные графики и диаграммы. Для отображения прогресса могут применяться линейные графики (динамика веса и силовых показателей), столбчатые диаграммы (количество выполненных тренировок по неделям) и индикаторы выполнения целей. Такие методы повышают наглядность и мотивацию пользователя.

При проектировании интерфейса применяются методы UX/UI-дизайна, основанные на принципах минимализма и интуитивности. Алгоритмы навигации строятся по схеме «главный экран — раздел — действие», что снижает когнитивную нагрузку на пользователя. Для выбора упражнений и программ используются выпадающие списки и карточки, что соответствует современным практикам мобильного дизайна.

Для обеспечения персонализации могут использоваться простые алгоритмы рекомендаций, основанные на выбранной цели и уровне подготовки. Например, при выборе цели «похудение» приложение автоматически предлагает кардио-программы и упражнения с высокой интенсивностью, а при выборе цели «сила» — силовые комплексы с прогрессией нагрузки. Такие алгоритмы не требуют сложных вычислений, но значительно повышают удобство использования.

Таким образом, решение поставленной задачи базируется на сочетании модульной архитектуры, встроенных средств хранения данных, алгоритмов визуализации и методов UX/UI-дизайна. Такой подход позволяет создать приложение, которое будет одновременно простым в использовании, функциональным и адаптированным под широкий круг пользователей.

1.3 Выбор инструментов для реализации утилиты

Для успешной реализации мобильного приложения, предназначенного для планирования и отслеживания тренировочного процесса, необходимо тщательно подобрать инструменты разработки, обеспечивающие надёжность, удобство и расширяемость проекта. Выбор инструментов определяется как функциональными требованиями, так и особенностями целевой платформы.

В качестве основной среды разработки целесообразно использовать Android Studio и Xcode, что позволит обеспечить кроссплатформенную поддержку Android и iOS. Для унификации кода и ускорения разработки может быть применён фреймворк Flutter (язык Dart) или React Native (JavaScript/TypeScript). Эти технологии позволяют создавать единый код приложения с последующей компиляцией под разные мобильные платформы, что значительно сокращает время и ресурсы на разработку.

Для хранения данных о тренировках, истории и профиле пользователя оптимальным решением является использование встроенной базы данных

SQLite, которая обеспечивает лёгкость интеграции, высокую скорость работы и автономность. В случаях, когда требуется синхронизация данных между устройствами, может быть подключено облачное хранилище, например Firebase Realtime Database или Firestore.

При проектировании интерфейса рекомендуется использовать инструменты прототипирования и дизайна, такие как Figma или Adobe XD, которые позволяют заранее создать макеты экранов, протестировать навигацию и согласовать визуальный стиль приложения. Это обеспечивает удобство в разработке и снижает вероятность ошибок на этапе реализации.

Для управления проектом и контроля версий кода будет применяться система Git с размещением репозитория на платформе GitHub или GitLab. Это позволит организовать командную работу, отслеживать изменения и обеспечивать надёжное хранение исходного кода.

1.4. Диаграммы взаимодействия с системой

Процесс авторизации обеспечивает доступ к персонализированным данным пользователя, включая историю тренировок, цели, прогресс и настройки. После выбора существующей учётной записи и подтверждения операции входа, система инициирует загрузку пользовательского профиля и связанных данных, как представлено на рисунке 1.1.

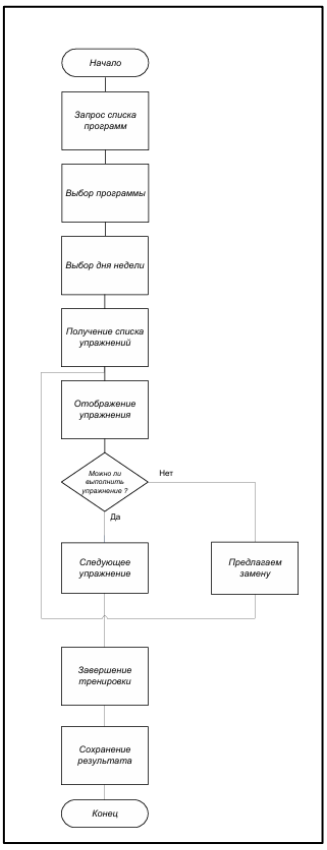

Рисунок 1.1 – Диаграмма запуска программы

Диаграмма классов отображает статическую структуру системы, включая основные сущности приложения, их атрибуты, методы и взаимосвязи, представлена на рисунке 1.2.

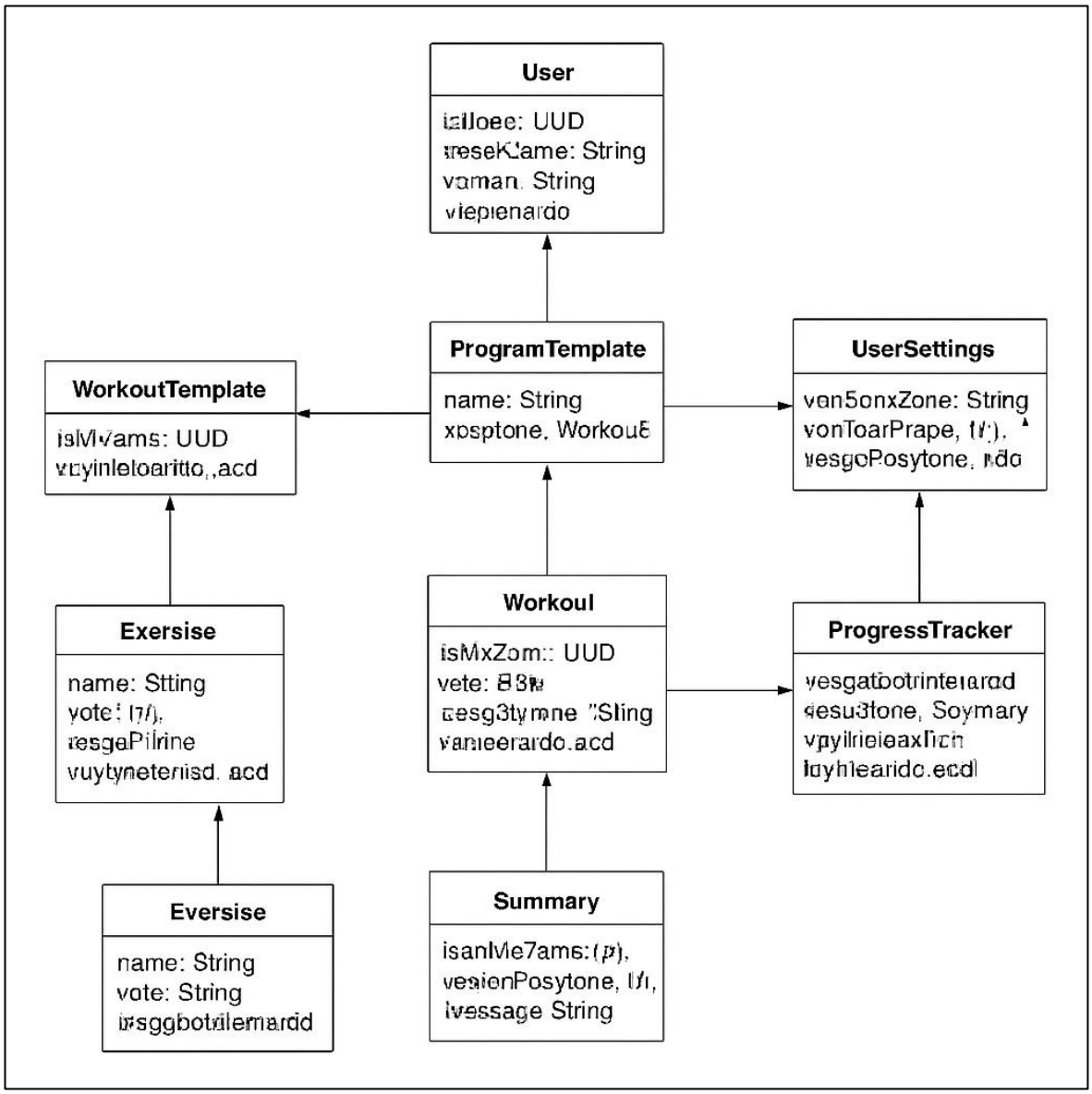

Рисунок 1.2 – Диаграмма классов программы

Диаграмма отражает поведенческие сценарии взаимодействия пользователя с функциональными компонентами мобильного приложения, предназначенного для планирования, выполнения и анализа тренировочного процесса. Центральным элементом схемы является актор «Пользователь», от которого исходят ветви, соответствующие основным группам действий. Данная диаграмма представлена на рисунке 1.3:

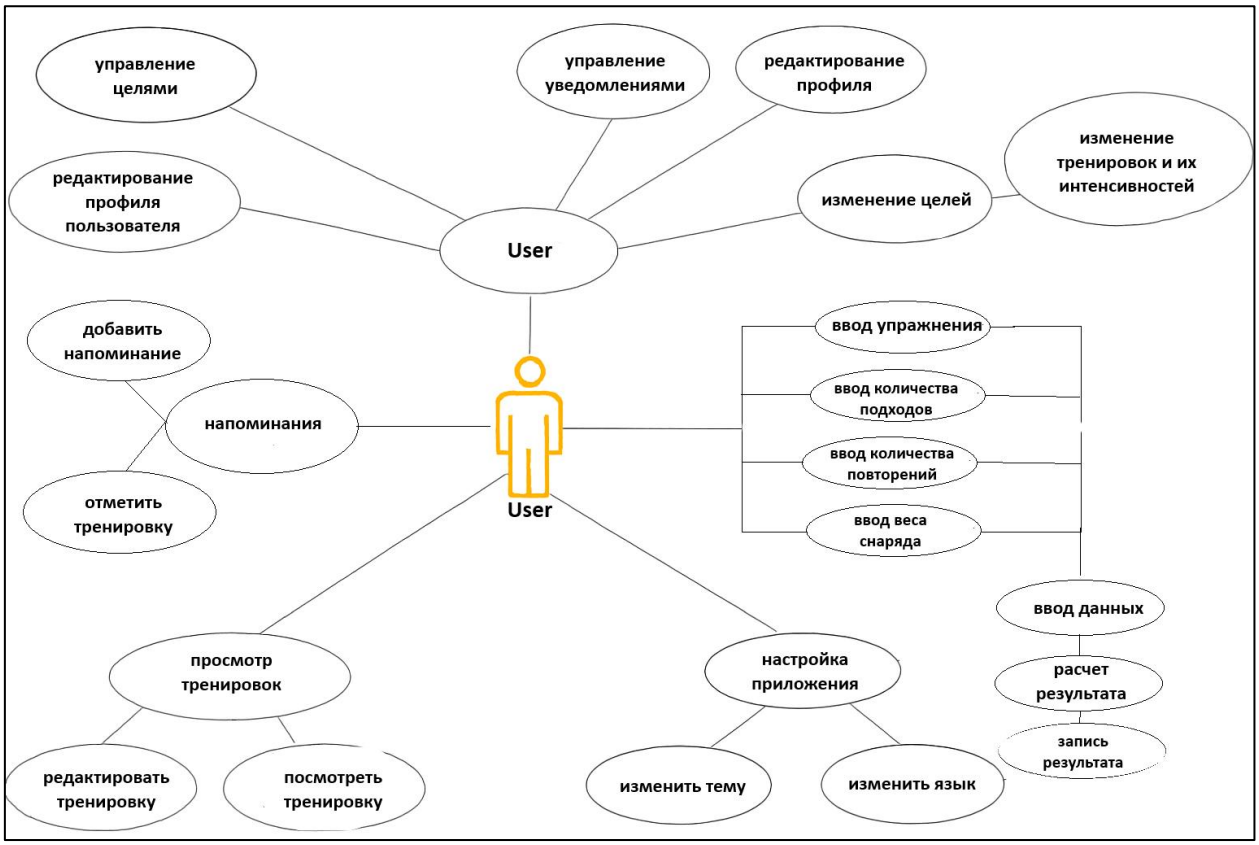

Рисунок 1.3 – Диаграмма взаимодействия пользователя с приложением

Голосарий представлен в таблице 1.1

Таблица 1.1 - Голосарий

Термин (Класс/Атрибут/Метод)	**Тип**	**Определение и Назначение**
`User`	Класс	Представляет зарегистрированного пользователя, ведущего тренировочный процесс.
`userID`	Атрибут	Уникальный идентификатор пользователя, присваивается при регистрации.
`Workout`	Класс	Сессия тренировки, включающая один или несколько упражнений.
`exerciseList`	Атрибут	Массив упражнений, входящих в тренировку.

Продолжение таблицы 1.1

Exercise	Класс	Отдельное упражнение с параметрами: название, подходы, повторы, вес.
sets	Атрибут	Количество подходов в упражнении.
Reps	Атрибут	Количество повторов в одном подходе.
weight	Атрибут	Вес, используемый в упражнении (если применимо).
startWorkout()	Метод	Инициализирует тренировку, фиксирует дату и выбранные упражнения.
completeWorkout()	Метод	Завершает тренировку, сохраняет прогресс и отмечает выполненные упражнения.
skipExercise()	Метод	Помечает упражнение как пропущенное, добавляет рекомендацию по замене.
ProgressTracker	Класс	Отслеживает прогресс пользователя по неделям, группам мышц и достижениям.
achievementList	Атрибут	Список достижений пользователя (например, «3 тренировки подряд»).
generateWeeklySummary()	Метод	Формирует отчёт по неделе: количество тренировок, пропуски, прогресс.
ProgramTemplate	Класс	Шаблон тренировочной программы с разбивкой по дням и группам мышц.
dayPlan	Атрибут	Структура тренировок по дням недели

Продолжение таблицы 1.1

`selectDayWorkout()`	Метод	Загружает тренировку, соответствующую выбранному дню недели.
`UserSettings`	Класс	Настройки пользователя: имя, аватар, цель, уровень.
`updateProfile()`	Метод	Обновляет имя и аватар пользователя, без зависаний интерфейса.

2 ТРЕБОВАНИЯ ПОЛЬЗОВАТЕЛЯ

2.1 Программные интерфейсы

Приложение GymTracker будет взаимодействовать с рядом внешних библиотек и сервисов, необходимых для реализации функциональности напоминаний, хранения данных, визуализации и построения пользовательского интерфейса. Ниже перечислены основные программные интерфейсы:

- Flask / FastAPI — фреймворк для создания REST API, обеспечивающий взаимодействие между мобильным интерфейсом и логикой приложения
- SQLite — встроенная база данных для хранения информации о тренировках, расписании и статистике
- SQLAlchemy — ORM-библиотека для удобной работы с базой данных
- Plyer — библиотека для отправки локальных уведомлений на мобильном устройстве
- Schedule — библиотека для запуска задач по расписанию (например, проверка выполнения тренировок в конце недели)
- Matplotlib / Plotly — библиотеки для построения графиков и визуализации прогресса пользователя
- Kivy / KivyMD — фреймворки для создания мобильного интерфейса на Python с поддержкой Material Design
- Firebase Cloud Messaging (опционально) — сервис для реализации push-уведомлений, если потребуется облачная доставка
- DB Browser for SQLite (внешний инструмент) — GUI-программа для визуального управления базой данных во время разработки
- Postman / REST Client (внешние инструменты) — средства для тестирования API и отладки взаимодействия между компонентами

2.2 Интерфейс пользователя

Пользователь взаимодействует с приложением GymTracker через мобильный интерфейс, построенный на основе Kivy/KivyMD. Основной сценарий работы включает планирование тренировок, получение напоминаний, фиксацию результатов и получение еженедельной статистики.

Пользователь может:

- Задать расписание тренировок на неделю
- Получать уведомления в заданное время
- Вводить данные о выполненной тренировке (тип, упражнения, вес, повторения, самочувствие)
- Просматривать статистику недели и сравнение с предыдущими
- Получать поздравление при успешном выполнении всех тренировок

Система реагирует на действия пользователя мгновенно, сохраняя данные локально и обновляя аналитику. Взаимодействие построено по принципу «минимум кликов — максимум результата».

Пользовательский интерфейс спроектирован с акцентом на интуитивную понятность и минималистичный дизайн, что обеспечивает комфортное взаимодействие даже для неподготовленных пользователей. Навигационная структура приложения включает последовательность логически связанных экранов:

Стартовый экран предлагает три основных варианта действий: выбор языка для приложения, данный экран представлен на рисунке 2.1:

Рисунок 2.1 – Стартовый экран

Экран регистрации включает в себя 3 этапа : Ввод имени, выбор уровня подготовки, выбор цели тренировок, ввод веса пользователя и выбор дней недели для тренировок. Данные шаги представлены на рисунках 2.2 - 2.6:

Рисунок 2.2 – Ввод имени пользователя

Рисунок 2.3 – Ввод уровня физической подготовки пользователя

Рисунок 2.4 – Ввод цели, которую хочет достичь пользователь

Рисунок 2.5 – Ввод веса пользователя

Рисунок 2.6 – Ввод удобных дней тренировок для пользователя

После успешного пользователем данных появится окно с его данными, данный экран представлен на рисунке 2.7:

Рисунок 2.7 – Окно успешной регистрации

2.3 Характеристики пользователей

Приложение GymTracker ориентировано на индивидуальных пользователей, занимающихся спортом самостоятельно. Ниже представлены основные группы пользователей и их характеристики:

1) Новичок

Это те, кто только начинает заниматься фитнесом — без опыта или с минимальной практикой.

Что им нужно:

- Напоминания о тренировках
- Мотивация через визуальные элементы
- Простая фиксация прогресса без лишних шагов

Что они ждут от приложения:

- Минимум кликов
- Понятный интерфейс
- Визуальная статистика, которая сразу показывает результат

2) Средний уровень

Это те, кто регулярно тренирующиеся, понимают базовую структуру тренировок, хотят расти, они умеют уверенно работать с приложениями, разбираются в типах нагрузок и цикличности.

Что им нужно:

- Отслеживание силовых показателей
- Сравнение прогресса по неделям
- Контроль регулярности занятий

Что они ждут от приложения:

- Точная аналитика
- Гибкое расписание
- Надёжная система напоминаний

3) Продвинутые

Это уверенные пользователи, которые хотят контролировать тренировочный процесс и данные.

Что им нужно:

- Прозрачная структура
- Экспорт данных
- Кастомизация интерфейса и расписания Что они ждут от приложения:
- Стабильность
- Расширяемость
- Без лишней теории и «воды»

2.4 Предложения и зависимости

1. Модель данных и управление

1) Первичный источник данных: Основой для формирования аналитики и рекомендаций служат данные, вручную вводимые пользователем (параметры тренировок, вес, цели, субъективные оценки).

2) Синхронизация: Функция синхронизации данных между устройствами и их облачного хранения требует стабильного интернет-соединения на стороне клиента.

3) Интеграция со сторонними сервисами: В будущих версиях планируется интеграция с платформами Google Fit и Apple Health для автоматического сбора данных. Реализация зависит от:

4) Доступности и стабильности соответствующих API.

5) Явного согласия пользователя на обмен данными.

2. Функциональные требования

1) Пользовательский интерфейс: Реализовать два адаптивных режима отображения интерфейса («Простой» и «Расширенный») с возможностью переключения в настройках профиля.

2) Первоначальное знакомство: Внедрить интерактивный онбординг с сохранением прогресса и возможностью его повторного прохождения.

3) Экспорт и планирование: Обеспечить функционал экспорта отчётов в форматы PDF и CSV, а также интеграцию с системными календарями для планирования тренировок.

4) Управление функционалом: Использовать механизм фич-флагов для контроля за выпуском новых функций, что позволит осуществлять их постепенное внедрение и быстрый откат.

3. Технические требования и зависимости

1) Серверная часть: Требуется развертывание серверной инфраструктуры для облачного хранения данных, аутентификации пользователей, сервиса синхронизации и генерации аналитики.

2) Хранение данных: На начальном этапе используется локальная БД (SQLite). При росте числа пользователей необходима миграция на масштабируемую облачную базу данных (например, PostgreSQL или Firebase).

3) Внешние интеграции: Необходима разработка модулей для работы с API сторонних сервисов (Google Fit, Apple Health, календари), а также модулей для экспорта данных и резервного копирования.

4) Клиентская часть: Приложение должно включать:

5) Адаптивный UI с поддержкой динамических шрифтов.

6) Локальное кэширование данных (включая состояние онбординга).

7) Настройку отображения системных подсказок.

4. Нефункциональные требования

1) Производительность: Интерфейс и логика должны быть оптимизированы для работы на устройствах с ограниченными ресурсами (оперативная память, процессор).

2) Надёжность и отказоустойчивость: Необходимо внедрить систему логирования, мониторинга ключевых метрик, регулярного резервного копирования данных и стратегию отката обновлений.

3) Масштабируемость: Архитектура серверной части должна позволять горизонтальное масштабирование и балансировку нагрузки.

5. Безопасность и соответствие

1) Защита данных: Все персональные и медицинские данные пользователей должны шифроваться (как при передаче, так и при хранении) в соответствии с применимым законодательством (например, ФЗ-152).

2) Качество аналитики: Точность предоставляемых приложением расчётов и рекомендаций напрямую зависит от полноты и корректности данных, введённых пользователем.

6. Управление изменениями и рисками

1) Внешние факторы: Изменения в политике платформ (iOS/Android) или в законодательстве могут потребовать доработки механизмов уведомлений, доступа к хранилищу и обновления используемых библиотек.

2) Масштабирование: Рост пользовательской базы потребует плановой миграции данных и рефакторинга архитектуры систем хранения и синхронизации.

3) Процесс выпуска обновлений: Для каждой новой функции должна составляться матрица зависимостей (внешние API, библиотеки, нагрузка). Выпуск критических изменений должен сопровождаться комплексом тестов (модульные, интеграционные, нагрузочные) и поэтапным rollout с использованием фич-флагов.

3 СИСТЕМНЫЕ ТРЕБОВАНИЯ

3.1 Функциональные требования

Приложение должно обеспечивать персонализированный опыт через профиль пользователя с настройками целей, уровня и предпочтений, поддерживать переключение режимов отображения «Простой» и «Расширенный», а также реализовывать механизм фич-флагов для поэтапного включения и быстрой деактивации новых функций. Все действия пользователя должны быть атомарными и откатываемыми; критичные изменения сохраняются с возможностью отмены в течение короткого окна.

Регистрация и авторизация реализуются через приложение. Профиль должен позволять изменять имя, аватар, цель, уровень, параметры отображения и уведомлений; поддерживается мультипрофильность на одном устройстве и хранение состояния онбординга и настроек в профиле как локально, так и в облаке.

Приложение должно позволять создание, редактирование и удаление программ тренировок с набором тренировок и упражнений, предоставлять библиотеку готовых программ с метаданными (длительность, частота, уровень, описание) и обеспечивать создание пользовательских тренировок (выбор упражнений, количество подходов, повторений, отдых, вес, заметки). Планировщик обязан поддерживать назначение тренировок на календарь, повторяющиеся события и экспорт в системный календарь; необходима поддержка шаблонов тренировок и возможность их клонирования и модификации.

Система трекинга должна фиксировать выполненные тренировки с временными метками, детализацией упражнений и результатами (вес, повторения, подходы, время), а также отслеживать дополнительные параметры — вес, самочувствие, настроение, сон. Дашборд прогресса должен отображать ключевые метрики: количество тренировок, суммарное время, серия дней подряд, выполнение плана. Система аналитики должна предоставлять отчёты: сравнение по неделям, динамику силовых показателей, распределение типов тренировок, с возможностью фильтрации по датам и типам данных и экспортом в PDF и CSV.

Уведомления реализуются через push и локальные напоминания о запланированных тренировках, пропущенных занятиях, достижениях и целях; в профиле должна быть гибкая настройка типов уведомлений и расписание «тихого режима». Необходима поддержка групповых тренировок и приглашений: создание события, приглашение участников и подтверждение участия.

Интеграции с внешними сервисами по API (например, Google Fit, Apple Health) реализуются при согласии пользователя для чтения и записи данных. Синхронизация данных между устройствами осуществляется через облачный backend с авторизацией; предусмотрен офлайн-режим с локальным кешированием и очередью синхронизации, а также механизм миграции данных при переходе от локального хранилища к облачной базе. Для надёжности

реализуются бэкапы и восстановление данных, журналирование операций и мониторинг ошибок.

Безопасность предусматривает обязательное шифрование при передаче и хранении персональных данных, разграничение прав доступа и использование протоколов аутентификации для серверных API в соответствии с применимыми нормативами о защите данных. Точность аналитики и релевантность рекомендаций зависят от полноты и корректности исходных данных, вводимых пользователями.

3.2 Нефункциональные требования

3.2.1 Атрибуты качества

Производительность: приложение должно быть быстрым и отзывчивым; время загрузки основных экранов (главный дашборд, экран тренировки, журнал тренировок) не должно превышать 2 секунд с момента запроса пользователя; основные интерактивные операции (запись тренировки, сохранение результата, открытие карточки упражнения) — отклик интерфейса ≤ 200 мс на типичных устройствах и ≤ 500 мс на устройствах с ограниченными ресурсами; пакетные фоновые синхронизации не должны блокировать UI и выполняются не реже чем раз в 5 минут.

Надёжность: система гарантирует сохранность всех введённых пользователем данных; для критичных операций реализована атомарная запись, транзакционная целостность и журнал изменений; при ошибках или сбоях данные не теряются — включены локальное автосохранение и регулярные бэкапы на сервере; предусмотрен механизм восстановления и проверяемая процедура отката; RTO для основных сервисов ≤ 2 часа, RPO ≤ 15 минут.

Безопасность: все конфиденциальные данные (аккаунты, показатели здоровья, персональные настройки, платёжная информация при наличии) должны храниться и передаваться в зашифрованном виде; все сетевые соединения через HTTPS/TLS 1.2+; хранение данных в покое — шифрование AES-256 или эквивалент; серверные API защищены через RBAC и авторизацию OAuth2/JWT; хранение секретов и ключей — в защищённом менеджере секретов; предусмотрены механизмы защиты от брутфорса и опциональная двухфакторная аутентификация.

Масштабируемость: архитектура `backend` рассчитана на горизонтальное масштабирование; решение должно выдерживать до 100 000 активных пользователей без потери производительности с возможностью дальнейшего роста; база данных и файловое хранилище проектируются с возможностью миграции от локального SQLite к распределённым решениям (PostgreSQL, объектное хранилище) по мере роста нагрузки.

Удобство использования: ввод ключевых данных (запись тренировки, отметка выполненного занятия) занимает не более 3 тапов; основные действия доступны с главного экрана; тексты кратки и ориентированы на действие; предусмотрен режим «Простой» для быстрого доступа и режим

«Расширенный» для продвинутых пользователей; онбординг пошаговый, с возможностью повторного запуска и отключения подсказок.

Совместимость и адаптивность: приложение поддерживает текущие мажорные версии мобильных ОС; интерфейс адаптируется к разным размерам экранов и плотности пикселей; предусмотрена graceful degradation на устаревших платформах; ресурсоёмкие визуальные элементы адаптируются под возможности устройства.

Мониторинг и эксплуатация: централизованный сбор логов и метрик с алертами по ключевым показателям (ошибки 5xx, рост латентности, заполнение диска); хранение логов минимум 90 дней с возможностью выборки по пользователю или транзакции; автоматизированные бэкапы, тестирование восстановления и регулярные проверочные процедуры.

Качество релизов: CI/CD-пайплайн с автоматическими тестами (unit, integration, e2e), поэтапным развёртыванием и использованием фич-флагов; обязательная тестовая среда, зеркально воспроизводящая продакшен; перед релизом критичных изменений проводится набор тестов и staged rollout.

Экономические и операционные ограничения: решения по инфраструктуре и хранению данных принимаются с учётом прогнозируемых затрат и плана оптимизации при росте базы; SLA для критичных инцидентов и регламент обработки инцидентов обеспечиваются службой поддержки.

4 ТЕСТЫ ДЛЯ ПРИЛОЖЕНИЯ GYMTRACKER ULTIMATE

4.1. Юнит-тесты

- Проверка валидации ввода:
- Вес: число от 1 до 200 кг
- Повторения: целое число от 1 до 50
- Сеты: целое число от 1 до 10
- Проверка расчёта прогресса (корректность формул).
- Проверка экспорта данных в XML (структура, кодировка).
- Проверка сохранения и загрузки истории тренировок.
- Проверка обработки ошибок при пустых или некорректных данных.

4.2. Интеграционные тесты

- Добавление упражнения → сохранение в базе → отображение в истории.
- Запуск тренировки → получение списка упражнений → выполнение → сохранение.
- Экспорт отчёта → скачивание файла → повторное открытие.
- Синхронизация с облаком (если предусмотрено).

4.3. UI/UX тесты

- Корректное отображение форм и кнопок на разных устройствах.
- Понятные сообщения об ошибках.
- Возможность вернуться назад без потери данных.
- Проверка тёмной/светлой темы.
- Проверка локализации (русский/английский).

4.4. Функциональные тесты

- Регистрация и вход.
- Настройка профиля.
- Выбор программы и дня.
- Запуск тренировки.
- Выполнение упражнения (корректные и некорректные данные).
- Пропуск или замена упражнения.
- Завершение тренировки.
- Просмотр истории.
- Экспорт отчёта.

4.5. Нагрузочные тесты

- Работа с историей из 1000+ записей.
- Экспорт больших объёмов данных.
- Одновременные действия (ввод данных + экспорт).

4.6. Тесты безопасности

- Проверка SQL-инъекций (если используется база данных).
- Проверка XSS в текстовых полях.
- Шифрование паролей.
- Защита от повторной отправки форм.

4.7. End-to-End тесты

1. Регистрация пользователя.
2. Настройка профиля.
3. Запуск тренировки.
4. Ввод упражнения (с ошибкой и исправлением).
5. Завершение тренировки.
6. Просмотр истории.
7. Экспорт отчёта.

4.8. Регрессионные тесты

- Проверка, что после добавления новой функции (например, «замена упражнения») старые сценарии работают корректно.

4.9. Тесты удобства (Usability)

- Понятность сообщений об ошибках.
- Логичность расположения кнопок.
- Минимальное количество действий для запуска тренировки.

5 СЦЕНАРИИ ВЗАИМОДЕЙСТВИЯ

5.1 Запуск программы

Пользователь авторизован и выбрал программу тренировок. Основной поток:

1. Пользователь открывает приложение.
2. Выбирает программу и день недели.
3. Приложение загружает список упражнений.
4. Первое упражнение отображается на экране.
5. Пользователь начинает выполнение.

Тренировка запущена, данные готовы к вводу.

5.2 Выполнение упражнения

Тренировка запущена, упражнение отображено. Основной поток:

1. Пользователь вводит вес, количество повторений и сетов.
2. Приложение проверяет корректность данных.
3. Если данные корректны — сохраняет результат.
4. Если данные некорректны — выводит ошибку и рекомендации по весам.

После выполнения упражнение отмечено как выполненное.

5.3 Замена упражнения

Если пользователь не может выполнить текущее упражнение.
Основной поток:

1. Пользователь выбирает «Заменить упражнение».
2. Приложение предлагает список альтернатив.
3. Пользователь выбирает замену.
4. Приложение отображает новое упражнение.

Упражнение заменено, тренировка продолжается.

5.4 Завершение тренировки

Все упражнения просмотрены. Основной поток:

1. Пользователь нажимает «Завершить тренировку».
2. Приложение сохраняет результаты.
3. Отображает уведомление об успешном завершении.

Тренировка сохранена в истории.

5.5 Просмотр истории тренировок

Есть завершённые тренировки. Основной поток:

1. Пользователь открывает раздел «История».

2. Выбирает тренировку для просмотра.
3. Приложение отображает детали (упражнения, веса, прогресс).
4. Пользователь выбирает «Экспорт в XML».
5. Приложение формирует и предлагает скачать файл.
История просмотрена, отчёт сохранён.

5.6 Сценарий использования

1. Start Workout Пользователь запускает тренировку из главного экрана приложения.
2. Choose Workout Program Из списка доступных программ пользователь выбирает нужную (например, «Сила», «Выносливость»).
3. Choose Day Уточняется день недели, для которого будет загружена тренировка.
4. Get Exercise List Приложение формирует список упражнений на выбранный день.
5. Perform Exercise Пользователь выполняет упражнение, вводит данные: вес, повторения, подходы.
6. Mark as Complete После выполнения упражнение отмечается как завершённое.
7. Skip Exercise Если упражнение невозможно выполнить, пользователь может пропустить его.
8. Suggest Alternative Exercise В случае пропуска приложение предлагает альтернативу (например, замену на аналогичную группу мышц).
9. View Calculations History После завершения тренировки пользователь может просмотреть историю расчётов: объём нагрузки, прогресс, выполненные цели.

Сценарий использования представлен на рисунке 5.1:

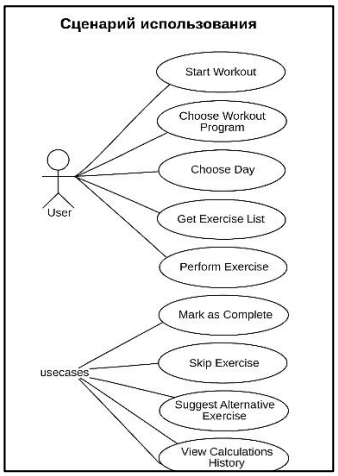

Рисунок 5.1 – Сценарий использования