МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
(ФГБОУ ВО «КубГУ»)

Факультет компьютерных технологий и прикладной математики
Кафедра прикладной математики

КУРСОВАЯ РАБОТА

РАЗРАБОТКА ПРИЛОЖЕНИЯ ПОСТРОЕНИЯ ПЛАНА ТРЕНИРОВОК ДЛЯ РАЗВИТИЯ ОПРЕДЕЛЕННЫХ НАВЫКОВ

Работу выполнил(а) _____ В. И. Корнейчук

Направление подготовки 09.03.03 Прикладная информатика курс 3
Направленность (профиль) Прикладная информатика в экономике

Научный руководитель
д-р. физ.-мат. наук, проф. _____ М. Х. Уртенов

Нормоконтролер
преподаватель _____ Е. В. Горбачева

Краснодар
2025

РЕФЕРАТ

Курсовая работа 40 с., 25 рис., 8 источн.

ПРИЛОЖЕНИЕ, БАЗА ДАННЫХ, МОТИВАЦИЯ, УВЕДОМЛЕНИЯ, ANDROID

Объектом исследования в данной работе является процесс разработки приложения для занятия спортом и отслеживания результатов.

Целью курсовой работы является разработка приложения построения плана тренировок для развития определенных навыков в среде разработки Android Studio со встроенным эмулятором для адаптивной разработки.

В результате работы было проведено исследование платформ для написания кода и выбор наиболее эффективных и удобных функций для достижения желаемого результата.

Практическая значимость проекта заключается в разработке инструмента для формирования индивидуальных тренировочных планов с учетом всех особенностей и желаний пользователей. Софт наполнен системой уведомлений, настройкой расписания и личных параметров – это повышает эффективность тренировок. В программном продукте предусматривается внедрение нейронной сети для генерации рекомендаций. Приложение может послужить платформой для личного пользования, а также для коммерческих фитнес сервисов.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Около десяти лет назад никто не мог подумать, что для большинства людей смартфоны смогут стать для них самым популярным и востребованным гаджетом в сфере правильных привычек. В настоящее время с помощью смартфона можно удобно фиксировать питание, прием витаминов, сколько воды было выпито за день и общую активность за день. Приложения для развития навыков могут заменить наставника личностного роста.

Человеку необходима постоянная мотивация для поддержания активности. Фитнес-приложения строятся на принципе стимулирующих извещениях, виртуальных наград за достигнутые результаты. Эта концепция действительно положительно влияет на пользователя и настраивает его на продуктивность.

Курсовая работа нацелена на разработку приложения построения плана тренировок для развития определенных навыков. Основная цель работы – обеспечить аудиторию удобным инструментом, содержащим персонализированные планы тренировок, который всегда будет под рукой.

Для написания описываемого приложения проведен анализ аналогов, выявлена лучшая среда для разработки, разработана клиентская и серверная части продукта. Реализуемое решение будет соответствовать всем потребностям пользователя и в итоге приведет его к желаемым результатам.

В процесс разработки входят: аналитический этап и техническое проектирование.

Целью курсовой работы является разработка приложения для развития определенных навыков.

В соответствии с поставленной целью определены основные четыре задачи исследования:

1) проанализировать существующий рынок решений и доступных сред разработки;

2) спроектировать систему хранения данных каждого юзера;

3) конкретизировать нужные опции и визуал системы;

4) изучить вопросы внедрения нейронной сети в разрабатываемый проект.

Курсовая работа состоит из трех глав, введения, заключения и списка литературы.

Первая глава курсовой работы содержит теоретические основы создания приложения и выбор технологий. Рассмотрены аналоги разрабатываемого приложения, актуальные функции и методы разработки.

Вторая глава посвящена проектированию и разработке приложения построения плана тренировок. В главе выявлены слабые и сильные стороны платформы для написания кода, определены ключевые критерии для выбора подходящих функций для разрабатываемого приложения.

Третья глава содержит оценку работоспособности приложения, описание процесса тестирования функциональности.

В заключении описываются результаты курсовой работы.

1 Теоретические основы создания приложений и выбор технологий

1.1 Сравнительный анализ рынка аналогичных приложений

Стратегией развития спорта в Российской Федерации до 2030 года определены главные направления трансформации отрасли физической культуры и спорта, в числе которых обозначена и цифровая трансформация, призванная стимулировать вовлеченность людей в сферу спорта и активности путем внедрения онлайн-сервисов в мобильных приложениях с тренировочными планами [1].

В 2024 году российская телекоммуникационная компания “Билайн” проанализировала частоту и число скачивания спорт-приложений. Анализ привел к выводу, что клиенты компании стали уделять большее внимание активностям дома, чем в спортивном зале [2]. Стоит отметить, что женская часть целевой аудитории значительно превалирует над мужской частью. Распределение пользователей по полу представлено в круговой диаграмме на рисунке 1 [2]:

Рисунок 1 – Процентное соотношение целевой аудитории по полу

А также анализ показал, что основной возраст пользователей находится в диапазоне от 35 до 40 лет [2], а пик пользования приложением и планами тренировок приходится на будние дни, в особенности на: понедельник, вторник и среду.

Современные пользователи нуждаются не просто в цифровых трекерах, а в действительно умных решениях, которые способны подстраиваться под желания каждого юзера. Рынок приложений с планами тренировок большой и это не предел. С каждым годом появляется все больше функций, способных привести процесс тренировок к идеалу.

Издание Forbes [3] утверждает, что инвесторы оптимистично смотря на будущее основных моделей онлайн-фитнеса. Перспективы таких проектов очень велики, поскольку на данный момент нет монополиста среди подобных спорт-решений. Мобильные приложения способствуют развитию спорта: увеличению количества соревнований, вовлеченности населения в систематические занятия спортом и росту спортивной индустрии в целом [4]. Цифровые технологии кардинально меняют сферу физкультуры и спорта, расширяя возможности пользователей [1].

Для создания качественного программного продукта в области спорта проведем анализ приложений, которые уже существуют и пользуются спросом. Оценка приложений основано на сравнении параметров: стоимость использования, персонализированные функции, удобство для пользователей. Рассмотрим таблицу аналогов на рисунке 2 для выявления сильных и слабых сторон, существующих на рынке решений:

	Отчёт о выполненной работе	Настройка здоровья	Составление расписания	Наличие обучающих видео	Синхронизация с другимии устройствами	Возможность изменять план тренировки	Премиум версия	Возможность сохранять план тренировки	Система уведомлений	Графики,отображающие прогресс	Взаимодействие с нейросетью	Возможность выбора уровня нагрузки	Наличие рекламы
Шпагат за 30 дней	+	+	+	+	+	-	+	+	+	+	-	+	+
Ежедневные Тренировки - фитнес	+	-	-	+	-	-	+	-	+	-	-	-	+
Bend: растяжка и гибкость app	+	+	+	+	+	-	+	+	+	-	-	+	-
FITSTARS	+	+	+	+	+	-	+	+	+	-	-	+	+
Фитнес,Спорт,здоровье.Фитнес центр в Telegram	-	-	-	-	-	-	-	+	-	-	-	-	-
BodBot	+	-	+	+	+	+	+		+	+	+	+	-
DDX Fitness	-	-	+	+	-	-	+	+	+	-	-	-	-
PUMATRAC	+	-	+	+	+	-	-	+	+	-	-	+	-
JEFIT	+	-	+	+	+	+	+	+	+	+	-	-	-
Virtuagym	+	-	+	+	+	+	+	+	+	-	-	-	-
Приложение "Алгус" для развития гибкости	+	+	+	-	+	+	-	+	+	+	+	+	-

Рисунок 2 – Таблица аналогов

Опираясь на данные, указанные в таблице, можно сделать вывод, что в Play Market есть множество сервисов с различным наполнением. Расширения программ, такие как: изменение шаблонного плана, адаптивность под каждого клиента, графики прогресса уже используются конкурентами, но зачастую на платной основе, либо разбросаны по разным разработкам. Стоимость подписки на дополнительные услуги колеблется от 800 рублей до 2500 рублей. Важно отметить, что в дополнительных подписках приложения предлагают персонализированные планы. трекинг прогресса, встроенные консультации с искусственным интеллектом и отсутствие рекламы. Проектируемый ресурс предлагает все ведущие функции на бесплатной основе. Это решение позволит приложению занять лидирующую позицию за счет доступности для аудитории.

1.2 Выбор среды разработки

В 2005 году Google увидел потенциал Android Inc. и приобрел ее. С этого момента началась настоящая революция. Google не только открыла Android Open Source Project (AOSP) для разработчиков со всего мира, но и внесла инновации, которые позволили Android стать самой популярной операционной системой на планете. В таблице указана популярность операционных систем с 2020 года в России. На рисунке 3 изображена статистика популярности операционных систем за последние пять лет:

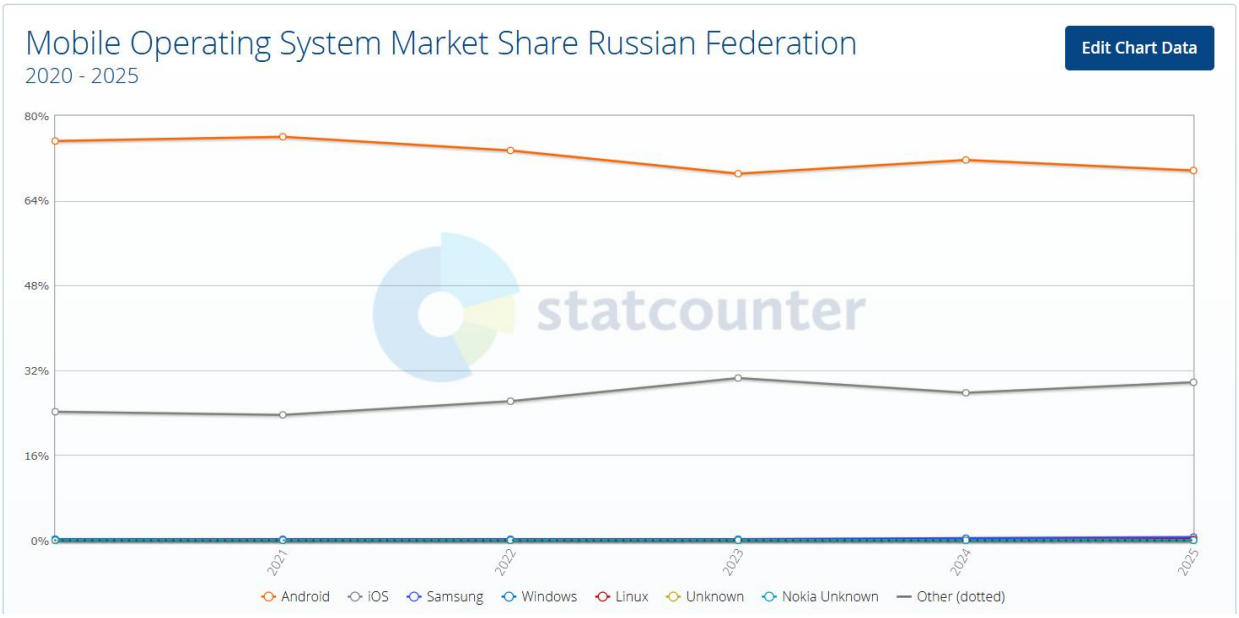


Рисунок 3 – Популярность операционных систем в сравнении друг с другом [6]

Выбор среды разработки один из самых важных этапов создания приложения на ОС Android. Современные инструменты дают полную свободу разработчикам в выборе языка программирования и среды разработки. На сегодняшний день одним из основных программных инструментов для написания приложений является Android Studio. Разумеется, есть и другие IDE, среди которых особенной популярностью пользуются Eclipse и IntelliJ [6], также на рынке представлены и кроссплатформенные фреймфорки, например, Flutter, React Native.

Сравнение сред разработки показано на рисунке 4:

Характеристика	Android Studio	Flutter	React Native	Eclips	IntelliJ
Тип	Нативная разработка, официальная IDE для Android	Кроссплатформенный фреймворк	Кроссплатформенный фреймворк	Устаревшая IDE	Универсальная IDE
Языки	Java, Kotlin	Dart	Java Script	Java	Kotlin, Java, Python
Поддержка Google	Полная	Поддерживается, но как фреймворк	Нет	Прекращена	Через плагины
Интеграция с Android SDK	Полная поддержка	С помощью плагинов	Через плагины	Через плагины	С помощью плагинов
Эмулятор	Встроен	Необходим сторонний	Необходим сторонний	Базовый эмулятор	Требуется настройка

Рисунок 4 – Сравнение IDE и фремворков

Исходя из характеристик, указанных на рисунке 4, видно, что Android Studio является лидером и имеет преимущество в полной поддержки и адаптации под систему Android. Eclips является устаревшей IDE, которую давно не поддерживает Google. IntelliJ популярна среди Android-разработчиков, однако замедляет процесс написания программного кода из за зависимостей от сторонних библиотек. Фреймворки удобны для написания Android и IOS проектов, но для поставленных задач курсовой работы они не подходят, потому что Flutter и React Native имеют ограниченные возможности оптимизации под конкретную платформу и при работе не исключены трудности с доступом к новым API платформ. Следовательно, большинство показателей Android Studio лучше, чем у других рассмотренных сред разработки. Android Studio - это нативная разработка, тесно связанная с работой Android SDK, является официальной IDE и имеет все необходимые инструменты для разработки, включая возможность подключения Android библиотек и Google плагинов без проблем. Используя в курсовой работе Android Studio, можно получить полный опыт работы с ведущей мобильной ОС Android.

Для того чтобы посмотреть, как работают аппликации, можно использовать либо подключенный к компьютеру реальный аппарат (смартфон

или планшет), либо эмулятор реального аппарата, существующий как внутренняя часть Android Studio.

Официальная поддержка языка Kotlin для разработки под Android была объявлена на Google I/O в 2017 году [7]. До этого существовало небольшое движение разработчиков Android, использующих Kotlin, несмотря на отсутствие официальной поддержки. С 2017 года Kotlin получил широкое распространение, и сегодня это наиболее предпочтительный язык для разработчиков на Android. Фреймворк Android был первоначально написан на Java. Это означает, что большинство классов Android, с которыми взаимодействуют акторы, — это классы Java. К счастью, Kotlin совместим с Java, поэтому проблем с этим не возникнет [7]. Kotlin общепринятый язык программирования для Android, многие приложения от Google уже переписаны с Java на Kotlin, что является главным показателем того, что Kotlin стандартом в Android-разработке. Java синтаксически сложнее своего аналога, программисты чаще допускают ошибки, используя этот язык программирования. Также Java не поддерживает Jetpack Compose и требует ручные проверки ошибок runtime во избежание аварийных завершений кода.

Таким образом, из анализа различных сред разработки лидером является Android Studio с классическим xml-файлом, которая напрямую связана с языками программирования Kotlin. Упомянутый язык программирования дает сильные характеристики разрабатываемому приложению.

2 Проектирование и разработка приложения

2.1 Реализация Frontend и анкетирование

Программируемое IT- решение в Android Studio на основе шаблона Empty Views Activity представляет собой стартовый экран для быстрого запуска разработки. Шаблон имеет базовую структуру с файлом Activity для написания кода обработки элементов, а также связанный с ним layout- файл, с помощью которого можно сконцентрироваться на логике приложения [7]. Отношение между Activity и layout показано на рисунке 5:

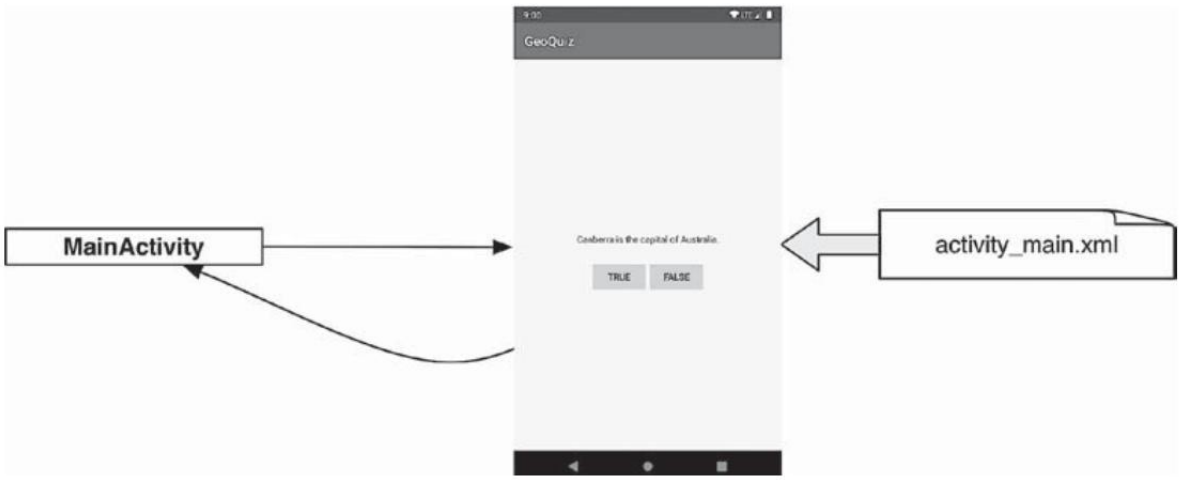


Рисунок 5 – Activity управляет интерфейсом

Выбранный шаблон имеет преимущество за счет простейшего функционала без лишних элементов управления. Каждый файл Activity имеет жизненный цикл, в котором прописывается обработка разрабатываемого приложения в условиях, когда приложение активно, свернуто, обработка паузы и закрытии. Представление жизненного цикла на рисунке 6:

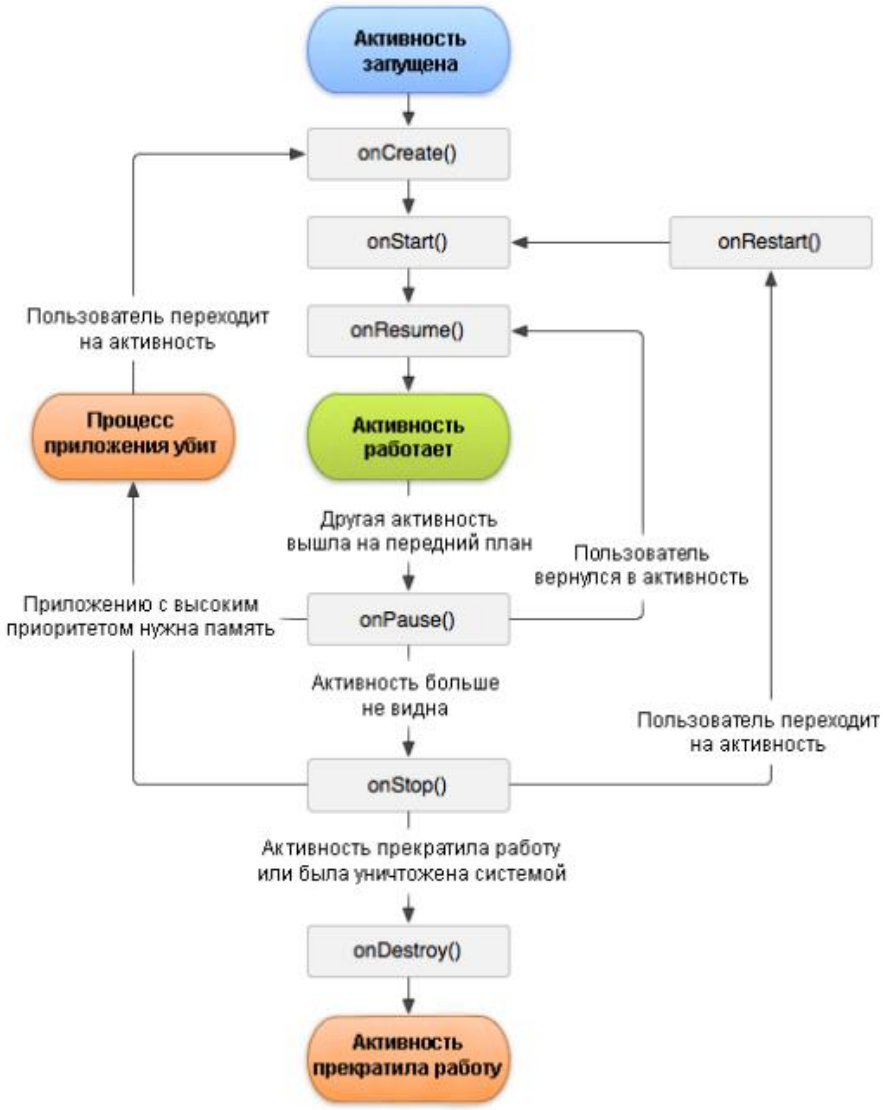
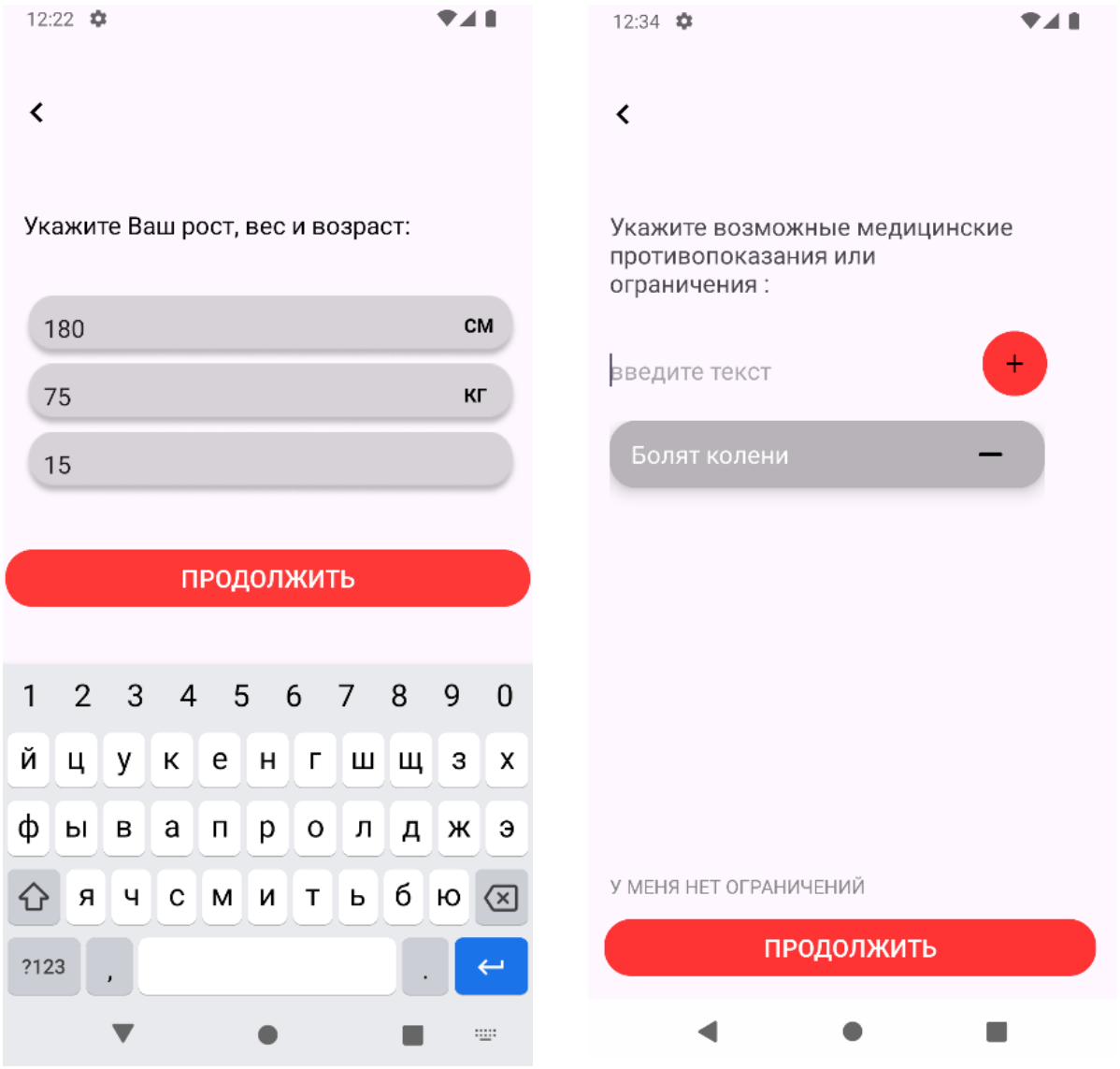


Рисунок 6 – Жизненный цикл Activity

Для качественного выполнения поставленной задачи курсовой работы необходимо создать продуманный фронтенд, опираясь на мнение будущих потребителей ПО. Было проведено анкетирование с помощью Google Forms среди студентов 1–3 курсов факультета компьютерных технологий и прикладной математики. Опрос учащихся помог сформировать дизайн приложения. Интерфейс должен содержать в себе возможность редактирования данных, просмотр плана тренировок, просмотр статистики активности и доступ к чату с искусственным интеллектом.

Большинство опрашиваемых подтвердили значимость учета физического состояния и медицинских противопоказаний, основные этапы сбора информации о здоровье клиенте показаны на рисунке 7:



(а) (б)

Рисунок 7 – (а) Заполнение параметров пользователя; (б) Медицинские противопоказания

Заполнение данных о здоровье является ключевым этапом создания персонализированного плана тренировок, поскольку это и есть улучшенный алгоритм гибкости системы. Не все приложения включают в себя указание состояния физического здоровья. В разрабатываемом сервисе можно не просто выбрать болезнь из списка, а самостоятельно ввести личные характеристики состояния организма.

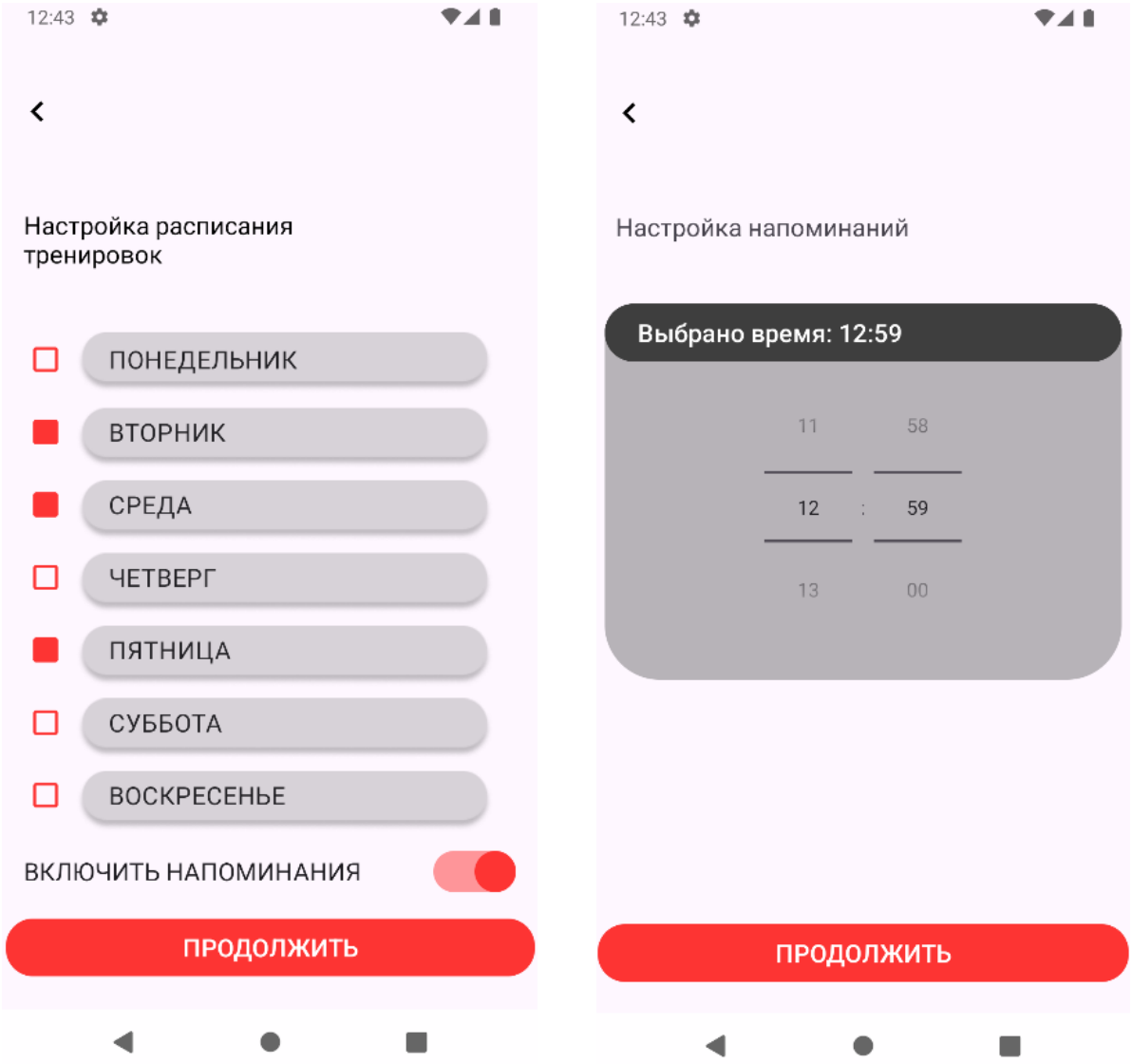
Также студенты, принимающие участие в анкетировании, отметили нехватку мотивации и регулярных напоминаний. Для удовлетворения этой надобности используются мотивирующие сообщения, которые приходят в назначенное программистом время и уведомления о тренировке, которые устанавливает непосредственно клиент. Систему уведомлений в ОС Android можно описать тремя пунктами:

1) установление разрешения в Manifest файле – это обязательный пункт для систем Android 13 и выше. В версия до тринадцатой разрешение имеет автоматическое согласие;

2) создание канала уведомлений – он создается один раз при первом вызове;

3) создание объекта NotificationReceiver – в объекте указывается ID канала, имя канала, содержание оповещения и иконка, важность уведомления.

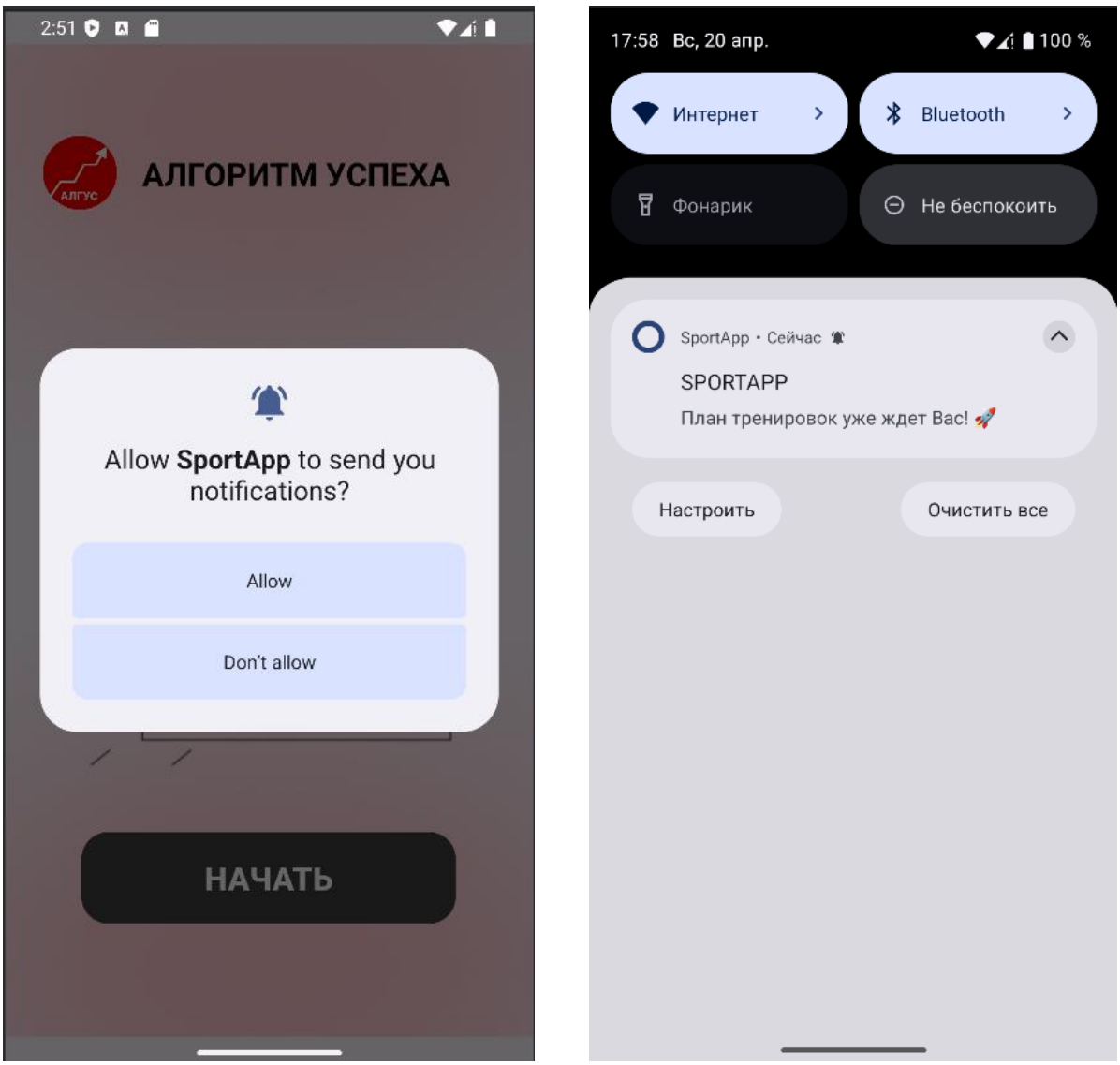
На рисунке 8 показан экран пользователя для установки извещения:



(а) (б)

Рисунок 8 – (а) Настройка дней уведомлений; (б) Настройка времени

На рисунке 9 демонстрируется напоминание, которое видит клиент:



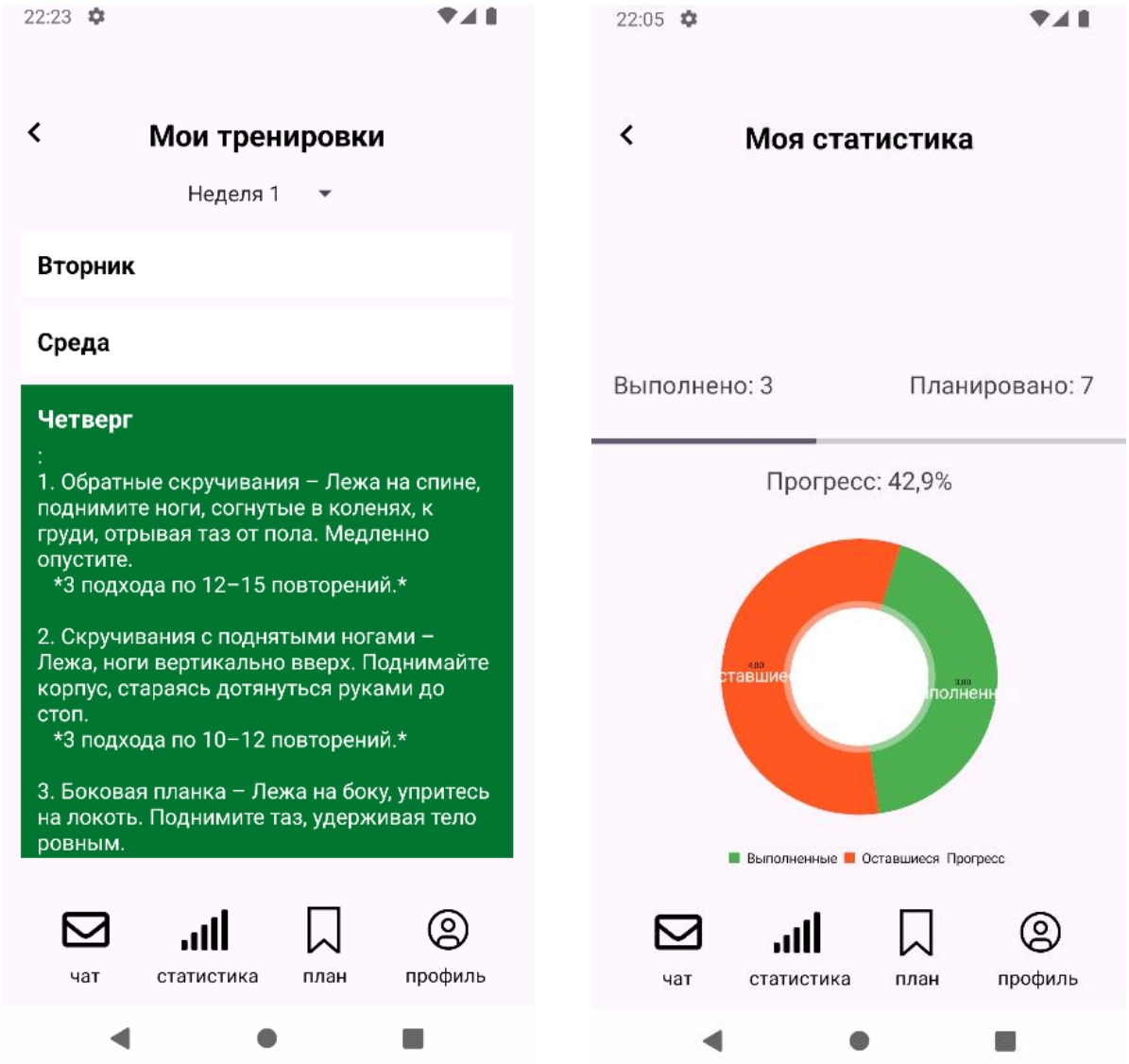
(а) (б)

Рисунок 9 – (а) Запрос разрешения; (б) Демонстрация созданного напоминания

Панель управления в личном кабинете каждого пользователя показана на рисунке 10:

Рисунок 10 – Навигация

Участники анкетирования хотят видеть наглядную статистику своих тренировок. Поэтому для стимулирования аудитории реализована система отслеживания прогресса, продемонстрированная на рисунке 11.



(а) (б)

Рисунок 11 – (а) Тренировочные планы; (б) Статистика

Абоненты могут отметить пройденные тренировки и посмотреть сводку в фрагменте «статистика».

Основная логика приложения разработана и частично показана в рисунках.

Реализованная визуальная часть обеспечена грамотно построенным жизненным циклом, ориентирована на целевую аудиторию и имеет сильные технические характеристики.

Логика приложения построена на принципе удобного взаимодействия клиента и продукта, добавлен график для визуальной наглядности успехов пользователя и удобно оформлен список тренировок для удобного ориентира в неделях и днях недели.

2.2 Разработка Backend и работа с нейронной сетью

Сфера Backend скрыта от глаз пользователя и происходит вне его глаз и устройства. Backend разработчики создают программы, которые выполняются на серверах и осуществляют вычисления, необходимые для работы разрабатываемой системы.

Для корректной работы приложения нужно много компонентов API слоя: база данных, аутентификация, аналитика и тд. В качестве разработки серверной части была выбрана облачная система Firebase от Google, так как она является готовым решением, содержащим все необходимые компоненты. Firebase также помогает защитить программное обеспечение, проверяя вводимые пользователями данные и отслеживая использование вашего приложения, чтобы гарантировать, что оно остается в соответствующих границах, защищая от неправильного использования или злоумышленников. Наглядная демонстрация преимущества облачной системы на рисунке 12:

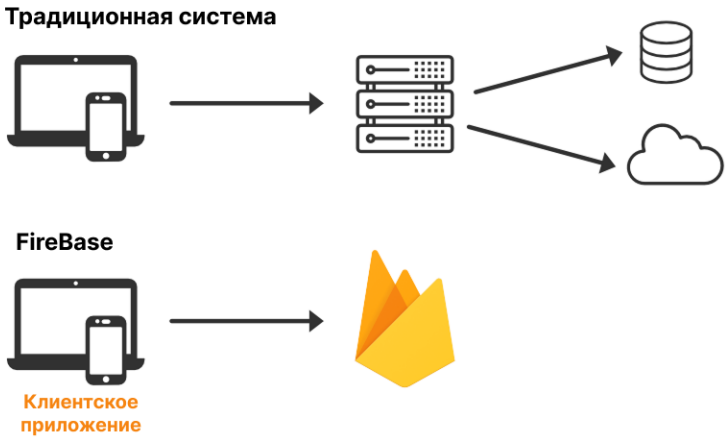


Рисунок 12 – Преимущество Firebase

Выбранное СУБД легко подключается к приложению. Разберем четыре этапа настройки серверной части:

1) для начала работы с Firebase нужно создать новый проект и добавить разрабатываемый проект с указанием уникального идентификатора, который находится в файле build. gradle;

2) далее для объединения базы данных и проекта требуется разместить конфигурационный JSON-файл в Android Studio, подключить плагин и зависимости для интеграции с Google;

3) подключить нужные интеграции: аналитику, базу данных, аутентификацию;

4) проверить корректность соединения тестовым запуском проекта.

На рисунке 13 показаны данные о проекте, которые находятся Firebase:

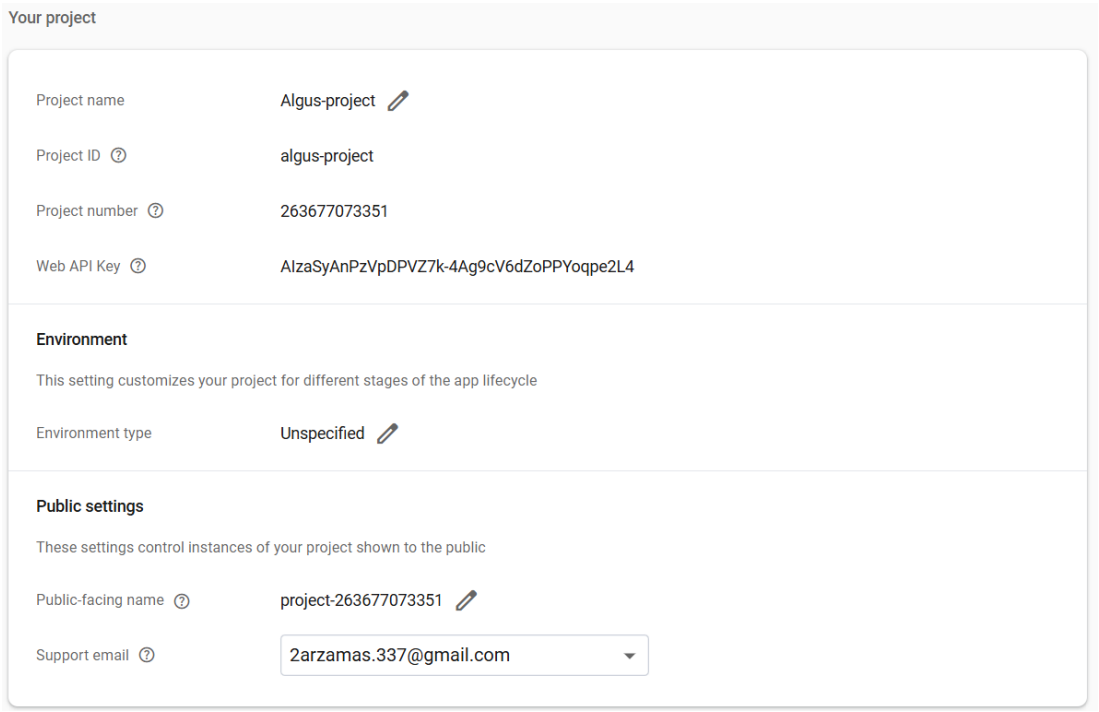


Рисунок 13 – Проект в Firebase

В MainActivity приложения создается USER_ID и передается на все последующие экраны, чтобы записывать все изменения в базу данных. В Realtime Database есть правила безопасности. Для записи, чтения и защиты данных устанавливаем разрешения, показанные на рисунке 14:

```
{
  "rules": {
    "users": {
      "$userId": {
        ".read": "auth != null && auth.uid == $userId",
        ".write": "auth != null && auth.uid == $userId",

        ".validate": "newData.hasChildren(['age', 'gender', 'name', 'height', 'weight'])",

        "age": {
          ".validate": "newData.isNumber() && newData.val() >= 12 && newData.val() <= 100"
        },
        "diseases": {
          ".validate": "newData.isString()"
        },
        "experience": {
          ".validate": "newData.isString() && (newData.val() == 'Новичок' || newData.val() == 'Средний' || newData.val() == 'Профессионал')"
        },
        "gender": {
          ".validate": "newData.isString() && (newData.val() == 'Мужской' || newData.val() == 'Женский')"
        },
        "height": {
          ".validate": "newData.isNumber() && newData.val() >= 50 && newData.val() <= 250"
        },
        "name": {
          ".validate": "newData.isString() && newData.val().length > 0"
        },
        "notification_hour": {
          ".validate": "newData.isNumber() && newData.val() >= 0 && newData.val() <= 23"
        },
        "notification_minute": {
          ".validate": "newData.isNumber() && newData.val() >= 0 && newData.val() <= 59"
        },
        "notifications_enabled": {
          ".validate": "newData.isBoolean()"
        },
        "plan_for": {
          ".validate": "newData.isString() && (newData.val() == 'Неделя' || newData.val() == 'Месяц' || newData.val() == 'Год')"
        },
        "selected_days": {
          ".validate": "newData.isString()"
        },
        "training_time": {
          ".validate": "newData.isString() && (newData.val() == 'Короткие (15-20 мин)' || newData.val() == 'Средние (30-40 минут)' || newData
        },
        "weight": {
          ".validate": "newData.isNumber() && newData.val() >= 30 && newData.val() <= 200"
        },
        "workoutType": {
          ".validate": "newData.isString()"
        }
      }
    },

    ".read": false,
    ".write": false
  }
}
```

Рисунок 14 – Правила для работы с данными

В правилах указана проверка авторизации пользователя, только зарегистрированный юзер может записывать и редактировать свои данные. Также в правилах базы данных прописана проверка валидации каждого пункта из анкетирования, для корректной записи параметров и генерирования тренировок. Валидация переменных исключает ошибки, например, отрицательный вес или маленький возраст. Выбранная база данных удобна тем, что быстро обновляется и сразу фиксирует корректность работы приложения. Также главным достоинством Firebase является максимальная защита данных. Самые популярные проблемы, с которыми сталкиваются пользовательские данные, это утечка данных, SQL-инъекции. Благодаря Firebase Authentication решаются проблемы безопасности за счет

автоматического хеширования паролей методом bcrypt и предотвращению атак по перебору паролей. База данных Firebase Realtime Database не уязвима для атак с использованием SQL-инъекций. С Firebase не нужно создавать команду SQL для выполнения запроса, вместо этого используется API, предоставляемый SDK.

Firebase содержит инструмент Analytics, с помощью которой разработчик может отследить активность пользователей, поведение и количественные параметры продуктивности. Наглядная диаграмма компонентов- рисунок 15.

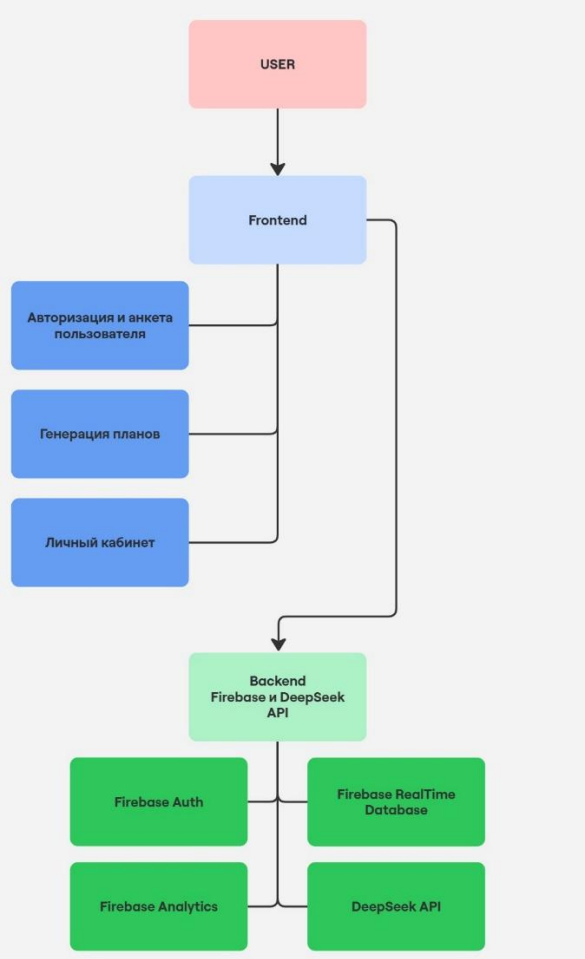


Рисунок 15 – Диаграмма компонентов

Исходя из вышеуказанного, работа с Firebase дает не только место для хранения данных и аналитику, но и дает возможность обеспечить цифровую платформу взаимодействием с нейронной сетью. Для разрабатываемого приложения была выбрана нейронная сеть DeepSeek. DeepSeek имеет преимущество перед другими нейронными сетями, которые существуют тем, что модель не требует обходить ограничения для российских пользователей, китайский искусственный интеллект умеет размышлять и анализировать, в отличие от других популярных вариантов сетей.

Алгоритм использования искусственного интеллекта прост в понимании и реализации. К проекту нейросеть подключается через доступный в сети API ключ. Для генерации персонализированных планов в методе класса считываются все сохраненные данные USER_ID из Firebase. Затем формируется скриптовый запрос, прописанный в программном коде. В шаблонный запрос добавляются характеристики и пожелания пользователя, затем сообщение сохраняется в базе данных и отправляется к API DeepSeek с использованием HTTP POST-запроса, формат отправки запроса application/json. Запрос отправляется на адрес https://api.deepseek.com/v1/chat/completions с помощью библиотеки OkHttpClient, которая помогает осуществлять сетевые запросы в многопоточном режиме и в случае ошибки в возврате ответа от нейронной сети, библиотека выводит сообщение с причиной возникшей проблемы. На рисунке 16 показаны установленные параметры запроса в нейросеть.

```
144            val json = JSONObject().apply {
145                put( name: "model",  value: "deepseek-chat")
146                put( name: "messages", JSONArray().apply {
147                    put(JSONObject().apply {
148                        put( name: "role",  value: "user")
149                        put( name: "content", message)
150                    })
151                })
152                put( name: "temperature",  value: 0.7)
153                put( name: "max_tokens",  value: 2000)
154            }
155
156            val body = json.toString().toRequestBody("application/json".toMediaType())
157            val request = Request.Builder()
158                .url(url)
159                .post(body)
160                .addHeader( name: "Authorization",  value: "Bearer $apiKey")
161                .addHeader( name: "Content-Type",  value: "application/json")
162                .build()
```

Рисунок 16 – Параметры запроса в DeepSeek

Параметр Authorization защищает доступ к API с помощью передачи ключа авторизации в заголовке каждого запроса. Это дает гарантию, что запрос подается авторизированным юзером. В коде указывается роль отправителя – user и содержание – message. Temperature отвечает за креативность модели, выбранное значение 0,7 сделает планы тренировок разнообразными, но в пределах нужной точности и соответствия желаниям

клиента. На рисунке 17 представлен код, в котором отправляется сообщение и проверяется ответ от DeepSeek:

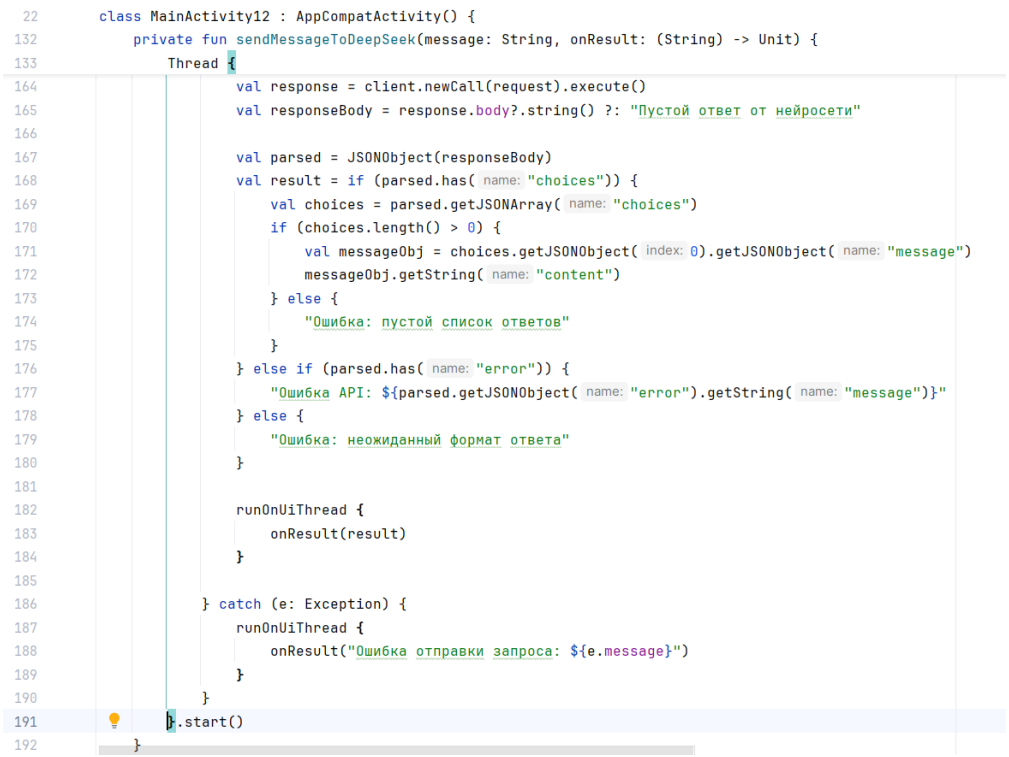


Рисунок 17 – Метод отправки сообщения

В переменной response вызывается метод с execute(), который и устанавливает связь с сервером DeepSeek.

После завершения генерации, планы тренировок сохраняются в личном кабинете пользователя и в информационной системе. Порядок сохранения данных можно увидеть на рисунке 18:

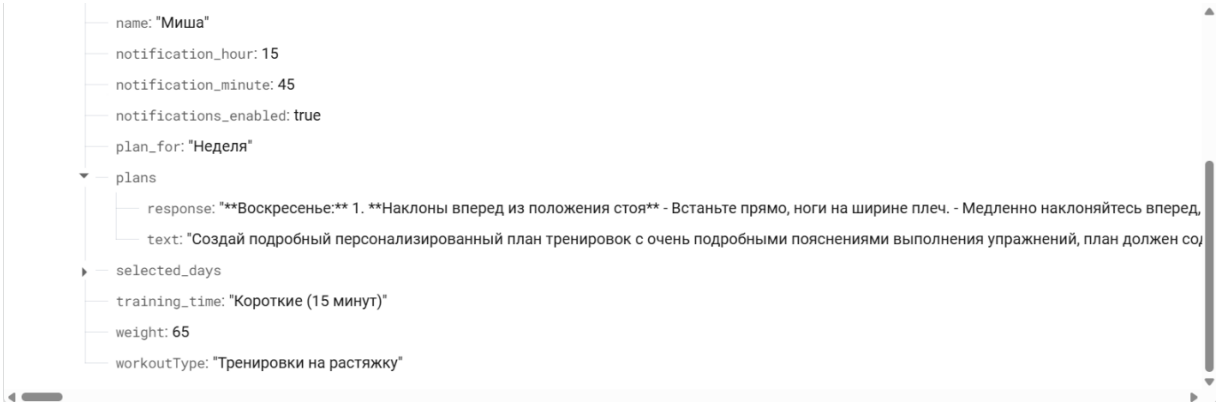


Рисунок 18 – Сохраненные данные

3 Тестирование и оценка корректности работы приложения

3.1 Проведение тестирования

Тестирование программного обеспечения – это процесс анализа программного средства и сопутствующей документации с целью выявления дефектов и повышения качества работоспособности. Тестирование разделяют на классификации: функциональное и нефункциональное.

Функциональное тестирование направлено на проверку корректности работы функциональности приложения. На этом этапе проверяется правильность реализации функциональных требований [8].

Нефункциональное тестирование – это тестирование, нацеленное на проверку нефункциональных требований приложения. К нефункциональным требованиям относятся: удобство использования, совместимость, производительность, безопасность и т.д. [8].

Несмотря на то, что те или иные виды тестирования принято причислять к функциональному или нефункциональному тестированию, в них всё равно присутствуют обе составляющие (как функциональная, так и нефункциональная), пусть и в разных пропорциях. Более того: часто проверить нефункциональную составляющую невозможно, пока не будет реализована соответствующая функциональная составляющая [8].

Для проверки удобства пользовательской части приложения были написаны пользовательские сценарии:

Сценарий 1. Создание нового плана тренировок.

Цель: создание нового тренировочного плана.

Акторы: пользователь.

Начальное состояние: пользователь открыл приложение и находится в личном кабинете на вкладке «профиль».

Основной сценарий

1. Пользователь выбирает опцию «Пересоздать план».

2. Система отображает анкету, чтобы юзер мог указать все данные заново.

3. Пользователь заполняет необходимые поля.

4. Система пересоздает план.

5. Система сохраняет план и отображает его на соответствующей вкладке «план».

Альтернативный сценарий

Предусловие:

На шаге 3 пользователь не заполнил обязательные поля.

6. Система не дает доступ перехода к следующему этапу анкетирования.

7. Пользователь заполняет недостающие поля и переход становится доступен.

Сценарий 2. Настройка уведомлений.

Цель: создать напоминания, чтобы не пропускать тренировки.

Акторы: пользователь.

Начальное состояние: пользователь открыл приложение и находится в личном кабинете на вкладке «профиль».

Основной сценарий

1. Пользователь выбирает кнопку «Расписание и напоминание».

2. Система отображает фрагмент с кнопками выбора дней и времени напоминаний.

3. Пользователь выбирает удобные для себя дни и нажимает switch для подтверждения согласия на отправку уведомлений.

4. Система отображает кнопку «Продолжить» для сохранения изменений.

5. Пользователь возвращается на экран «Расписание и напоминание».

6. Пользователь переходит на экран «Настроить время уведомлений» и выбирает удобный для себя тайминг.

7. Система отображает кнопку «Продолжить» для сохранения изменений.

8. Пользователь возвращается на экран «Расписание и напоминание».

9. Система запоминает план в формате день: время.

Альтернативный сценарий

Предусловие:

На шаге 6 пользователь не выбрал время.

7. Система не дает доступ перехода к следующему этапу анкетирования.

8. Пользователь заполняет недостающие поля и переход становится доступен.

Для нефункционального тестирования в коде используется инструмент Logcat. Logcat — это инструмент командной строки, который выводит журнал системных сообщений, включая сообщения, которые написаны в приложении

с помощью класса Log. Логи помогают отслеживать поведение приложения на устройстве и эмуляторе. Отправленные системой логи видны на рисунке 19:

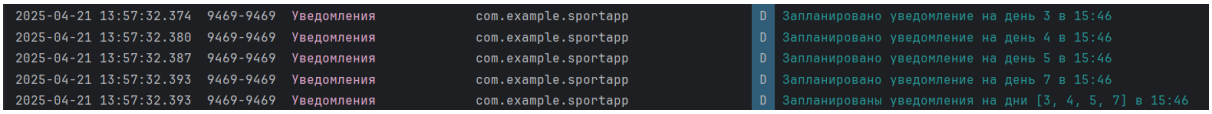


Рисунок 19 – Логирование системы информирования

В результате тестирования с помощью Logcat было выявлено, что функция планирования работает согласно требованиям, однако не всегда фактическое время отправленного уведомления совпадает с желаемым временем пользователя. Для решения проблемы, которая связана с неправильным временем отправки и получения напоминания, класс создания оповещений теперь использует AlarmManager, вместо WorkManager, который в свою очередь предназначался для фоновых задач.

Функциональное тестирование нацелено на тестирование графического пользовательского интерфейса, которое проверяет большую часть действий пользователей и взаимодействие компонентов приложения.

Для функционального тестирования личного кабинета был выбран фрагмент профиля юзера. Необходимо проверить корректность отображения имени пользователя. Тест проводится на фреймворке Espresso. Автоматизированный тест для фрагмента проиллюстрирован на рисунке 20:

```
@RunWith(AndroidJUnit4.class)
class ProfileFragmentTest {
    @Test
    fun userNameIsDisplayedCorrectly() {
        val bundle = Bundle().apply {
            putString("USER_ID", "testUserId")
        }
        launchFragmentInContainer<ProfileFragment>(fragmentArgs = bundle)
        onView(withId(R.id.textViewUserName))
            .check(matches(isDisplayed()))
    }
}
```

Рисунок 20 – Тест для проверки загрузки данных

В процессе функционального тестирования загрузки пользовательских данных и отображения был отмечен должный результат. Имя верно показывается на экране и внесенные изменения сразу принимаются в работу.

Для теста отображения и работу кнопок используется тест, показанный на рисунке 21:

```
@RunWith(AndroidJUnit4.class)
class ProfileFragmentTest {
    @Test
    fun buttonClick() {
        val bundle = Bundle().apply {
            putString("USER_ID", "testUserId")
        }
        launchFragmentInContainer<ProfileFragment>(fragmentArgs = bundle)
        onView(withId(R.id. settingsButton))
            .check(matches(isDisplayed()))

onView(withId(R.id. settingsButton))
            .perform(matches(isDisplayed()))

    }
}
```

Рисунок 21 – Тест для проверки отображения элементов на экране

Тестирование показало хорошие результаты функциональности кнопок и пользовательские сценарии функционируют без ошибок.

В Android Studio Profiler было проведено оценивание производительности. Вывод можно наблюдать на рисунке 22:

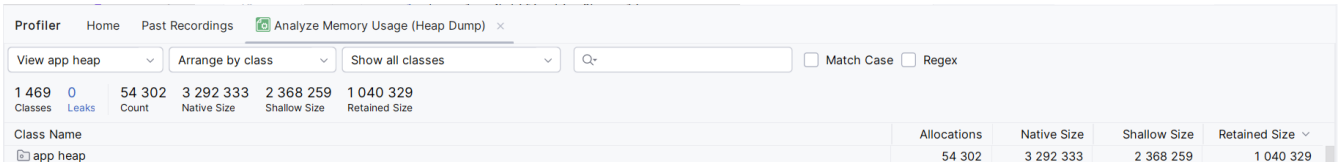


Рисунок 22 – Analyze Memory Usage

Общие выводы из полученной таблицы: количество утечек памяти 0; количество объектов составляет 54302 — это число находится на границе средним приложением и сложным, в случае с анализируемой системой такое число объектов напрямую связано с подключенной нейронной сетью и реализованным чатом в личном кабинете, в перспективе стоит рассмотреть возможное наличие лишних аллокаций; Native память занимает 3,29 Мб-это норма для среднего приложения с тем учетом, что в мобильном приложении имеются изображения и анимации; большая разница между Retained Size и Shallow size свидетельствует о зависимости классов между собой, это очевидно, поскольку каждый экран передает друг другу ID клиента для корректной записи в базу данных всех изменений.

По результатам всех проведенных тест-кейсов, тестируемый продукт был проверен на различных эмуляторах с отличными друг от друга экранными разрешениями и таких системах, как: Oreo (API 27), S (API 31), Vanilla Cream (API 35). Все запуски приложения были успешны, возможные ошибки были устранены. Общая продуктивность программного продукта положительная и имеет хорошую базу, которую можно будет улучшать и оптимизировать.

3.2 Анализ результатов аналитики и будущее проекта

В серверной части создаваемой платформы была организована работа с Firebase, которая предоставляет разработчикам собранную аналитику по проекту. В Analytics разработчик может отследить количество активных пользователей – это полезная возможность для понимания надобности разработанного сервиса; контроль количества новых пользователей дает понять, насколько хорошо работает продвижение; самая полезная функция аналитики – это отображение часто используемых фрагментов, таким образом просто сделать выводы о наиболее важных функциях приложения, которые нравятся целевой аудитории и которые можно сделать удобнее и красивее для клиентов.

В тестировании принимали участие 9 студентов 18–20 лет, которые использовали приложение в повседневной жизни на протяжении 14 дней. Вовлеченность студентов показана на рисунке 23:

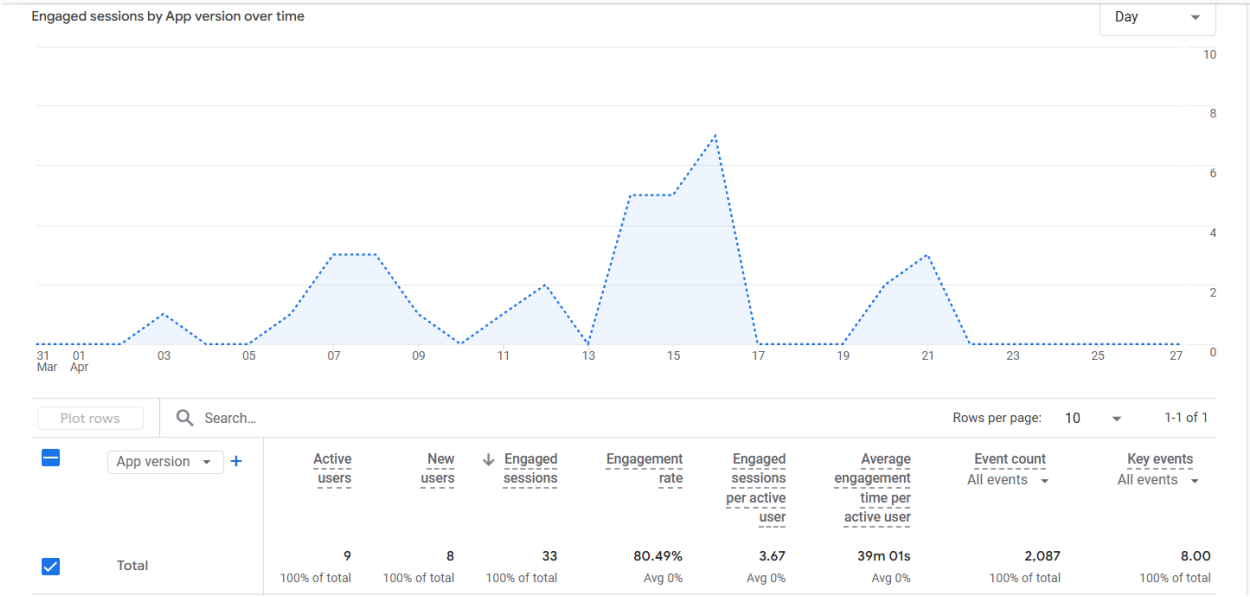


Рисунок 23 – Метрика вовлеченности пользователей

На этапе тестирования можно заметить, что в один день максимальное число пользователей было 7. Этого достаточно, чтобы сделать позитивные выводы, поскольку всего было задействовано 10 студентов и общая заинтересованность составляет 80%. Среднее время использования – 39 минут, а значит, что пользователи действительно тренировались и были увлечены идеей системы. В Analytics отмечается частое посещение экранов с планами тренировок и настройки. На рисунке 24 показана статистика популярных экранов.

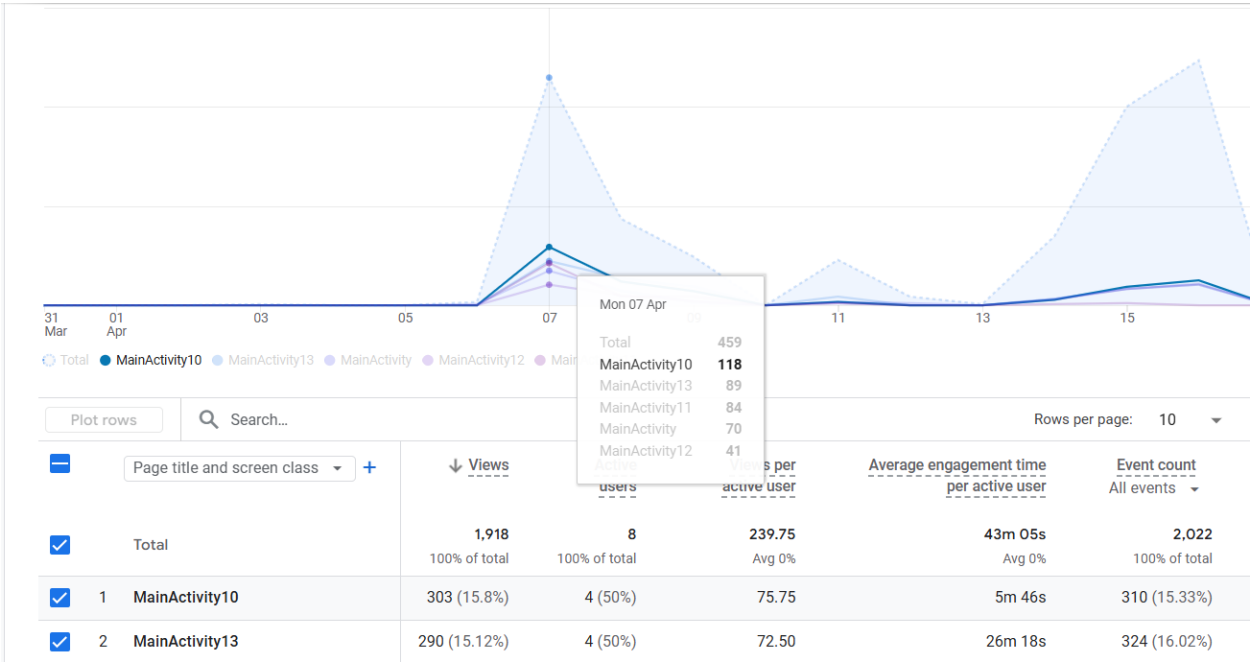


Рисунок 24 – Статистика экранов

По статистике видно, что чаще всего пользователи переходят на MainActivity10 (экран выбора дней для тренировок), MainActivity13 (личный кабинет) и MainActivity11 (экран настройки времени для уведомлений). Таким образом, самыми популярными функциями являются напоминания и личный кабинет со всеми реализованными фрагментами. Это свидетельствует о заинтересованности в максимальной адаптивности тренировок.

NPS опрос полезный инструмент в маркетинге. Во фрагменте «статистика» было принято решение использовать такой формат обратной связи для фиксирования качества продукта. На рисунке 25 продемонстрирован NPS опрос:

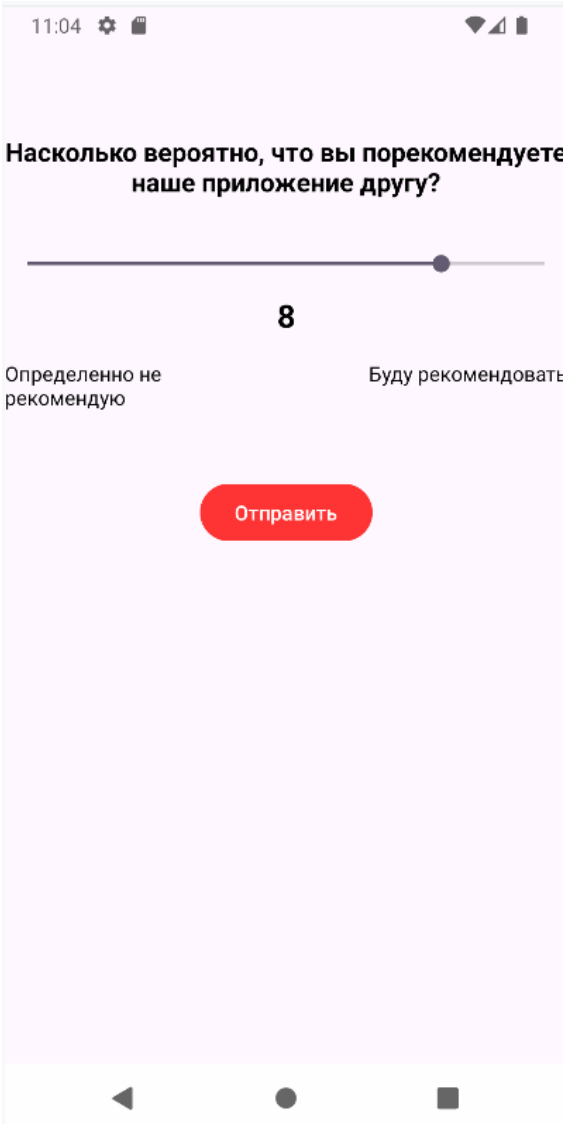


Рисунок 25 – Статистика экранов

Преимущество такой обратной связи заключается в ненавязчивости, вопрос будет отображаться у клиентов спустя неделю использования

мобильного приложения. Итог показал, что 1 из 9 ответов был негативным (оценка 0–6), а остальные нейтральные и позитивные (оценка 6–10).

На основании аналитики можно сделать выводы о дальнейших расширениях проекта. Платформу можно совершенствовать до бесконечности, но на данном этапе все работает корректно и без перебоев. В перспективах возможно расширение тренировочных планов с сопровождением видео инструкций. В коммерческих целях можно рассмотреть внедрение монетизации с помощью Firebase, подключив рекламу. Jetpack Compose — это современный набор инструментов Android для создания собственного пользовательского интерфейса. Он упрощает и ускоряет разработку пользовательского интерфейса на Android, однако не поддерживается старыми версиями Android (от 27 и ниже). Jetpack Compose можно рассматривать для постепенного улучшения проекта, добавляя на экраны красивые переходы и интересные анимации.

Касаясь вопроса улучшения проекта, нельзя игнорировать интеграции с другими гаджетами, например, умными часами. Wear OS поддерживается Google, а значит в будущем несложно будет сделать синхронизацию приложения с часами. Просмотр уведомлений и отслеживание активности станет еще удобнее за счет мгновенного доступа, так как умные часы всегда на запястье человека.

Таким образом, аналитические данные помогают программистам оптимизировать уже реализованные функции, а еще реализовать новые на основании желаний аудитории.

ЗАКЛЮЧЕНИЕ

В результате выполнения данной курсовой работы было разработано мобильное приложение построения плана тренировок, позволяющее упростить тренировочный процесс и поиск подходящей информации, потому что цифровая платформа оснащено интеграцией с нейронной сетью, которая участвует как помощник и тренер.

В процессе работы были изучены ключевые моменты создания программного обеспечения. Прошло ознакомление с архитектурой Android Studio 2024.3.1, с синтаксисом языка программирования Kotlin версии 2.0.0, с правилами работы хранения данных в Firebase 16.4.0 и ключевым моментом – подключением нейросети к сервису. Также реализован ключевой функционал, направленный на увеличение числа желающих сделать свою жизнь активнее.

Несмотря на хороший результат работы, в процессе создания ПО были выявлены ограничения и сложности. Например, Firebase имеет лимит количества операций аутентификации. При масштабируемости проекта нужно будет поменять тарифный план с бесплатного Spark на платный и расширенный Blaze, стоимость Blaze – 2073 рублей. Сложность работы с DeepSeek связана с задержкой ответов от нейросети при большой нагрузке на DeepSeek. Недостатком выбранного стека технологий является скорость создания интерфейса. XML разметка удобна для новичков, потому что разработчик наглядно видит изменения экранов, однако Jetpack Compose позволила бы оформить визуальную часть быстрее.

Приложение, созданное в рамках курсовой работы, представляет собой удобный и функциональный инструмент для пользователей, желающих развивать гибкость и улучшить свою физическую форму. В процессе разработки были учтены все требования к UI (User Interface) и функционалу, что позволило создать электронный сервис с хорошей пользовательской поддержкой и высокой степенью персонализации.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Цифровая реальность: как мобильные фитнес-приложения повышают качество жизни [сайт]. – URL: https://cyberleninka.ru/article/n/tsifrovaya-realnost-kak-mobilnye-fitnes-prilozheniya-povyshayut-kachestvo-zhizni/viewer (дата обращения: 18.04.2025).

2. Билайн: аналитика в России вырос спрос на спортивные приложения [сайт]. – URL: https://beelinenow.ru/articles/bilayn-analitika-v-rossii-vyros-spros-na-sportivnye-prilozheniya/ (дата обращения: 18.04.2025).

3. Непаханое поле для вложений: почему взлетели проекты в сфере онлайн-фитнеса [сайт] // Forbes. – URL: https://www.forbes.ru/karera-i-svoy-biznes/427971-nepahanoe-pole-dlya-vlozheniy-pochemu-vzleteli-proekty-v-sfere-onlayn (дата обращения: 18.04.2025).

4. Национальная экономика: учебник и практикум для бакалавриата и магистратуры / под редакцией А. В. Сидоровича. — Москва: Издательство Юрайт, 2017. — 485 с. — (Высшее образование). — ISBN 978-5-534-03278-9. — URL: https://urait.ru/bcode/403552 (дата обращения: 18.04.2025).

5. Статистика рынка мобильных ОС [сайт] // StatCounter. – URL: https://gs.statcounter.com/os-market-share/mobile/worldwide/#yearly-2013-2020 (дата обращения: 18.04.2025).

6. Фрайман, З. С. Создание приложений для смартфонов и планшетов под ОС Android: Практический курс / З. С. Фрайман. — Москва: ЛЕНАНД, 2019. — 504 с. — ISBN 978-5-9710-6098-7.

7. Android. Программирование для профессионалов / Б. Филлипс, К. Стюард, К. Марсиакано, Б. Гарднер. — 4-е изд. — Санкт-Петербург: Питер, 2021. — 704 с. — (Серия «Для профессионалов»). — ISBN 978-5-4461-1657-7.

8. Куликов, С. С. Тестирование программного обеспечения. Базовый курс / С. С. Куликов. — 3-е изд. — Минск: Четыре четверти, 2020. — 312 с. — ISBN 978-985-581-362-1.