

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет

вычислительной техники

Кафедра

«Вычислительная техника»

Направление подготовки: 09.03.01 «Информатика и вычислительная техника»

Профиль подготовки: «Вычислительные машины, комплексы, системы и сети»

БАКАЛАВРСКАЯ РАБОТА

на тему

Информационная система контроля

тренировки спортсменов

Студент


(подпись, дата)

Агеев Сергей Анатольевич
(ФИО полностью)

Руководитель


(подпись, дата)

Трокоз Д. А.
(фамилия, инициалы)

Нормоконтролёр


(подпись, дата)

А.В. Кучин
(фамилия, инициалы)

Работа допущена к защите (протокол зас. кафедры от 21.05.2019 г. № 38-09-14)

Заведующий кафедрой


(подпись)

М.А. Митрохин
(фамилия, инициалы)

Работа защищена с отметкой хорошо (протокол заседания ГЭК от 01.07.2019г. № 19)

Секретарь ГЭК


(подпись)

Е.И. Гурин
(фамилия, инициалы)

Пенза, 2019

Федеральное государственное бюджетное образовательное учреждение
высшего образования

ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

«Утверждаю»

Заведующий кафедрой

 М. А. Митрохин

 2019г.

**ЗАДАНИЕ
НА БАКАЛАВРСКУЮ РАБОТУ**

1. Студент Агеев Сергей Анатольевич гр. 15BB1 факультета вычислительной техники.

2. Тема работы: «Информационная система контроля тренировки спортсменов» Тема утверждена приказом ПГУ № 554/о от "29"апреля 2019 г.

3. Руководитель работы к.т.н., доцент каф. ВТ ПГУ Трокоз Д. А.

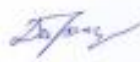
4. Задание на работу (назначение разработки, исходные данные и т.п.).

Информационная система предназначена для создания и отслеживания прогресса по тренировочным планам.

5. Перечень подлежащих разработке вопросов.

Требуется произвести анализ предметной области, выбрать системы управления базой данных, выбрать среду разработки, определить интерфейс приложения, спроектировать приложение, реализовать приложение, отладить и провести тестирование разработанного приложения.

6. Календарный график выполнения работы.

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1.	Анализ предметной области	10	07.05.19	
2.	Определение требований к системе	5	12.05.19	
3.	Проектирование приложения	10	18.05.19	
4.	Написание программы	25	25.05.19	
5.	Отладка и тестирование приложения	20	06.06.19	
6.	Оформление выпускной работы	30	23.06.19	

Дата выдачи задания "29" апреля 2019 г.

Руководитель бакалаврской работы  Трокоз Д. А.
(подпись, дата)

Задание к исполнению принял студент  Агеев С. А.
(подпись, дата)

Работу к защите допустить

Декан факультета  Фионова Л. Р.
(подпись, дата)

РЕФЕРАТ

Пояснительная записка содержит 72 страниц формата А4, 28 рисунков, 3 таблиц, 11 источников, 3 приложения.

Объект разработки - информационная система контроля тренировки спортсменов

ANDROID, БАЗА ДАННЫХ, ИНФОРМАЦИОННАЯ СИСТЕМА, SQL-ЗАПРОСЫ, ЛОКАЛЬНОЕ ПРИЛОЖЕНИЕ, СЕРВЕРНОЕ ПРИЛОЖЕНИЕ.

Целью дипломной работы является разработка информационной системы контроля тренировки спортсменов, предназначенной для создания тренировочных планов и отслеживания по ним прогресса. Это достигается путем сбора данных с устройства в процессе выполнения тренировки и занесения этих данных в SQLite и Firebase RealtimeDatabase для хранения и дальнейшего сбора статистики.

В результате проведенной работы создан проект, протестированы программные средства, удовлетворяющие поставленной задаче.

В пояснительной записке приведен анализ предметной области, обзор технологии разработки и отладки программы.

Графическая часть содержит ER-диаграмму БД и UML-диаграммы программы.

					ПГУ 1.090301.01.001 ПЗ			
Изм	Лист	№ докум.	Подпись	Дата	Информационная система контроля тренировки спортсменов Пояснительная записка	Литер.	Лист	Листов
Разработ.	Агеев С. А.					У	4	72
Проверил	Трокоз Д.А.					ФВТ, каф. ВТ Гр. 15ВВ1		
Н. контр.	Кучин А. В.							
Утвердил	Митрохин М.А.							

Содержание

РЕФЕРАТ.....	4
ПЕРЕЧЕНЬ ПРИНЯТЫХ СОКРАЩЕНИЙ	7
ВВЕДЕНИЕ.....	8
1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ	10
1.1 Описание предметной области	10
1.2 Обзор существующих решений.....	10
1.3 Актуальность выбранной темы	12
1.4 Выбор средств разработки	12
2 ОПИСАНИЕ ТРЕБОВАНИЙ К СИСТЕМЕ.....	15
2.1 Требования к надежности	15
2.2 Требования к пользователю ИС	15
2.3 Требования к информационной и программной совместимости.....	16
2.4 Требования к программной документации	16
3 ПРОЕКТИРОВАНИЕ ПРИЛОЖЕНИЯ	17
4 РЕАЛИЗАЦИЯ ПРОЕКТА.....	21
4.1 Процесс разработки	21
4.1.1.Реализация Аутентификации пользователей.....	21
4.1.2 Реализация БД.....	23
4.1.3 Реализация интерфейса приложения	26
4.2 Отладка и тестирование	31
5 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ	34
ЗАКЛЮЧЕНИЕ	40
СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ.....	41
ПРИЛОЖЕНИЕ А	42
ЛИСТИНГ ПРОГРАММЫ.....	42
Приложение А.1 Файл LoginActivity.java	43
Приложение А.2 Файл RegisterActivity.java.....	46
Приложение А.3 Файл MainActivity.java.....	48
Приложение А.4 Файл GuidePagerActivity.java	53
Приложение А.5 Файл ProfileFragment.java.....	55

					ПГУ 1.090301.01.001 ПЗ			
Изм	Лист	№ докум.	Подпись	Дата				
Разработ.	Агеев С. А.				Информационная система контроля тренировки спортсменов Пояснительная записка	Литер.	Лист	Листов
Проверил	Трокоз Д.А.					У	5	72
						ФВТ, каф. ВТ Гр. 15ВВ1		
Н. контр.	Кучин А. В.							
Утвердил	Митрохин М.А.							

Приложение А.6 Файл PlanFragment.java.....	57
Приложение А.7 Файл WithoutPlanFragment.java	59
Приложение А.7 Файл CreatePlanActivity.java	60
ПРИЛОЖЕНИЕ Б.....	64
РИСУНКИ И ДИАГРАММЫ	64
ПРИЛОЖЕНИЕ В	67
ПРЕЗЕНТАЦИЯ	67

					ПГУ 1.090301.01.001 ПЗ			
<i>Изм</i>	<i>Лист</i>	<i>№ докум.</i>	<i>Подпись</i>	<i>Дата</i>	Информационная система контроля тренировки спортсменов Пояснительная записка	<i>Литер.</i>	<i>Лист</i>	<i>Листов</i>
<i>Разработ.</i>	Агеев С. А.					У	6	72
<i>Проверил</i>	Трокоз Д.А.					ФВТ, каф. ВТ Гр. 15ВВ1		
<i>Н. контр.</i>	Кучин А. В.							
<i>Утвердил</i>	Митрохин М.А.							

ПЕРЕЧЕНЬ ПРИНЯТЫХ СОКРАЩЕНИЙ

ВКР – Выпускная квалификационная работа.

ИС – Информационная система.

ПО – Программное обеспечение.

СУБД – Система управления базами данных.

БД – База данных.

MVVM – Model View ViewModel.

ПКМ – Правая кнопка мыши.

ЛКМ – Левая кнопка мыши.

ПИ (UI) – Пользовательский интерфейс

					ПГУ 1.090301.01.001 ПЗ	Лист
						7
Изм.	Лист	№ докум.	Подпись	Дата		

ВВЕДЕНИЕ

В настоящее время практически любые задачи в той или иной степени поддаются автоматизации. Самым простым примером будет использование, например, роботов в сборочных цехах, вместо обычного ручного труда. Это колоссально повышает производительность и эффективность труда, а также позволяет людям сосредоточиться на более важных задачах, вместо выполнения монотонных и рутинных задач.

Помимо этого, огромное количество вещей переходят в так называемый «цифровой формат». Электронные книги, вместо обычных бумажных, электронные деньги, заменяющие наличные, электронные письма, как замена обычным конвертам и т.д.

По отдельности, каждая из этих тенденций позволяют увеличить эффективность труда, качество жизни, комфорт и доступность многих вещей, которые раньше не были так доступны как сейчас. Тогда как за счет конвергенции этих областей создаются по-настоящему меняющие привычный образ жизни людей.

Смартфон и интернет, наверное, самые яркие примеры «слияния» областей автоматизации и диджитализации. Они настолько прочно и стремительно вошли в нашу жизнь, что теперь могут заменять огромное количество совершенно разных вещей одним устройством.

Что касается сферы спорта, то ее также коснулись изменения от перехода к так называемому цифровому обществу. Огромное количество датчиков, сенсоров, ежесекундно собирающих тонны информации, устройства, позволяющие сделать спорт менее подверженным к «человеческому фактору» и т.д.

От перехода к цифровому образу жизни есть и свои минусы. Объемы генерируемой информации растут огромными темпами и вопрос эффективного и доступного хранения данных с каждым днем становится все более острым.

					ПГУ 1.090301.01.001 ПЗ	Лист
						8
Изм.	Лист	№ докум.	Подпись	Дата		

Все более очевидным становится то, что только развитие всех сопутствующих сфер информационных технологий поможет избежать множество проблем на пути к созданию информационных систем и устройств способных удовлетворить все слои общества.

Целью данной выпускной квалификационной работы является разработка информационной системы предназначенной для контроля тренировок спортсменов. Контроль тренировок будет осуществлять пользователем путем создания персонализированного плана тренировок, отслеживания различной информации в процессе их выполнения.

Для достижения поставленной цели были поставлены следующие задачи:

- Анализ предметной области;
- Постановка задачи на разработку программного средства для создания тренировочных планов и отслеживания общего прогресса по ним;
- разработка схем БД (локальной и серверной);
- реализация разработанной схемы в СУБД (SQLite, Firebase RealtimeDatabase);
- создание макетов пользовательского интерфейса для работы с локальной и серверной БД;
- автоматизация работы с макетами пользовательского интерфейса, базами данных и сервером;
- тестирование ИС.

1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1 Описание предметной области

Разрабатываемая информационная система предназначена для контроля и учета статистики по тренировкам спортсменов в контексте плана, либо в рамках только одной тренировки. Выбор плана осуществляется пользователем, он может достаточно детализировано настроить план, выбирая сколько недель он будет длиться, сколько всего будет тренировочных дней и какие будут типы тренировок. После создания плана пользователь может просмотреть всю детальную информацию о нем и приступить к тренировкам. В процессе тренировки, в зависимости от ее типа, приложение собирает все необходимые данные и затем формируется статистика.

1.2 Обзор существующих решений

На сегодняшний момент существует несколько приложений, позволяющих пользователям тренироваться по определенному плану, следить за своими результатами и прогрессом. В данном разделе будут рассмотрены не все существующие решения, но наиболее пользующиеся спросом из них.

Nike Run Club. Приложение, разработанное и поддерживаемое компанией Nike, которое в основном сфокусировано на работе с бегунами. Имеет хороший дизайн, позволяет выбирать план тренировок, присутствуют различные напоминания о тренировках, детально отслеживаются данные, позволяет даже видеть карту пробежки, в купе с данными о скорости бега, длительности и даже примерным количеством сожженных калорий. Одним из основных минусов приложения является то, что пользователь не может сам, полностью настроить свой график. Приложение само определяет план, трудность тренировок, основываясь на указанных данных и прогрессе. Но, для пользователей, которые более продвинуты в спорте, было бы неплохо, в добавок к уже существующей функциональности, добавить функцию самостоятельной настройки плана.

					ПГУ 1.090301.01.001 ПЗ	Лист
						10
Изм.	Лист	№ докум.	Подпись	Дата		

Также, стоит отметить то, что приложение ориентировано не только на беговые тренировки, но и на общефизические упражнения, но, для этого потребуется скачать дополнительное приложение, чтобы получить возможность не только бегать. Большинство пользователей, не особо охотно скачивают много приложений, ограничиваясь только необходимыми, поэтому, это может быть существенным минусом, так как пользователь может просто решить отказаться от этого вида тренировок, в таком случае, а это означает потерю большей части функциональности.

Runtastic. Данное приложение в основном обладает теми же характеристиками, что и предыдущее, но с более детальными настройками и отображением данных. Оно добавляет и улучшает все функции предыдущего приложения, например добавляя пользователю ленту новостей, в которой, зачастую публикуются весьма полезные статьи о правильном питании, как правильной бегать или выполнять упражнения и т.д.

В дополнение ко всему, приложение, обладает теми же минусами: для простых тренировок, например, отжиманий, пользователю нужно скачивать дополнительное приложение. Детальной настройки плана, для продвинутых пользователей также нет. За счет лучшей проработки функциональности, приложение уже становится не бесплатным и обладает моделью подписки, где пользователю каждый месяц нужно будет платить, для работы всех доступных функций в приложении.

Стоит также отметить, что оба приложения все же снижают воздействие этих минусов на пользователей, например, добавляя отображение данных из разных приложений в одном (как правило это приложения от одного разработчика), а в случае с настройкой плана, позволяют пользователю менять дни тренировок, в течение недели. Более качественная реализация функциональности, также снижает воздействие этих минусов.

					ПГУ 1.090301.01.001 ПЗ	Лист
						11
Изм.	Лист	№ докум.	Подпись	Дата		

1.3 Актуальность выбранной темы

Исходя из информации, представленной в разделе «Обзор имеющихся решений», мы видим, что даже самые широкие по охвату аудитории приложения обладают минусами. Nike Run Club хоть и бесплатное приложение, но в нем заметно меньше функциональности и оно реже обновляется, реже добавляется новый функционал или совершенствуется уже существующий. Runtastic, в свою очередь, является платным приложением (чтобы использовать весь доступный функционал), хоть и с более качественной реализацией, частыми обновлениями, но. Приложения, помимо беговых тренировок, позволяет отслеживать общефизические тренировки, но только в другом приложении, за счет чего, есть вероятность того, что пользователи пропустят мимо весь функционал отдельного приложения.

Таким образом, актуальность выбранной темы обосновывается тем, что разработанное приложение будет бесплатным, с открытым исходным кодом, что позволит любому желающему разработчику внести вклад в развитие приложения, а также расширять доступные виды спорта в одном приложении.

1.4 Выбор средств разработки

В ходе разработки были проанализированы различные средства, технологии и языки программирования.

Программные средства, использованные в данной работе:

- Облачная платформа Firebase, для предоставления серверных услуг, в т.ч. NoSQL БД Firebase RealtimeDatabase и средства для аутентификации пользователей в приложении;
- СУБД SQLite в качестве локальной базы данных, которая по умолчанию встроена в Android;
- Android Studio – как среда разработки.

Использованные языки программирования, языки разметки и языки запросов к БД:

					ПГУ 1.090301.01.001 ПЗ	Лист
						12
Изм.	Лист	№ докум.	Подпись	Дата		

- SQL для работы с локальной БД;
- JSON для построения дерева NoSQL серверной Firebase RealtimeDatabase БД
- XML для создания макетов экрана;
- Java для написания логики приложения.

Выбор средств разработки исходил из функциональных возможностей, высокой производительности и бесплатного использования.

Firebase

Firebase – это платформа, предоставляющая различные облачные услуги для веб-сайтов и приложений. Простыми словами, Firebase предоставляет бэкенд для приложения, позволяя добавить в приложение облачную базу данных, чтобы хранить все данные пользователей не только локально, но еще и на сервере, а также делает возможным авторизацию пользователей на основе электронной почты и пароля.

Firebase RealtimeDatabase

Это облачная NoSQL БД. Данные хранятся в формате JSON-дерева и синхронизируются в режиме реального времени с каждым подключенным клиентом. Также, эта БД имеет свой собственный «язык правил» Firebase Realtime Database Security Rules, который позволяет определять как данные будут структурироваться и кто сможет их читать/изменять.

СУБД SQLite

СУБД SQLite – это компактная встраиваемая СУБД. «Встраиваемый» означает, что SQLite не использует парадигму клиент-сервер, то есть движок SQLite не является отдельно работающим процессом, с которым взаимодействует программа, а представляет собой библиотеку, с которой программа компонуется, и движок становится составной частью программы.

Таким образом, в качестве протокола обмена используются вызовы функций (API) библиотеки SQLite. Такой подход уменьшает накладные

					ПГУ 1.090301.01.001 ПЗ	Лист
						13
Изм.	Лист	№ докум.	Подпись	Дата		

расходы, время отклика и упрощает программу. SQLite хранит всю базу данных (включая определения, таблицы, индексы и данные) в единственном стандартном файле на том компьютере, на котором выполняется программа. Простота реализации достигается за счёт того, что перед началом исполнения транзакции записи весь файл, хранящий базу данных, блокируется.

Несколько процессов или потоков могут одновременно без каких-либо проблем читать данные из одной базы. Запись в базу можно осуществить только в том случае, если никаких других запросов в данный момент не обслуживается; в противном случае попытка записи оканчивается неудачей, и в программу возвращается код ошибки.

Android Studio

Android Studio – это интегрированная среда разработки для работы с платформой Android. Android Studio, основанная на программном обеспечении IntelliJ IDEA от компании JetBrains, - официальное средство разработки Android приложений.

С каждой версией функционал Android Studio растёт и улучшается, позволяя упрощать разработку приложений для Android. На данный момент доступно огромное количество функций, например, следующие:

- Расширенный редактор макетов: WYSIWYG, способность работать с UI компонентами при помощи Drag-and-Drop, функция предпросмотра макета на нескольких конфигурациях экрана.
- Сборка приложений, основанная на Gradle.
- Различные виды сборок и генерация нескольких .apk файлов
- Статический анализатор кода (Lint), позволяющий находить проблемы производительности, несовместимости версий и другое.
- Встроенный ProGuard и утилита для подписывания приложений.

					ПГУ 1.090301.01.001 ПЗ	Лист
						14
Изм.	Лист	№ докум.	Подпись	Дата		

2 ОПИСАНИЕ ТРЕБОВАНИЙ К СИСТЕМЕ

2.1 Требования к надежности

Основными требованиями к надёжности системы являются: бесперебойное функционирование системы и защита от неправильного ввода данных в систему.

В системе должен быть реализован контроль над корректностью вводимой информации. Ввод данных в информационную систему должен осуществляться только после исправления пользователем ошибок ввода. При вводе новой информации в случае ошибки во входных параметрах, должны выдаваться соответствующие сообщения и отменяться операция ввода с целью сохранения целостности базы данных. Система не должна допускать дублирования ввода уже имеющейся информации и фиксировать информацию, пока она не введена по всем полям в базе данных.

Также, к требованиям надежности относятся правила ввода регистрационных данных. Хотя и большую часть на себя берет система Firebase Authentication, но на клиентской стороне также должна быть реализована проверка ввода электронной почты и пароля, например, электронной почтой будет считаться только строка, заканчивающаяся идентификаторами основных поставщиков электронной почты - @gmail.com, @mail.ru, @yandex.ru и т.д.

2.2 Требования к пользователю ИС

Данной информационной системой может пользоваться даже не уверенный пользователь устройств смартфонов, так как у системы простой и понятный интерфейс, простота в вводе данных и их использовании. Система предназначена как для профессиональных спортсменов, так и для любителей.

					ПГУ 1.090301.01.001 ПЗ	Лист
						15
Изм.	Лист	№ докум.	Подпись	Дата		

2.3 Требования к информационной и программной совместимости

Информационная система контроля тренировок спортсменов должна быть реализована в среде разработки Android Studio на языке программирования Java.

На рабочих станциях сотрудников должно быть установлено:

- Операционная система Android не ниже версии 4.4 “KitKat”
- Google Play Services

2.4 Требования к программной документации

В состав пояснительной записки дипломного проекта должны быть включены следующие разделы и документы:

- техническое задание;
- описание программы;
- программа и методика испытаний;
- описание применения.

					ПГУ 1.090301.01.001 ПЗ	Лист
						16
Изм.	Лист	№ докум.	Подпись	Дата		

3 ПРОЕКТИРОВАНИЕ ПРИЛОЖЕНИЯ

В ходе реализации Android приложения, изначально была выбрана архитектура, на основе которой будет строить приложение. Грамотно выбранная архитектура позволит в дальнейшем «безболезненно» вносить изменения, добавляя или улучшая функционал.

В данной работе будет использоваться архитектура MVVM (Model-View-ViewModel). Его концепция заключается в отделении логики представления данных приложения, пользовательского интерфейса и управляющей логики на три отдельных компонента:

- Модель (Model) - это логика, которая связана с данными приложения. Другими словами - это классы работы с API, базой данных и пр.
- Представление (View) - отвечает за отображение данных модели пользователю, реагируя на изменения модели.
- МодельПредставление (ViewModel) - это объект, в котором описывается логика поведения View в зависимости от результата работы Model. Можно назвать его моделью поведения View. Это может быть как форматирование текста, так и логика управления видимостью компонентов или отображения состояний, таких как загрузка, ошибка, пустые экраны и т.д. Также в ней описывается поведение, которое было инициировано пользователем (ввод текста, нажатие на кнопку, свайп и т.п.).

На рисунке 1 представлена примерная схема MVVM архитектуры приложения.

					ПГУ 1.090301.01.001 ПЗ	Лист
						17
Изм.	Лист	№ докум.	Подпись	Дата		

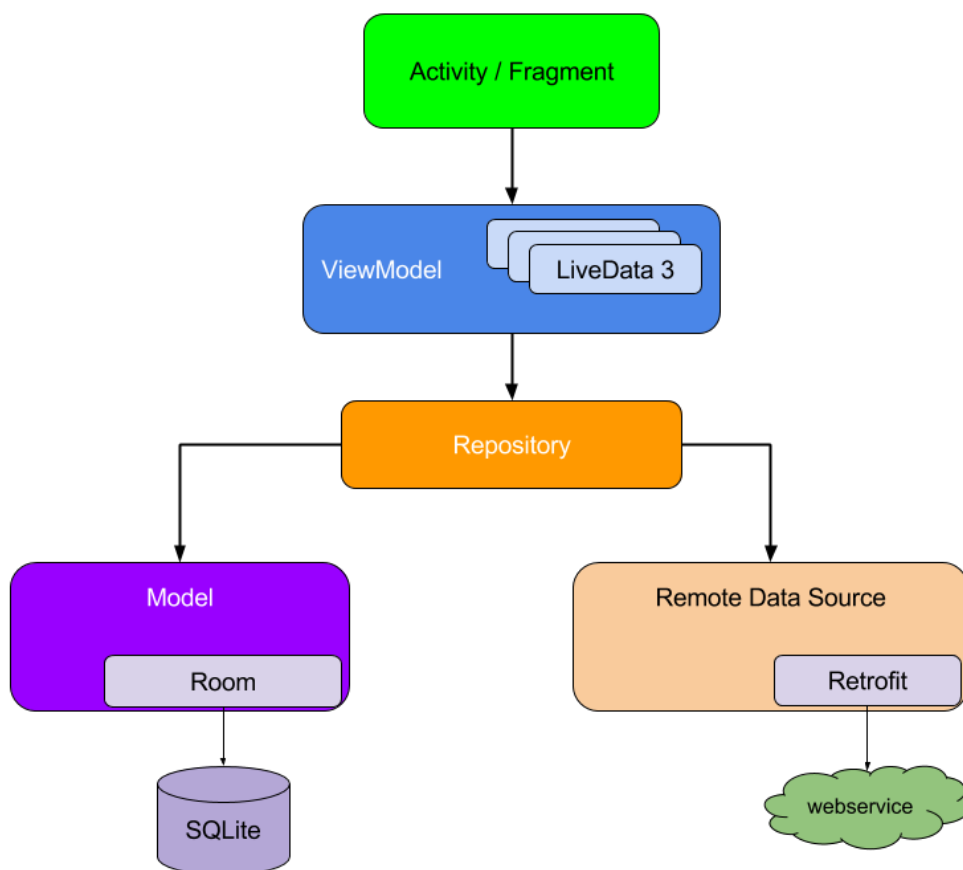


Рисунок 1 - Схема MVVM

Что же это дает нам в конечном итоге?

- Гибкость разработки;
- Тестирование;
- Разграничение логики.

Рассмотри схему архитектуры MVVM более детально:

1. **Activity/Fragment.** Это по сути и есть Представление (View) схемы. Здесь определяются способы отображения данных в тех или иных элементах пользовательского интерфейса. Также определяются и вызываются необходимые методы ViewModel для изменения или получения необходимых данных модели (Model).
2. **ViewModel.** Это и есть МодельПредставление (ViewModel) схемы. Задача этого класса в том, чтобы предоставлять необходимые данные для UI приложения. Данные должны быть актуальными. Также в

задачи этого класса входит определение того, как эти данные модели изменить в ответ на действия пользователя в UI. При помощи архитектурного компонента LiveData – ViewModel может быть получена в любом классе пользовательского интерфейса, что в дальнейшем позволит определить нужные запросы, которые будут при помощи LiveData вызываться всякий раз, когда локальная БД изменится.

3. **Repository.** Это обще хранилище всех данных в приложении. Здесь хранится вся информация как о локальном хранилище данных, так и о серверном. Также, именно здесь определяется логика получения эти данных из локальной и серверной баз данных.
4. **Model.** Это и есть Модель (Model) из схемы MVVM. Это и есть то, с чем пользователь взаимодействует. В приложении, например моделью является объект «Пользователь», «План», «Неделя», «День» и т.д. Именно объекты модели хранятся в базах данных.

В приложении используются локальная и серверная БД.

Локальная – это RoomDatabase, по сути, «обертка» для работы с локальной встроенной БД – SQLite. Это классическая реляционная SQL БД.

Серверная – NoSQL БД вида JSON-дерева Firebase RealtimeDatabase.

Приложение должно обладать интуитивно понятным интерфейсом и позволять производить различные действия над базами данных: получение данных из серверной БД, запись их в локальную БД или добавление новых записей в локальную БД и их синхронизация с серверной.

Запросы к локальной базе данных SQLite должны конструироваться на основании того или иного действия пользователя и представлять из себя команды на языке SQL.

					ПГУ 1.090301.01.001 ПЗ	Лист
						19
Изм.	Лист	№ докум.	Подпись	Дата		

Запросы к серверной БД представляют собой запросы вида «ключ:значение». Сама БД по структуре – это JSON дерево, перемещаясь по узлам которого (аналог таблиц в реляционной структуре) можно получить необходимые данные.

Возможные варианты взаимодействия пользователя с базами данных представлены на UML – диаграмме вариантов использования, показано в приложении Б рисунок 19.

Для реализации этих возможностей необходимо будет разместить в окне приложения кнопки, меню, поля ввода или предусмотреть иные интуитивно понятные способы осуществления указанных действий.

Действия пользователя должны вызывать соответствующие обработчики. Поскольку выбранная локальная СУБД поддерживает возможность взаимодействия с базой данных посредством выполнения запросов на языке SQL, в обработчиках должны конструироваться команды, в которые подставляются названия таблиц и/или столбцов в соответствии с синтаксисом той или иной конкретной команды.

Работа с серверной NoSQL БД очень схожа по работе с обычной реляционной БД, только вместо обращения к таблиц здесь происходит обращение к узлам JSON дерева, а для получения данных нужно воспользоваться запросом вида «ключ:значение».

Схема алгоритма обработки запросов, представлено в приложении Б на рисунке 19. Отправляемые приложением запросы и обработка результата можно изобразить с помощью UML – диаграммы состояний, представлено в приложении Б на рисунке 20.

4 РЕАЛИЗАЦИЯ ПРОЕКТА

4.1 Процесс разработки

4.1.1. Реализация Аутентификации пользователей

Пользователь при регистрации должен ввести минимум информации. Во-первых, это конечно e-mail и пароль, по которым в дальнейшем пользователь будет осуществлять вход в приложение. Также потребуется ввести имя, но это как дополнительная опция, которая не требуется самой системой авторизации Firebase Authentication.

При первом входе, у пользователя создается уникальный идентификатор, и ставятся в соответствие остальные регистрационные данные, в локальной и серверной базах создаются соответствующие записи. При чем пароль, не хранится как таковой, а хранится только его уникальный хэш. На рисунке 2 представлен скриншот параметров хэширования паролей.

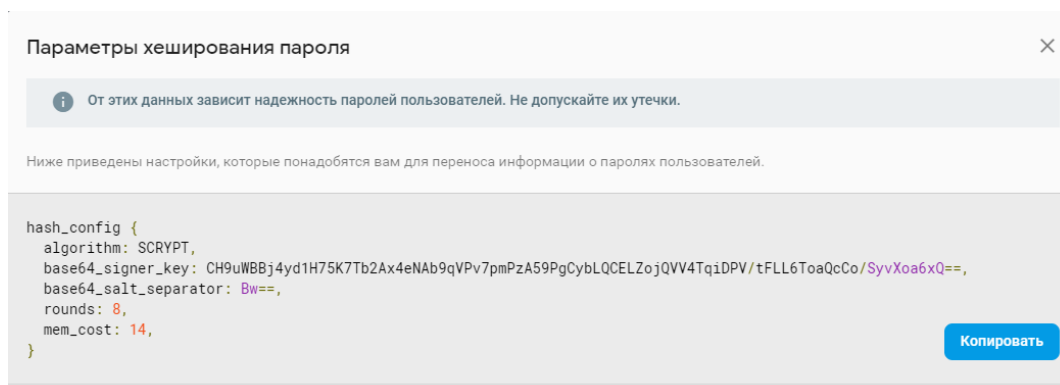


Рисунок 2 – Параметры хэширования паролей

Зарегистрировавшись, пользователь также получает свой уникальный идентификатор, который в дальнейшем понадобится для хранения и идентификации всех записей пользователя в базах данных. На рисунке 3 представлена информация о хранения пользовательских данных на сервере.

					ПГУ 1.090301.01.001 ПЗ	Лист
						21
Изм.	Лист	№ докум.	Подпись	Дата		

<div> <div> <div>🔍</div> <div>Эл. почта, телефон или идентификатор пользователя</div> </div> <div> <div>Добавить пользователя</div> <div>↺</div> <div>⋮</div> </div> </div>				
Идентификатор	Поставщики	Время создания	Последний вход	Уникальный идентификатор пользователя ↑
a.sergey9758@gmail.com	✉	24 июн. 2019 г.	27 июн. 2019 г.	zjz7GXuGrZdl74ITbIS00sH1btj2
<div> <div>Количество строк на странице: 50</div> <div>1-1 из 1</div> <div>⏪</div> <div>⏩</div> </div>				

Рисунок 3 – Хранение данных пользователя

Далее, локально, на устройстве создается файл – авторизационный токен пользователя, который в дальнейшем, на определенный промежуток времени позволит пользователю аутентифицироваться в приложении без ввода e-mail и пароля. На рисунке 4 представлен скриншот файла и его содержимого локально на устройстве пользователя.

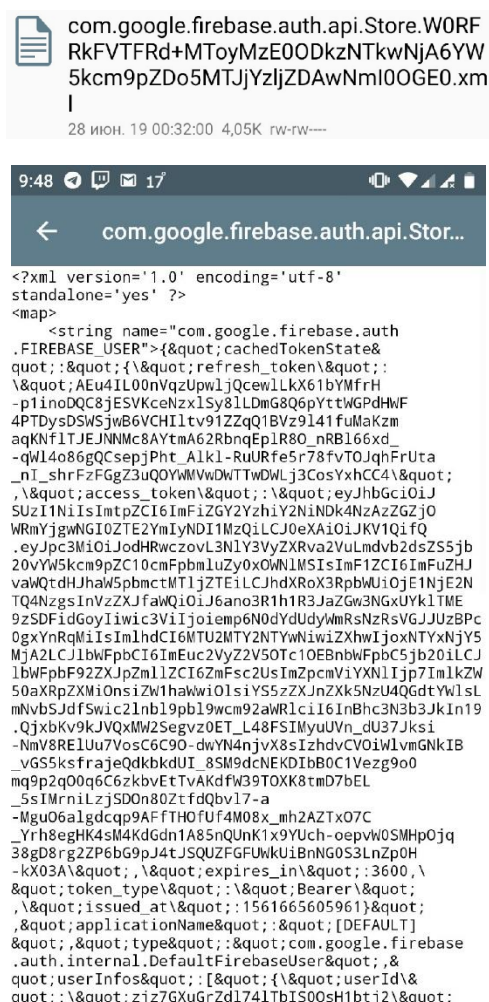


Рисунок 4 – Авторизационный токен пользователя и его содержимое

					ПГУ 1.090301.01.001 ПЗ	Лист
						22
Изм.	Лист	№ докум.	Подпись	Дата		

4.1.2 Реализация БД

Необходимо спроектировать базу данных для хранения информации о планах, неделях и днях тренировок. Также, так как в приложении используются как локальная, так и серверная базы данных, при чем одна из них классическая реляционная SQL – SQLite, а другая – NoSQL вида JSON-деревя, то соответственно нужно продумать как будут храниться и какие именно поля в одной БД и в другой. А также, как они будут взаимодействовать друг с другом, то есть, как, например, при синхронизации данных, с клиента будут отправляться данные на сервер или наоборот, с сервера приходить на клиент.

В таблице 1 представлено описание типов сущностей. Эти сущности будут присутствовать в обеих БД, также, как и все поля из описания, различия будут только в виде хранения этих данных.

Таблица 1 – Описание типов сущностей

№	Наименование	Описание
1	Планы	id, количество недель, количество дней, статус, дата старта, дата завершения
2	Недели	id, id плана, количество дней, номер недели, статус
3	Дни	id, id недели, номер дня, дата дня, тип тренировки, статус
4	Беговые тренировки	id, id дня, количество километров, средняя скорость
5	Упражнения	id, id дня, количество упражнений, время тренировки
6	Пользователь	id, имейл, Имя, Фамилия, вес, рост, дата рождения

В таблице 2 представлены связи между сущностями в схеме локальной SQLite БД.

Таблица 2 – Описание типов связей между типами сущностей предметной области

№	Сущность 1	Сущность 2	Описание	Мощность	Обязательность	Ограничения
1	План	Недели	У плана несколько недель	1:M	Обязательно	
2	Недели	Дни	У недели несколько дней	1:M	Обязательно	Одна и та же тренировка не может быть в Беговой и в Упражнениях Одна и та же тренировка не может быть в Беговой и в Упражнениях
3	Дни	Беговая тренировка	Определенный уникальный день связан с уникальной информацией о нем	1:1	Обязательно	
4	Дни	Упражнения	Определенный уникальный день связан с уникальной информацией о нем	1:1	Обязательно	

В то же время, в серверной данной все связи будут в виде «ключ:значение».

На рисунке 5 представлена схема хранения NoSQL данных в виде JSON-дерева

```

1  {
2    "zjz7GXuGrZdl741TbIS00sH1btj2" : {
3      "days" : {
4        "-LiCNinyaTTX76qdbdnG" : {
5          "-LiCNio4swGVWjMC1DTx" : {
6            "dayNumber" : 1,
7            "trainingDayType" : "Rest"
8          },
9          ...
10         },
11         "-LiCNio23uREnMen-80p" : {
12           "-LiCNioLn-lonsSUVXk6" : {
13             "dayNumber" : 8,
14             "trainingDayType" : "Gym"
15           },
16           ...
17         },
18       "plans" : {
19         "-LiCNinbdiSfZtc5Qi5L" : {
20           "allTrainingDaysCount" : 8,
21           "planStatus" : "CREATED",
22           "startDate" : "1561445198184",
23           "weeksCount" : 2
24         },
25       },
26       "weeks" : {
27         "-LiCNinbdiSfZtc5Qi5L" : {
28           "-LiCNinyaTTX76qdbdnG" : {
29             "trainingDaysCount" : 0,
30             "weekNumber" : 1,
31             "weekStatus" : "CREATED"
32           },
33           "-LiCNio23uREnMen-80p" : {
34             "trainingDaysCount" : 0,
35             "weekNumber" : 2,
36             "weekStatus" : "CREATED"
37           }
38         },
39       },
40     },
41   }

```

Рисунок 5 – Хранение данных на сервере

4.1.3 Реализация интерфейса приложения

В приложении были реализованы следующие макеты экранов:

- Макет экрана авторизации пользователей
- Splash-screen, отображающийся при запуске приложения
- Прототип макета экрана, необходимый для тестирования работоспособности БД, отображающий выбранную запись из запроса

Макет экрана авторизации пользователей. Существует две ситуации исходя из которых пользователь может попасть на этот экран. Первая – это когда пользователь заходит в приложение впервые: ему сразу же предлагается пройти краткий экскурс по возможностям предоставляемым приложением. Пользователь может как пропустить этот экран, так и пролистать все полезные руководства. В любом случае, после экрана руководства, представленного на рисунке 1, пользователю будет предложено пройти авторизацию (зарегистрироваться, если у пользователя еще нет аккаунта или войти в уже существующий аккаунт).



Рисунок 2 - Экран руководства в приложении

					ПГУ 1.090301.01.001 ПЗ	Лист
						26
Изм.	Лист	№ докум.	Подпись	Дата		

На экране авторизации, представленном на рисунке 2 пользователю нужно будет ввести логин и пароль и выбрать кнопку «Создать аккаунт» или «Войти», в зависимости от необходимого действия пользователя.

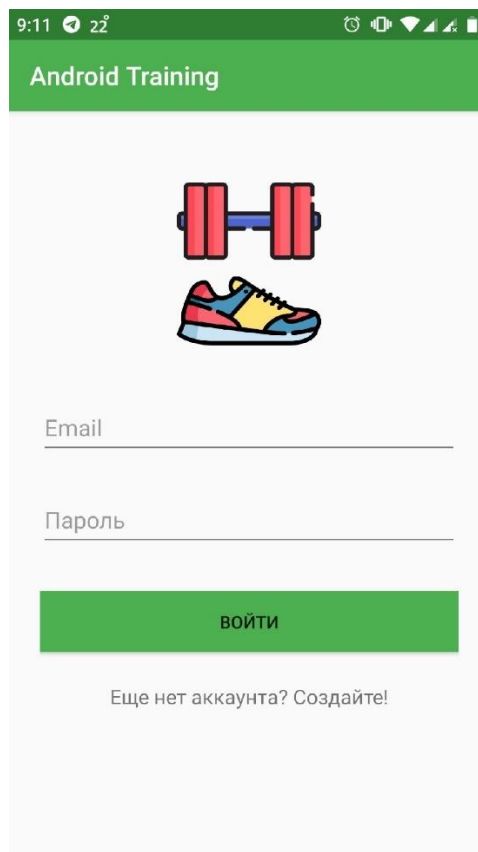


Рисунок 3 - Экран авторизации в приложении

Стоит отметить, что при вводе пользователем пароля, символы постепенно закрашиваются точками, что необходимо для безопасности.

					ПГУ 1.090301.01.001 ПЗ	Лист
						27
Изм.	Лист	№ докум.	Подпись	Дата		

Рисунок 4 - Экран регистрации в приложении

Splash-screen. Splash-screen представляет собой обычный экран-«заглушку», который отображается при запуске приложения. Когда пользователь запускает приложение, что называется «на холодную», то есть процесса для этого приложения еще не было создано и никаких данных приложения в оперативной памяти устройства не содержится. В таком случае, промежуток времени между нажатием пользователем на иконку приложения в лаунчере, да момента показа ему экранов самого приложения может пройти некоторое время (возможно секунда или две) и чтобы пользователю было понятно, что система не «повисла», а приложение запускается и отображается этот экран. Макет этого экрана представлен на рисунке 3. На данный момент это прототип и со временем он будет дорабатываться (желательно, должен быть схож со стилем приложения).

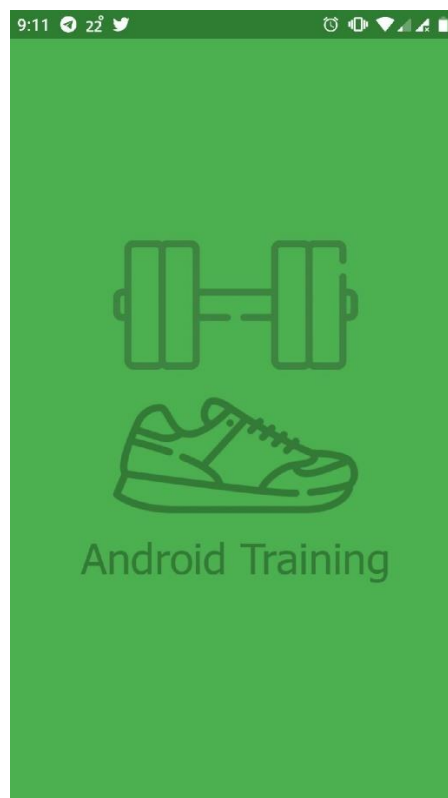


Рисунок 5 - Splash-screen, отображаемый при запуске приложения

Прототип макета экрана, необходимого для тестирования работоспособности БД. В данном случае – это обычный макет, с текстовыми полями, которые отображают значения столбцов записи, выбранной из таблица по определенному условию. Макет представлен на рисунке 6.

					ПГУ 1.090301.01.001 ПЗ	Лист
						29
Изм.	Лист	№ докум.	Подпись	Дата		

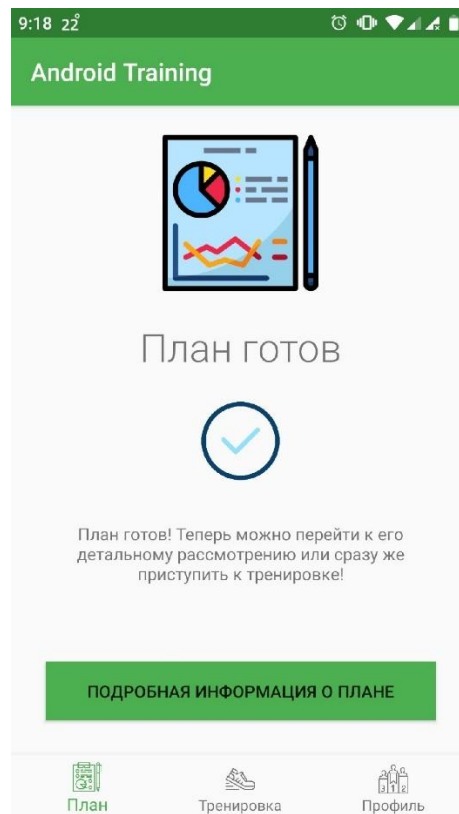


Рисунок 6 - Экран для тестирования локальной и серверной БД

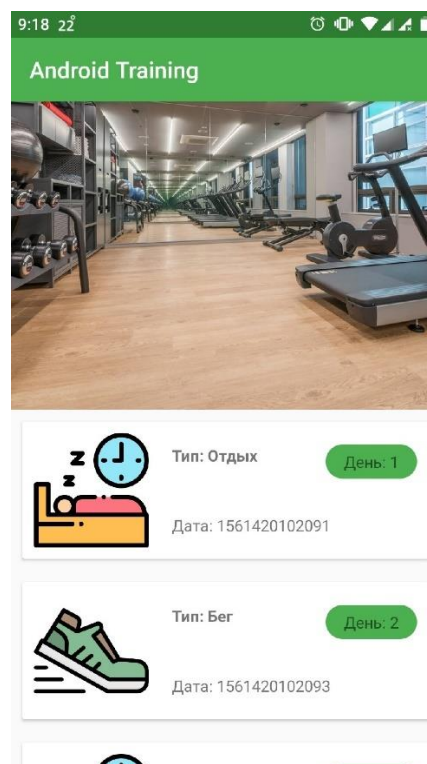


Рисунок 7 - Экран для тестирования локальной и серверной БД

					ПГУ 1.090301.01.001 ПЗ	Лист
						30
Изм.	Лист	№ докум.	Подпись	Дата		

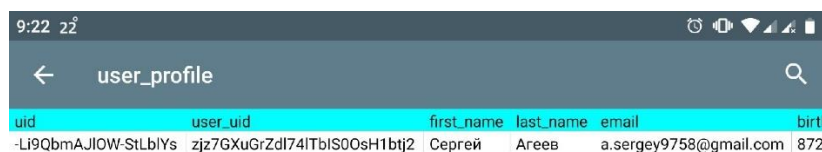
4.2 Отладка и тестирование

Программа написана и отлажена с использованием среды программирования Android Studio.

Отладка проводилась с помощью таких инструментов, как точка останова, выполнение кода по шагам, анализа содержимого переменных.

Тестирование проводилось в рабочем порядке, в процессе разработки, после завершения написания программы. В ходе тестирования было выявлено и исправлено множество проблем, связанных с синтаксисом конструируемых в программе SQL-запросов, взаимодействием приложения с базой данных, а также непосредственным доступом к базе данных.

Данные в локальной БД совпадают с данными из БД сервера. При удалении БД локально, при перезапуске приложение автоматически запросит с сервера данные и запишет их в локальную БД. Таким образом, данные не будут теряться.



The screenshot shows an Android application interface with a title bar 'user_profile' and a search icon. Below the title bar is a table with the following data:

uid	user_uid	first_name	last_name	email	birth
-Li9QbmAJlOW-StLbIYs	zjz7GXuGrZdl74ITbIS00sH1btj2	Сергей	Ареев	a.sergey9758@gmail.com	872

Рисунок 8 – Тест локальной БД по таблице «Профили»

uid	weeks_count	all_training_days_count	start_date	finish_date	plan_status
-LiA6vmylZGhhfhsJLTE	2	5	156140724046e	156140724046e	CREATED

Рисунок 9 – Тест локальной БД по таблице «Планы»

uid	plan_uid	training_days_count	week_number	week_status
-LiA6vn59sbbylKcX_Tp	-LiA6vmylZGhhfhsJLTE	0	1	CREATED
-LiA6vnA_hRTGCZpB8Xe	-LiA6vmylZGhhfhsJLTE	0	2	CREATED

Рисунок 10 – Тест локальной БД по таблице «Недели»

uid	week_uid	day_number	day_date	training_day_type	day_status
-LiA6vnDi5MvFKcQHDGw	-LiA6vn59sbbylKcX_Tp	1	1561420102091	REST	CREATED
-LiA6vnI5loKi4zQUyrt	-LiA6vn59sbbylKcX_Tp	2	1561420102093	Run	CREATED
-LiA6vnKwtRn_Ubnc5B0	-LiA6vn59sbbylKcX_Tp	3	1561420102094	REST	CREATED
-LiA6vnM6H-AD67eHq85	-LiA6vn59sbbylKcX_Tp	4	1561420102094	Gym	CREATED
-LiA6vnOZJRJRHx0a38C6	-LiA6vn59sbbylKcX_Tp	5	1561420102095	REST	CREATED
-LiA6vnPMEvJwUksTGx0	-LiA6vn59sbbylKcX_Tp	6	1561420102096	REST	CREATED
-LiA6vnRvxCOwzV40KTZ	-LiA6vn59sbbylKcX_Tp	7	1561420102097	REST	CREATED
-LiA6vnTVc1iVehAnulO	-LiA6vnA_hRTGCZpB8Xe	8	1561420102102	Run	CREATED
-LiA6vnVCvtB246gh4PD	-LiA6vnA_hRTGCZpB8Xe	9	1561420102103	REST	CREATED
-LiA6vnWu3xxl2g5_87f	-LiA6vnA_hRTGCZpB8Xe	10	1561420102104	Run	CREATED
-LiA6vnYC3f89hN8DI3K	-LiA6vnA_hRTGCZpB8Xe	11	1561420102105	REST	CREATED
-LiA6vn_z9DScTOL85_V	-LiA6vnA_hRTGCZpB8Xe	12	1561420102105	Gym	CREATED
-LiA6vnaAHIGB5kp1_KC	-LiA6vnA_hRTGCZpB8Xe	13	1561420102106	REST	CREATED
-LiA6vncztGPOEayNeTo	-LiA6vnA_hRTGCZpB8Xe	14	1561420102107	REST	CREATED

Рисунок 11 – Тест локальной БД по таблице «Дни»

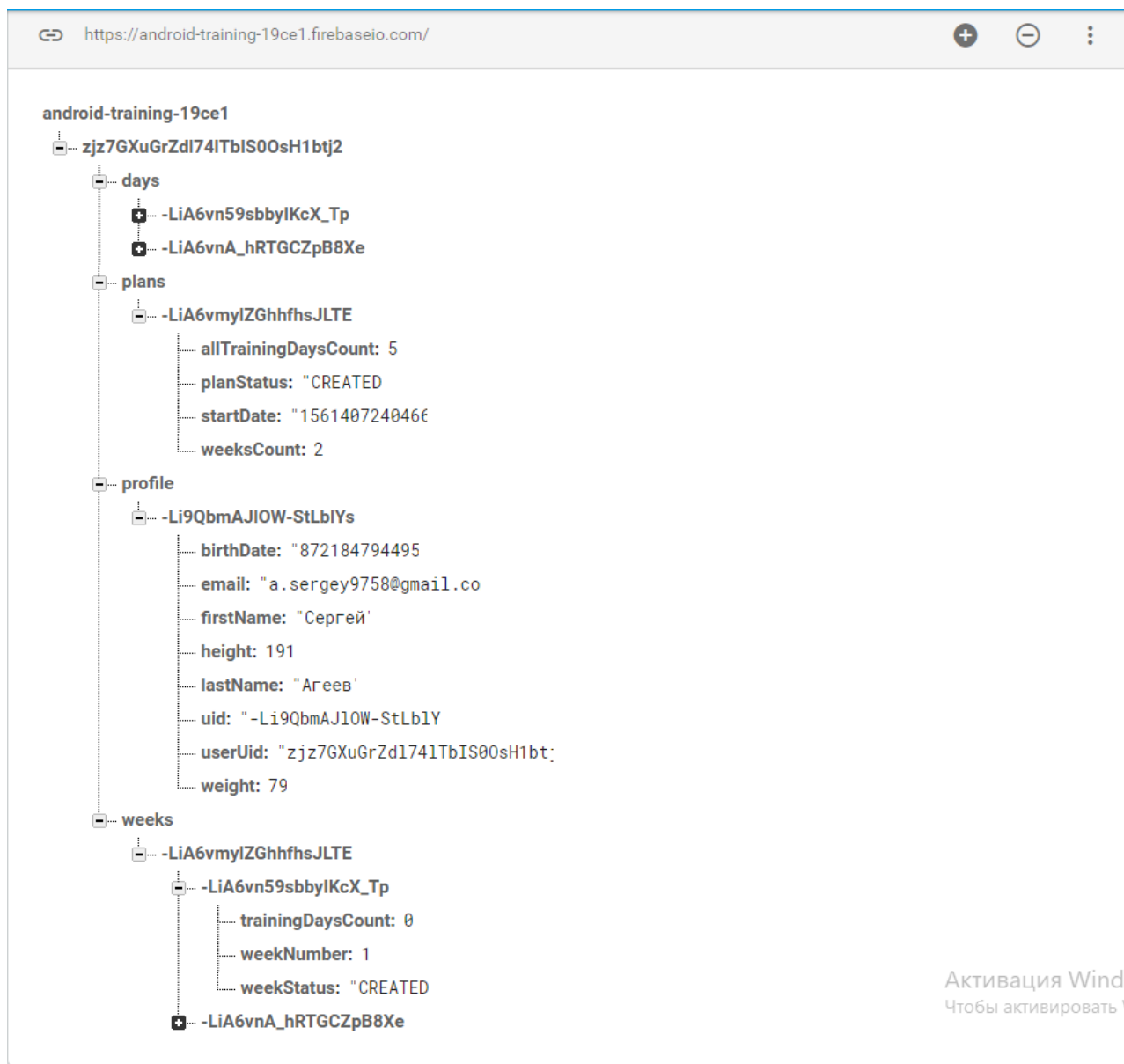


Рисунок 12 – Тест серверная БД Firebase RealtimeDatabase

5 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

Работа приложения начинается с открытия окна авторизации. Если приложение скачал «новый» спортсмен, его данные для входа можно зарегистрировать через приложение, нажав на кнопку регистрации. После ввода данных если все введено корректно, новый пользователь будет создан и вся информация сохранится в базе данных на клиенте и на сервере. Вид окна с примером заполнения полей представлен на рисунке 13.

Рисунок 13 – Заполненные поля регистрации

					ПГУ 1.090301.01.001 ПЗ	Лист
						34
Изм.	Лист	№ докум.	Подпись	Дата		

Рассмотрим основные возможности, которые представляет приложение. В нижней части макета окна расположено меню с вкладками:

- План. Данная страница необходима для просмотра детальной информации о планах.
- Тренировка. На данной вкладке можно посмотреть текущую тренировку и начать ее выполнять.
- Профиль. Представлена информация о профиле авторизованного пользователя.

Основная работа с БД происходит на странице «План». Когда плана не существует локально, то приложение пытается загрузить его из сети, если его нет и в сети, то пользователю предлагается возможность создать план самому, настроив все необходимые параметры под себя



Рисунок 14 – Макет экрана создания плана

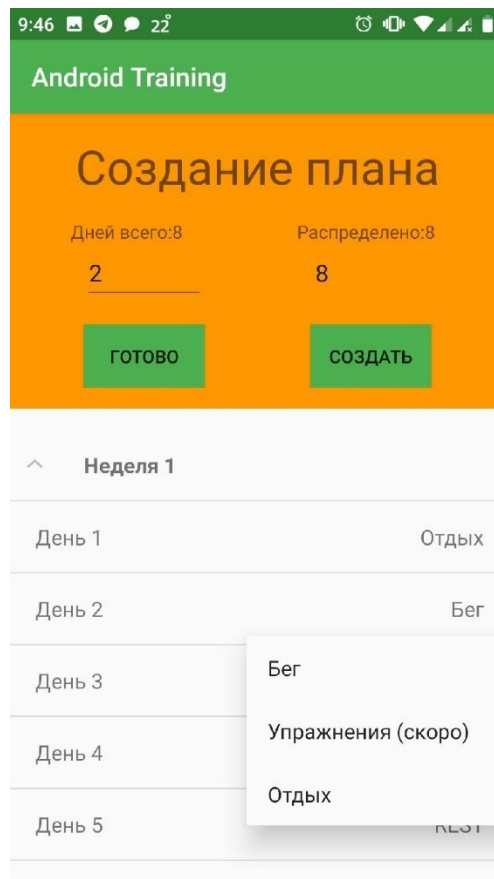


Рисунок 15 – Макет экрана создания плана

Контекстное меню позволяет выбрать необходимый тип тренировки.

После создания плана, пользователь может выбрать пункт «Тренировка» в меню снизу и затем ему будет выведена вся информация о текущей тренировке, например, при дне «Упражнений» пользователю будет предоставлен следующий экран, представленный на рисунке 16.

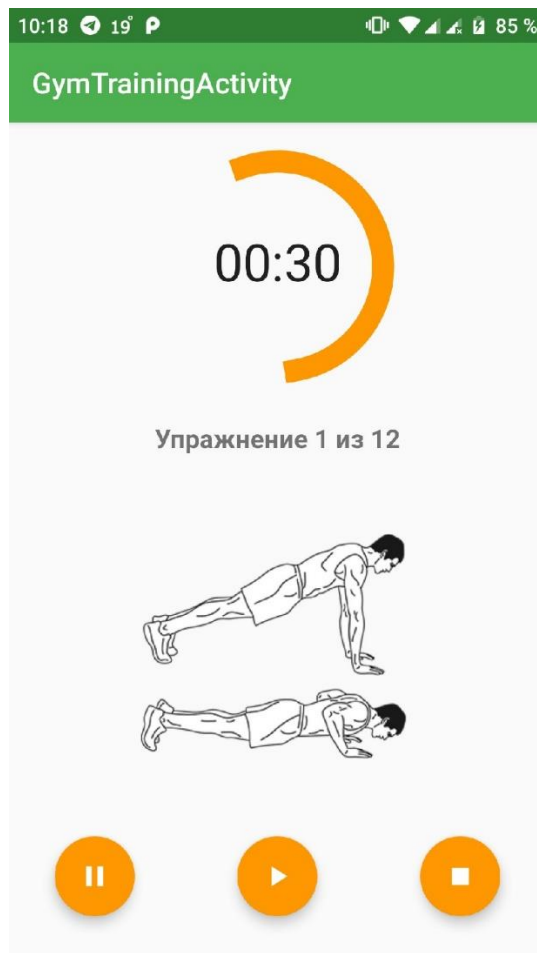


Рисунок 17 – Макет экрана «Профиль»

Пользователю показывается номер упражнения, которое ему необходимо выполнить, кнопки «Старт», «Стоп», «Пауза» и таймер. Таймер – это время выполнения текущего упражнения. Кнопка «Старт» запускает тренировку, «Пауза» приостанавливает, позволяя сделать перерыв, а «Стоп» завершает тренировку полностью и возвращается на предыдущий экран.

					ПГУ 1.090301.01.001 ПЗ	Лист
						37
Изм.	Лист	№ докум.	Подпись	Дата		

Также, пользователь может нажать на кнопку меню «Профиль» чтобы просмотреть все свои введенные регистрационные данные и, возможно, добавить еще несколько, например рост, вес, фамилию и год рождения:

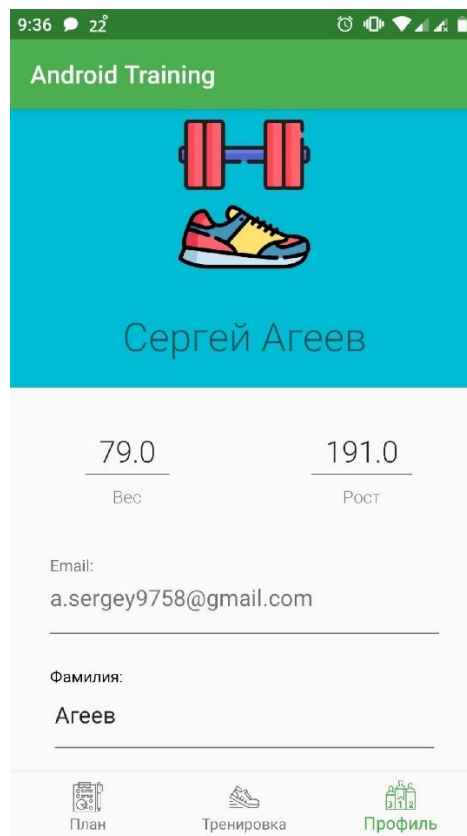


Рисунок 17 – Макет экрана «Профиль»

					ПГУ 1.090301.01.001 ПЗ	Лист
						38
Изм.	Лист	№ докум.	Подпись	Дата		

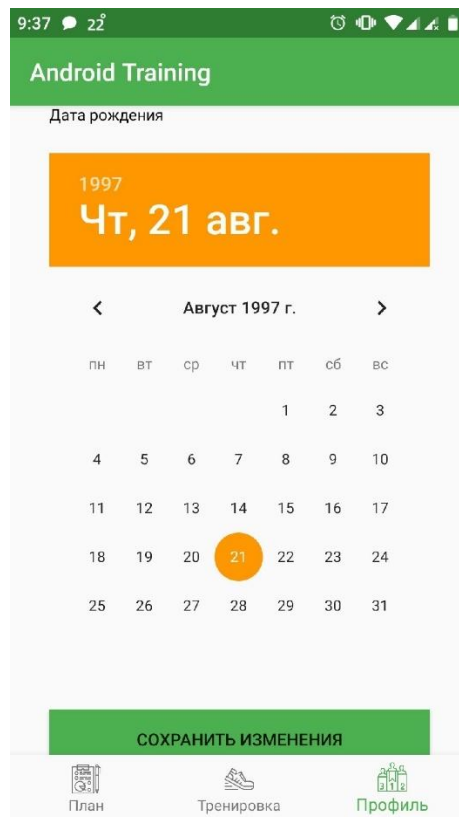


Рисунок 18 – Макет экрана «Профиль»

ЗАКЛЮЧЕНИЕ

Результатом работы является клиент-серверное приложение с интуитивно понятным пользовательским интерфейсом. Приложение позволяет работать с базой данных планов, недель и тренировочных дней. Данный проект полностью удовлетворяет поставленной цели.

Данная информационная система позволяет пользователю беспрепятственно вести учет тренировок по своему собственному плану. Данные вводятся самим пользователем и визуально отображаются в виде обычных числовых данных в таблицах, полях ввода и т.д.

Отладка и тестирование состоит из проверки синхронизации серверной NoSQL Firebase RealtimeDatabase вместе с локальной SQLite в обертке Room.

Работа над проектом позволила расширить знания и улучшить навыки создания клиентских Android приложений и работы с базами данных.

Программа может быть подвержена апгрейду в дальнейшем, в соответствии с потребностями пользователей, например, можно добавить большее количество упражнений, собирать больше информации во время бега, добавить поддержку фитнес браслетов.

					ПГУ 1.090301.01.001 ПЗ	Лист
						40
Изм.	Лист	№ докум.	Подпись	Дата		

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Дейтел П., Дейтел Х., Уолд А. Android для разработчиков. 3-е изд. – СПб.: Питер, 2017.
2. Мартин Р. Чистый Код: создание, анализ и рефакторинг. Пер. с англ. – М.: Издательский дом “Питер”, 2018.
3. Гамма Э., Хелм., Джонсон Р., Влиссидес.. Приемы объектно-ориентированного проектирования. Паттерны проектирования. – СПб.: Питер, 2018.
4. Пушников А. Ю. Введение в системы управления базами данных. [Электронный ресурс] - <http://www.citforum.ru/database/dblearn/index.shtml>
5. Дейт К. Дж. Введение в системы баз данных – М.: Издательский дом “Вильямс”, 2001.
6. Файлер М., Садаладж П. Дж.. NoSQL. Новая методология разработки нереляционных данных. Пер. с англ. – М.: Издательский дом “Вильямс”, 2018
7. Firebase RealtimeDatabase [Электронный ресурс] <https://firebase.google.com/docs/database>
8. Medium – Android Developers [Электронный ресурс] – <https://medium.com/androiddevelopers>
9. Герберт Шилдг. Java руководство для начинающих.:Пер. с англ. — М. : Издательский дом "Вильямс", 2015.
10. Шилдт Герберт- Java 8 Полное руководство. 2015.
11. Официальная документация по Android-разработке [Электронный ресурс] – <https://developer.android.com/>

ПРИЛОЖЕНИЕ А
ЛИСТИНГ ПРОГРАММЫ

					ПГУ 1.090301.01.001 ПЗ	Лист
						42
Изм.	Лист	№ докум.	Подпись	Дата		

Приложение А.1 Файл LoginActivity.java

```
package com.example.android.androidtraining.authentication;

import android.app.ProgressDialog;
import android.content.Intent;
import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.annotation.VisibleForTesting;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.text.TextUtils;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;

import com.example.android.androidtraining.R;
import com.example.android.androidtraining.main.MainActivity;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.android.material.snackbar.Snackbar;
import com.google.firebase.auth.AuthResult;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;

public class LoginActivity extends AppCompatActivity implements View.OnClickListener {

    private static final String TAG = LoginActivity.class.getSimpleName();
    private static final int REQUEST_REGISTER = 0;

    private EditText mEmailEditField;
    private EditText mPasswordEditField;

    private Button mLoginButton;

    private TextView mRegisterLink;

    @VisibleForTesting
    public ProgressDialog mProgressDialog;

    private FirebaseAuth mAuth;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_login);

        mEmailEditField = (EditText) findViewById(R.id.emailEditField);
        mPasswordEditField = (EditText) findViewById(R.id.passwordEditField);

        mLoginButton = (Button) findViewById(R.id.loginButton);
        mLoginButton.setOnClickListener(this);

        mRegisterLink = (TextView) findViewById(R.id.registerLink);
        mRegisterLink.setOnClickListener(this);

        mAuth = FirebaseAuth.getInstance();
    }

    @Override
    protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data)
    {
        super.onActivityResult(requestCode, resultCode, data);
        if (requestCode == REQUEST_REGISTER) {
```

					ПГУ 1.090301.01.001 ПЗ	Лист
						43
Изм.	Лист	№ докум.	Подпись	Дата		

```

        if (resultCode == RESULT_OK) {
            }
        }
    }

private void login(String email, String password) {
    if (!validateForm()) {
        return;
    }
    showProgressDialog();
    mAuth.signInWithEmailAndPassword(email, password).addOnCompleteListener(this,
        new OnCompleteListener<AuthResult>() {
            @Override
            public void onComplete(@NonNull Task<AuthResult> task) {
                if (task.isSuccessful()) {
                    FirebaseUser user = mAuth.getCurrentUser();
                    updateUI(user);
                } else {
                    updateUI(null);
                }
                hideProgressDialog();
            }
        });
}

private boolean validateForm() {
    boolean valid = true;

    String email = mEmailEditField.getText().toString();
    if (TextUtils.isEmpty(email)) {
        mEmailEditField.setError(getString(R.string.enter_email_hint));
        valid = false;
    } else {
        mEmailEditField.setError(null);
    }

    String password = mPasswordEditField.getText().toString();
    if (TextUtils.isEmpty(password)) {
        mPasswordEditField.setError(getString(R.string.enter_password_hint));
        valid = false;
    } else {
        mPasswordEditField.setError(null);
    }

    return valid;
}

private void updateUI(FirebaseUser user) {
    hideProgressDialog();
    if (user != null) {
        Intent intent = new Intent(LoginActivity.this, MainActivity.class);
        intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
        startActivity(intent);
        finish();
        return;
    } else {
        Snackbar.make(mLoginButton, "Не удалось войти!", Snackbar.LENGTH_SHORT).show();
    }
}

@Override
public void onClick(View v) {
    switch (v.getId()) {
        case R.id.loginButton:
            login(mEmailEditField.getText().toString(),

```

					ПГУ 1.090301.01.001 ПЗ	Лист
						44
Изм.	Лист	№ докум.	Подпись	Дата		

```

mPasswordEditField.getText().toString());
        break;
        case R.id.registerLink:
            Intent intent = new Intent(LoginActivity.this, RegisterActivity.class);
            startActivityForResult(intent, REQUEST_REGISTER);
            break;
        default:
            break;
    }
}

public void showProgressDialog() {
    if (mProgressDialog == null) {
        mProgressDialog = new ProgressDialog(this);
        mProgressDialog.setMessage(getString(R.string.loading_text));
        mProgressDialog.setIndeterminate(true);
    }
    mProgressDialog.show();
}

public void hideProgressDialog() {
    if (mProgressDialog != null && mProgressDialog.isShowing()) {
        mProgressDialog.dismiss();
    }
}

public void onStop() {
    super.onStop();
    hideProgressDialog();
}
}

```

					ПГУ 1.090301.01.001 ПЗ	Лист
						45
Изм.	Лист	№ докум.	Подпись	Дата		

Приложение А.2 Файл RegisterActivity.java

```

package com.example.android.androidtraining.authentication;

import android.app.ProgressDialog;
import android.content.Intent;
import androidx.annotation.NonNull;
import androidx.annotation.VisibleForTesting;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.text.TextUtils;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;

import com.example.android.androidtraining.R;
import com.example.android.androidtraining.main.MainActivity;
import com.example.android.androidtraining.utilities.Constants;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.android.material.snackbar.Snackbar;
import com.google.firebase.auth.AuthResult;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;

public class RegisterActivity extends AppCompatActivity implements View.OnClickListener {

    private static String TAG = RegisterActivity.class.getSimpleName();

    private EditText mInputFirstName;
    private EditText mEmailEditField;
    private EditText mPasswordEditField;

    private Button mRegisterButton;

    @VisibleForTesting
    public ProgressDialog mProgressDialog;

    private TextView mLoginLink;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_register);

        mInputFirstName = (EditText) findViewById(R.id.inputFirstName);
        mEmailEditField = (EditText) findViewById(R.id.emailEditField);
        mPasswordEditField = (EditText) findViewById(R.id.passwordEditField);

        mRegisterButton = (Button) findViewById(R.id.registerButton);
        mRegisterButton.setOnClickListener(this);

        mLoginLink = (TextView) findViewById(R.id.loginLink);
        mLoginLink.setOnClickListener(this);
    }

    @Override
    public void onClick(View v) {
        switch (v.getId()) {
            case R.id.registerButton:
                register(mEmailEditField.getText().toString(),
                mPasswordEditField.getText().toString());
                break;
        }
    }

```

					ПГУ 1.090301.01.001 ПЗ	Лист
						46
Изм.	Лист	№ докум.	Подпись	Дата		

```

        case R.id.loginLink:
            finish();
            break;
        default:
            break;
    }
}

private void register(String email, String password) {
    if (!validateForm()) {
        return;
    }
    showProgressDialog();
    FirebaseAuth mAuth = FirebaseAuth.getInstance();
    mAuth.createUserWithEmailAndPassword(email, password)
        .addOnCompleteListener(this, new OnCompleteListener<AuthResult>() {
            @Override
            public void onComplete(@NonNull Task<AuthResult> task) {
                if (task.isSuccessful()) {
                    FirebaseUser mUser =
FirebaseAuth.getInstance().getCurrentUser();
                    // SharedPreferencesUtils.saveSharedPreferences(this, ); //
Тогда нужно где-то брать context
                    // перенести добавление записи о том, что пользователь
залогинился в начало MainActivity
                    updateUI(mUser);
                } else {
                    updateUI(null);
                }
                hideProgressDialog();
            }
        });
}

private boolean validateForm() {
    boolean valid = true;

    String email = mEmailEditField.getText().toString();
    if (TextUtils.isEmpty(email)) {
        mEmailEditField.setError("Enter e-mail");
        valid = false;
    } else {
        mEmailEditField.setError(null);
    }

    String password = mPasswordEditField.getText().toString();
    if (TextUtils.isEmpty(password)) {
        mPasswordEditField.setError("Enter password");
        valid = false;
    } else {
        mPasswordEditField.setError(null);
    }

    String firstName = mInputFirstName.getText().toString();
    if (TextUtils.isEmpty(firstName)) {
        mInputFirstName.setError(getString(R.string.enter_first_name_hint));
        valid = false;
    } else {
        mInputFirstName.setError(null);
    }

    return valid;
}

private void updateUI(FirebaseUser user) {
    hideProgressDialog();

```

					ПГУ 1.090301.01.001 ПЗ	Лист
						47
Изм.	Лист	№ докум.	Подпись	Дата		

```

        if (user != null) {
            Intent intent = new Intent(RegisterActivity.this, MainActivity.class);
            intent.putExtra(Constants.INTENT_PROFILE_FIRST_NAME_EXTRAS,
mInputFirstName.getText().toString());
            startActivity(intent);
            finish();
        } else {
            Snackbar.make(mRegisterButton,
getString(R.string.account_not_created_warning_text), Snackbar.LENGTH_SHORT).show();
        }
    }

    public void showProgressDialog() {
        if (mProgressDialog == null) {
            mProgressDialog = new ProgressDialog(this);
            mProgressDialog.setMessage(getString(R.string.loading_text));
            mProgressDialog.setIndeterminate(true);
        }
        mProgressDialog.show();
    }

    public void hideProgressDialog() {
        if (mProgressDialog != null && mProgressDialog.isShowing()) {
            mProgressDialog.dismiss();
        }
    }
}

```

Приложение А.3 Файл MainActivity.java

```

package com.example.android.androidtraining.main;

import android.content.Intent;
import android.os.Bundle;
import androidx.annotation.NonNull;
import androidx.annotation.Nullable;

import android.view.MenuItem;

import androidx.appcompat.app.AppCompatActivity;
import androidx.fragment.app.Fragment;
import androidx.fragment.app.FragmentManager;
import androidx.lifecycle.Observer;
import androidx.lifecycle.ViewModelProviders;

import com.example.android.androidtraining.authentication.LoginActivity;
import com.example.android.androidtraining.db.entities.TrainingDays;
import com.example.android.androidtraining.db.entities.TrainingWeeks;
import com.example.android.androidtraining.guide.GuidePagerActivity;
import com.example.android.androidtraining.R;
import com.example.android.androidtraining.utilities.Constants;
import com.example.android.androidtraining.viewmodels.TrainingPlansViewModel;
import com.example.android.androidtraining.utilities.SharedPreferencesUtils;
import com.example.android.androidtraining.db.entities.TrainingPlans;
import com.google.android.material.bottomnavigation.BottomNavigationView;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;

import java.util.ArrayList;
import java.util.List;

public class MainActivity extends AppCompatActivity {

    private static final String TAG = MainActivity.class.getSimpleName();

```

					ПГУ 1.090301.01.001 ПЗ	Лист
						48
Изм.	Лист	№ докум.	Подпись	Дата		

```

private BottomNavigationView mBottomNavigationView;

final WithoutPlanFragment mWithoutPlanFragment = new WithoutPlanFragment();
final PlanFragment mPlanFragment = new PlanFragment();
final TrainingFragment mTrainingFragment = new TrainingFragment();
final ProfileFragment mProfileFragment = new ProfileFragment();
final FragmentManager mFragmentManager = getSupportFragmentManager();
Fragment mActive;

// как раз модификатор доступа - виден только из одного и того же пакета
static TrainingPlans mCurrentStartedPlan;

private static FirebaseAuth mAuth;
private static FirebaseUser currentUser;

private TrainingPlansViewModel mTrainingPlansViewModel;

private List<TrainingPlans> mAllPlans;
private TrainingPlans mCurrentPlan;

TrainingPlans planToAdd;
ArrayList<TrainingWeeks> trainingWeeksArrayList;
ArrayList<TrainingDays> trainingDaysArrayList;

private BottomNavigationView.OnNavigationItemSelectedListener
mOnNavigationItemSelectedListener
    = new BottomNavigationView.OnNavigationItemSelectedListener() {

    @Override
    public boolean onNavigationItemSelected(@NonNull MenuItem item) {
        switch (item.getItemId()) {
            case R.id.navigation_plan:
                boolean isUserHasPlan =
                    SharedPreferencesUtils.readSharedSettings(MainActivity.this,
                        Constants.PREFERENCES_FILE, Constants.PREFERENCE_USER_HAS_PLAN,
                        false);
                if (isUserHasPlan) {
                    //Bundle currentPlanBundle = new Bundle();
                    //currentPlanBundle.putParcelable("CurrentPlan", mCurrentPlan);
                    //mPlanFragment.setArguments(currentPlanBundle);

                    mFragmentManager.beginTransaction().hide(mActive).show(mPlanFragment).commit();
                    mActive = mPlanFragment;

                } else {
                    mFragmentManager.beginTransaction().hide(mActive).show(mWithoutPlanFragment).commit();
                    mActive = mWithoutPlanFragment;
                }
                return true;
            case R.id.navigation_training:
                mFragmentManager.beginTransaction().hide(mActive).show(mTrainingFragment).commit();
                mActive = mTrainingFragment;
                return true;
            case R.id.navigation_profile:
                mFragmentManager.beginTransaction().hide(mActive).show(mProfileFragment).commit();
                mActive = mProfileFragment;
                return true;
        }
        return false;
    }
}

```

					ПГУ 1.090301.01.001 ПЗ	Лист
						49
Изм.	Лист	№ докум.	Подпись	Дата		

```

};

private void checkDB() {
    boolean mUserLoggedIn = SharedPreferencesUtils.readSharedSettings(this,
Constants.PREFERENCES_FILE, Constants.PREFERENCE_USER_LOGGED_IN, false);
    boolean mUserHasPlan = SharedPreferencesUtils.readSharedSettings(this,
Constants.PREFERENCES_FILE, Constants.PREFERENCE_USER_HAS_PLAN, false);
    if (mUserLoggedIn) {
        if (!mUserHasPlan) {
            // Плана нет, пытаемся прочитать данные с сервера и добавить их локально
            // и затем обязательно проверяем переменную еще раз
            // а нужно ли и для чего?
            // скорее всего в этом нет необходимости, потому что это проверяться будет
дальше, а переменная изменится, если добавятся данные
            // если нет, то все ок

            //boolean mUserHasPlanAfterTryOnlineAdd =
SharedPreferencesUtils.readSharedSettings(this, Constants.PREFERENCES_FILE,
Constants.PREFERENCE_USER_HAS_PLAN, false);
            //if ()
            mTrainingPlansViewModel.loadPlanWeeksDaysFromServer(this);
            mTrainingPlansViewModel.loadUserProfile();
            // если добавятся, то как раз те observable, которые ниже, получат
необходимые данные
            // если нет, то все также корректно обрабатывается
        }
        // если пользователь не залогинился, то не пытаемся загружать данные с сервера
    }
}

@Override
protected void onCreate(Bundle savedInstanceState) {

    setTheme(R.style.AppTheme); // Убираем SplashScreen
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    mAuth = FirebaseAuth.getInstance();
    boolean mIsUserFirstTime =
SharedPreferencesUtils.readSharedSettings(MainActivity.this,
Constants.PREFERENCES_FILE, Constants.PREFERENCE_USER_FIRST_TIME, true);
    if (mIsUserFirstTime) {
        Intent guideIntent = new Intent(this,
GuidePagerActivity.class);
        guideIntent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
        startActivity(guideIntent);
        finish();
        return;
    } else {
        mCurrentUser = mAuth.getCurrentUser();
        if (mCurrentUser == null) {
            Intent authenticationIntent = new Intent(this,
LoginActivity.class);
            authenticationIntent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
            startActivity(authenticationIntent);
            finish();
            return;
        } else {
            SharedPreferencesUtils.saveSharedPreferences(this,
Constants.PREFERENCES_FILE, Constants.PREFERENCE_USER_LOGGED_IN, true);
        }
    }

    //mTextMessage = (TextView) findViewById(R.id.message);
    mBottomNavigationView = (BottomNavigationView)
findViewById(R.id.bottomNavigationView);

```

					ПГУ 1.090301.01.001 ПЗ	Лист
						50
Изм.	Лист	№ докум.	Подпись	Дата		

```

mBottomNavigationView.setOnNavigationItemSelectedListener(mOnNavigationItemSelectedListener
);
    // На момент создания ViewModel, а соответственно и Repository, аутентификация уже
будет произведена
    mTrainingPlansViewModel =
ViewModelProviders.of(this).get(TrainingPlansViewModel.class);

    // Активность была запущена из RegisterActivity, и только что созданного
пользователя надо добавить на сервер и локально
    Intent intent = getIntent();
    String firstName =
intent.getStringExtra(Constants.INTENT_PROFILE_FIRST_NAME_EXTRAS);
    // Если Bundle, который может придти только из CreatePlanActivity нулевой, то норм,
это
    // значит, что MainActivity вызван не послед создания плана, недель и дней
    if (firstName != null) {
        if (!firstName.isEmpty()) {
            mTrainingPlansViewModel.insertUserProfile(firstName);
        }
    }

    // Получив mTrainingPlansViewModel
    // Далее, нужно обязательно проверить, есть ли данные локально в БД, если нет, то
нужно
    // попытаться прочесть их с сервера, если же их и там нет, то все норм - это
первый запуск
    // или приложением не пользовались

    checkDB();

    // Определяем есть ли БД локально, если нет, то пытаемся загрузить с сервера
    // если и там нет, то все верно, устанавливаем настройку "плана есть" в false
    // и продолжаем работу приложения

    // Проверка intent не нужна, так как даже при первом запуске, ОС сама создает его
    // и он точно не будет равен null, а вот Bundle стоит проверять по этому ключу
    // ключ вынести в Constants
    Bundle planWeeksDaysBundle =
intent.getBundleExtra(Constants.BUNDLE_PLAN_WEEKS_DAYS);
    if (planWeeksDaysBundle != null) {

        trainingDaysArrayList = new ArrayList<>();
        trainingWeeksArrayList = new ArrayList<>();
        //planWeeksAndDays = new HashMap<>();

        planToAdd = planWeeksDaysBundle.getParcelable("TrainingPlan");
        trainingWeeksArrayList =
planWeeksDaysBundle.getParcelableArrayList("TrainingWeeksList");
        trainingDaysArrayList =
planWeeksDaysBundle.getParcelableArrayList("TrainingDaysList");

        // Добавление плана в БД локальную и серверную
        mTrainingPlansViewModel.insertPlanWeeksDays(planToAdd, trainingWeeksArrayList,
trainingDaysArrayList);

        SharedPreferencesUtils.saveSharedPreferences(this, Constants.PREFERENCES_FILE,
Constants.PREFERENCE_USER_HAS_PLAN, true);
    }

    mTrainingPlansViewModel.getAllPlans().observe(this, new
Observer<List<TrainingPlans>>() {
        @Override
        public void onChanged(@Nullable List<TrainingPlans> trainingPlans) {

```

					ПГУ 1.090301.01.001 ПЗ	Лист
						51
Изм.	Лист	№ докум.	Подпись	Дата		

```

        if (!trainingPlans.isEmpty()) {
            mAllPlans = trainingPlans;
            mCurrentPlan = trainingPlans.get(0);
            //Log.d(TAG, "Данные изменились!");
            SharedPreferencesUtils.saveSharedPreferences(MainActivity.this,
Constants.PREFERENCES_FILE, Constants.PREFERENCE_USER_HAS_PLAN, true);
        } else {
            // Планов нет, значит и недель и дней
            SharedPreferencesUtils.saveSharedPreferences(MainActivity.this,
Constants.PREFERENCES_FILE, Constants.PREFERENCE_USER_HAS_PLAN, false);
            checkDB();
        }
    }
});
mTrainingPlansViewModel.getPlansByStatus().observe(this, new
Observer<List<TrainingPlans>>() {
    @Override
    public void onChanged(@Nullable List<TrainingPlans> trainingPlans) {
        //mCurrentPlan = trainingPlans;
    }
});
mTrainingPlansViewModel.getCurrentStartedPlan().observe(this, new
Observer<TrainingPlans>() {
    @Override
    public void onChanged(@Nullable TrainingPlans trainingPlans) {
        mCurrentStartedPlan = trainingPlans;
    }
});

mFragmentManager.beginTransaction().add(R.id.bottomNavigationFragmentContainer,
mWithoutPlanFragment, "4").hide(mWithoutPlanFragment).commit();
mFragmentManager.beginTransaction().add(R.id.bottomNavigationFragmentContainer,
mPlanFragment, "3").hide(mPlanFragment).commit();
mFragmentManager.beginTransaction().add(R.id.bottomNavigationFragmentContainer,
mProfileFragment, "2").hide(mProfileFragment).commit();
mFragmentManager.beginTransaction().add(R.id.bottomNavigationFragmentContainer,
mTrainingFragment, "1").hide(mTrainingFragment).commit();
mBottomNavigationView.getMenu().getItem(1).setChecked(true);
mActive = mTrainingFragment;
mBottomNavigationView.setSelectedItemId(R.id.navigation_training);
}
}
}

```

					ПГУ 1.090301.01.001 ПЗ	Лист
						52
Изм.	Лист	№ докум.	Подпись	Дата		

Приложение А.4 Файл GuidePagerActivity.java

```
package com.example.android.androidtraining.guide;

import android.animation.ArgbEvaluator;
import android.content.Intent;
import androidx.fragment.app.Fragment;
import androidx.fragment.app.FragmentManager;
import androidx.fragment.app.FragmentPagerAdapter;
import androidx.core.content.ContextCompat;
import androidx.viewpager.widget.ViewPager;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.ImageButton;
import android.widget.ImageView;

import com.example.android.androidtraining.R;
import com.example.android.androidtraining.utilities.Constants;
import com.example.android.androidtraining.utilities.SharedPreferencesUtils;
import com.example.android.androidtraining.authentication.LoginActivity;

public class GuidePagerActivity extends AppCompatActivity implements View.OnClickListener {

    private static final String TAG = GuidePagerActivity.class.getSimpleName();
    private static final int NUMBER_OF_PAGES = 3;

    private ViewPager mViewPager;
    ImageButton mNextBtn;
    Button mSkipBtn, mFinishBtn;
    ImageView[] mIndicators;
    int[] mColorList;
    int mPage = 0;
    final ArgbEvaluator mEvaluator = new ArgbEvaluator();
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_guide_pager);
        mNextBtn = (ImageButton) findViewById(R.id.introBtnNext);
        mNextBtn.setOnClickListener(this);
        mSkipBtn = (Button) findViewById(R.id.introBtnSkip);
        mSkipBtn.setOnClickListener(this);
        mFinishBtn = (Button) findViewById(R.id.introBtnFinish);
        mFinishBtn.setOnClickListener(this);

        mIndicators = new ImageView[]{(ImageView)findViewById(R.id.introIndicator0),
        (ImageView)findViewById(R.id.introIndicator1),
        (ImageView)findViewById(R.id.introIndicator2)};

        mColorList = new int[]{ContextCompat.getColor(this, R.color.cyan),
        ContextCompat.getColor(this, R.color.orange),
        ContextCompat.getColor(this, R.color.green)};

        mViewPager = (ViewPager) findViewById(R.id.containerViewPager);
        mViewPager.setAdapter(new SectionsPagerAdapter(getSupportFragmentManager()));
        mViewPager.addOnPageChangeListener(new ViewPager.OnPageChangeListener() {
            @Override
            public void onPageScrolled(int i, float v, int i1) {
                int colorUpdate = (Integer) mEvaluator.evaluate(v, mColorList[i],
                mColorList[i == 2 ? i : i + 1]);
                mViewPager.setBackgroundColor(colorUpdate);
            }
        });
    }
    @Override
```

					ПГУ 1.090301.01.001 ПЗ	Лист
						53
Изм.	Лист	№ докум.	Подпись	Дата		

```

        public void onPageSelected(int i) {
            mPage = i;
            updateIndicators(mPage);

            mNextBtn.setVisibility(mPage == 2 ? View.GONE : View.VISIBLE);
            mFinishBtn.setVisibility(mPage == 2 ? View.VISIBLE : View.GONE);
        }

        @Override
        public void onPageScrollStateChanged(int i) {
        }

    });
}

void updateIndicators(int position) {
    for (int i = 0; i < mIndicators.length; i++) {
        mIndicators[i].setBackgroundResource(
            i == position ? R.drawable.indicator_selected :
R.drawable.indicator_unselected
        );
    }
}

@Override
public void onClick(View v) {
    Intent authenticationIntent;
    switch (v.getId()) {
        case R.id.introBtnSkip:
            SharedPreferencesUtils.saveSharedPreferences(GuidePagerActivity.this,
Constants.PREFERENCES_FILE, Constants.PREFERENCE_USER_FIRST_TIME, false);
            authenticationIntent = new Intent(GuidePagerActivity.this,
                LoginActivity.class);
            startActivity(authenticationIntent);
            finish();
            break;
        case R.id.introBtnNext:
            mPage += 1;
            mViewPager.setCurrentItem(mPage, true);
            break;
        case R.id.introBtnFinish:
            SharedPreferencesUtils.saveSharedPreferences(GuidePagerActivity.this,
Constants.PREFERENCES_FILE, Constants.PREFERENCE_USER_FIRST_TIME, false);
            authenticationIntent = new Intent(GuidePagerActivity.this,
                LoginActivity.class);
            startActivity(authenticationIntent);
            finish();
            break;
    }
}

public class SectionsPagerAdapter extends FragmentPagerAdapter {

    public SectionsPagerAdapter(FragmentManager fragmentManager) {
        super(fragmentManager);
    }

    @Override
    public Fragment getItem(int position) {
        return GuidePagerFragment.newInstance(position);
    }

    @Override
    public int getCount() {
        return NUMBER_OF_PAGES;
    }
}
}

```

					ПГУ 1.090301.01.001 ПЗ	Лист
						54
Изм.	Лист	№ докум.	Подпись	Дата		

Приложение А.5 Файл ProfileFragment.java

```
package com.example.android.androidtraining.main;

import android.os.Bundle;
import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.fragment.app.Fragment;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.DatePicker;
import android.widget.EditText;
import android.widget.TextView;

import androidx.lifecycle.Observer;
import androidx.lifecycle.ViewModel;
import androidx.lifecycle.ViewModelProviders;

import com.example.android.androidtraining.R;
import com.example.android.androidtraining.db.entities.UserProfile;
import com.example.android.androidtraining.utilities.Converters;
import com.example.android.androidtraining.viewmodels.TrainingPlansViewModel;

import java.util.Calendar;
import java.util.Date;

public class ProfileFragment extends Fragment {

    TextView mFirstLastNameProfileTextView;
    TextView mEmailProfileTextView;

    // Переделать в EditText, чтобы поля были редактируемыми
    EditText mWeightProfileEditText;
    EditText mHeightProfileEditText;

    EditText mLastNameProfileEditText;
    DatePicker mDateBirthDayPicker;

    Button mSaveChangesButton;

    UserProfile mCurrentUserProfile;

    ViewModel mTrainingPlansViewModel;

    public static Date getDateFromDatePicker(DatePicker datePicker){
        int day = datePicker.getDayOfMonth();
        int month = datePicker.getMonth();
        int year = datePicker.getYear();

        Calendar calendar = Calendar.getInstance();
        calendar.set(year, month, day);

        return calendar.getTime();
    }

    @Nullable
    @Override
    public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup
container, @Nullable Bundle savedInstanceState) {

        View view = inflater.inflate(R.layout.fragment_profile, container, false);

        mFirstLastNameProfileTextView =
```

					ПГУ 1.090301.01.001 ПЗ	Лист
						55
Изм.	Лист	№ докум.	Подпись	Дата		

```

view.findViewById(R.id.firstNameProfileTextView);
mWeightProfileEditText = view.findViewById(R.id.weightProfileTextView);
mHeightProfileEditText = view.findViewById(R.id.heightProfileTextView);
mEmailProfileTextView = view.findViewById(R.id.emailProfileTextView);

mLastNameProfileEditText = view.findViewById(R.id.lastNameProfileEditText);
mDateBirthDayPicker = view.findViewById(R.id.dateBirthDayPicker);

mSaveChangesButton = view.findViewById(R.id.saveChangesButton);
mSaveChangesButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        String lastName = mLastNameProfileEditText.getText().toString();
        float weight =
Float.parseFloat(mWeightProfileEditText.getText().toString());
        float height =
Float.parseFloat(mHeightProfileEditText.getText().toString());

        if (mDateBirthDayPicker.isEnabled()) {
            Date birthdayDate = getDateFromDatePicker(mDateBirthDayPicker);
            String birthdayDateString =
Long.toString(Converters.dateToTimestamp(birthdayDate));
            mCurrentUserProfile.setBirthDate(birthdayDateString);
        }

        if (!lastName.isEmpty()) {
            mCurrentUserProfile.setLastName(lastName);
        }
        if (weight != 0) {
            mCurrentUserProfile.setWeight(weight);
        }
        if (height != 0) {
            mCurrentUserProfile.setHeight(height);
        }

        ((TrainingPlansViewModel)
mTrainingPlansViewModel).updateProfileRecord(mCurrentUserProfile);
    }
});

mTrainingPlansViewModel =
ViewModelProviders.of(this).get(TrainingPlansViewModel.class);
((TrainingPlansViewModel)
mTrainingPlansViewModel).getCurrentUserProfile().observe(this, new Observer<UserProfile>()
{
    @Override
    public void onChanged(@Nullable UserProfile userProfile) {
        mCurrentUserProfile = userProfile;
        if (mCurrentUserProfile != null) {
            String firstLastName = "";
            if (mCurrentUserProfile.getLastName() == null) {
                firstLastName = mCurrentUserProfile.getFirstName();
            } else {
                if (!mCurrentUserProfile.getLastName().isEmpty()) {
                    firstLastName = mCurrentUserProfile.getFirstName() + " " +
mCurrentUserProfile.getLastName();
                }
            }
            mEmailProfileTextView.setText(mCurrentUserProfile.getEmail());
            mLastNameProfileEditText.setText(userProfile.getLastName());
            mFirstNameProfileTextView.setText(firstLastName);
            if (mCurrentUserProfile.getWeight() != 0) {

mWeightProfileEditText.setText(Float.toString(mCurrentUserProfile.getWeight()));
        }
    }
}

```

					ПГУ 1.090301.01.001 ПЗ	Лист
						56
Изм.	Лист	№ докум.	Подпись	Дата		

```

        if (mCurrentUserProfile.getHeight() != 0) {
mHeightProfileEditText.setText(Float.toString(mCurrentUserProfile.getHeight()));
        }
        if (mCurrentUserProfile.getBirthDate() != null) {
            //SimpleDateFormat sdf = new SimpleDateFormat("dd-MM-yyyy");
            Date birthdayDate =
Converters.fromTimestamp(Long.parseLong(mCurrentUserProfile.getBirthDate()));
            Calendar birthdayDateCalendar = Calendar.getInstance();
            birthdayDateCalendar.setTime(birthdayDate);

mDateBirthDayPicker.updateDate(birthdayDateCalendar.get(Calendar.YEAR),
birthdayDateCalendar.get(Calendar.MONTH), birthdayDateCalendar.get(Calendar.DAY_OF_MONTH));
        }
    }
}
});
return view;
}
}

```

Приложение А.6 Файл PlanFragment.java

```

package com.example.android.androidtraining.main;

import android.content.Intent;
import android.os.Bundle;
import androidx.annotation.NonNull;
import androidx.annotation.Nullable;

import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;

import androidx.fragment.app.Fragment;
import androidx.lifecycle.Observer;
import androidx.lifecycle.ViewModel;
import androidx.lifecycle.ViewModelProviders;

import com.example.android.androidtraining.R;
import com.example.android.androidtraining.db.entities.TrainingPlans;
import com.example.android.androidtraining.main.plansrecyclerview.PlansRecyclerView;
import com.example.android.androidtraining.viewmodels.TrainingPlansViewModel;

public class PlanFragment extends Fragment {

    TrainingPlans mCurrentStartedPlan;

    ViewModel trainingViewModel;

    Button mShowExtendedPlanInfo;

    @Nullable
    @Override
    public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup
container, @Nullable Bundle savedInstanceState) {

        View view = inflater.inflate(R.layout.fragment_plan, container, false);

        trainingViewModel = ViewModelProviders.of(this).get(TrainingPlansViewModel.class);
        ((TrainingPlansViewModel) trainingViewModel).getCurrentStartedPlan().observe(this,

```

					ПГУ 1.090301.01.001 ПЗ	Лист
						57
Изм.	Лист	№ докум.	Подпись	Дата		

```

new Observer<TrainingPlans>() {
    @Override
    public void onChanged(@Nullable TrainingPlans trainingPlans) {
        mCurrentStartedPlan = trainingPlans;
    }
});

mShowExtendedPlanInfo = (Button) view.findViewById(R.id.showExtendedPlanInfo);
mShowExtendedPlanInfo.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Intent intent = new Intent(getActivity(), PlansRecyclerActivity.class);
        startActivity(intent);
    }
});

return view;
}
}

```

					ПГУ 1.090301.01.001 ПЗ	Лист
						58
Изм.	Лист	№ докум.	Подпись	Дата		

Приложение А.7 Файл WithoutPlanFragment.java

```
package com.example.android.androidtraining.main;

import android.content.Intent;
import android.os.Bundle;
import androidx.fragment.app.Fragment;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;

import com.example.android.androidtraining.R;
import com.example.android.androidtraining.createplan.CreatePlanActivity;

public class WithoutPlanFragment extends Fragment implements View.OnClickListener {

    private Button mCreatePlanButton;

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle
savedInstanceState) {
        View view = inflater.inflate(R.layout.fragment_without_plan, container, false);

        mCreatePlanButton = (Button) view.findViewById(R.id.createPlanButton);
        mCreatePlanButton.setOnClickListener(this);

        return view;
    }

    @Override
    public void onClick(View v) {
        switch(v.getId()) {
            case R.id.createPlanButton:

//SharedPreferencesUtils.saveSharedPreferences(getActivity().getApplicationContext(),
//Constants.PREFERENCES_FILE, Constants.PREFERENCE_USER_HAS_PLAN,
true);
                Intent intent = new Intent(getActivity().getApplicationContext(),
CreatePlanActivity.class);
                startActivity(intent);
                //startActivityForResult(); вместе с Bundle???
                break;
            default:
                break;
        }
    }
}
```

					ПГУ 1.090301.01.001 ПЗ	Лист
						59
Изм.	Лист	№ докум.	Подпись	Дата		

Приложение А.7 Файл CreatePlanActivity.java

```
package com.example.android.androidtraining.createplan;

import android.content.Intent;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import android.view.ContextMenu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ExpandableListView;
import android.widget.TextView;

import com.example.android.androidtraining.R;
import com.example.android.androidtraining.db.entities.TrainingDays;
import com.example.android.androidtraining.db.entities.TrainingPlans;
import com.example.android.androidtraining.db.entities.TrainingWeeks;
import com.example.android.androidtraining.main.MainActivity;
import com.example.android.androidtraining.utilities.Constants;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;

public class CreatePlanActivity extends AppCompatActivity implements View.OnClickListener {

    //private RecyclerView mPlansRecyclerView;

    private static String TAG = CreatePlanActivity.class.getSimpleName();

    private EditText mWeeksPlanCount;
    private EditText mDaysPlanCount;

    private Button mDoneExpandableListViewButton;
    private Button mCreateExpandableListViewPlanButton;

    private int weeksCount;
    private int daysCount;

    private ExpandableListView listView;
    private ExpandableListAdapter listAdapter;

    //private List<String> weeksDataHeader;
    //private List<String> daysDataHeader;
    //private HashMap<String, List<String>> weeksAndDaysDataHeaders;

    //private List<TrainingWeeks> weeksList;
    //private List<TrainingDays> daysList;

    private ArrayList<TrainingDays> daysList;
    private ArrayList<TrainingWeeks> weeksList;
    private HashMap<TrainingWeeks, List<TrainingDays>> planWeeksAndDays;

    private TextView allDaysOfPlan;
    private TextView distributedDays;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_create_plan);

        mWeeksPlanCount = (EditText) findViewById(R.id.weeksPlanCount);
        mDaysPlanCount = (EditText) findViewById(R.id.daysPlanCount);
    }
}
```

					ПГУ 1.090301.01.001 ПЗ	Лист
						60
Изм.	Лист	№ докум.	Подпись	Дата		

```

        mDoneExpandableListViewButton = (Button)
findViewById(R.id.doneExpandableListViewPlanButton);
        mDoneExpandableListViewButton.setOnClickListener(this);

        mCreateExpandableListViewPlanButton = (Button)
findViewById(R.id.createExpandableListViewPlanButton);
        mCreateExpandableListViewPlanButton.setOnClickListener(this);

        allDaysOfPlan = (TextView) findViewById(R.id.allDaysOfPlan);
        distributedDays = (TextView) findViewById(R.id.distributedDays);

        listView = (ExpandableListView) findViewById(R.id.expandableWeeksAndDaysListView);
    }

    @Override
    public void onClick(View v) {
        switch (v.getId()) {
            case R.id.doneExpandableListViewPlanButton:
                //mWeeksPlanCount.setEnabled(false);
                //mDaysPlanCount.setEnabled(false);
                //mDoneExpandableListViewButton.setEnabled(false);

                // validate EditText fields methods need!
                weeksCount = Integer.parseInt(mWeeksPlanCount.getText().toString());
                daysCount = Integer.parseInt(mDaysPlanCount.getText().toString());

                createPlanWithWeeksAndDaysCount(weeksCount, daysCount);
                break;
            case R.id.createExpandableListViewPlanButton:
                // Добавить метод добавления всех записей (объекто недель и дней) из listHash в
БД
                // Все данные о неделях и днях находятся в HashMap<TrainingWeeks,
List<TrainingDays>> planWeeksAndDays;
                // нужно придумать, как оттуда передать их в главную активность, а оттуда уже
добавлять в БД
                Intent intent = new Intent(this, MainActivity.class);
                Bundle bundle = new Bundle();

                TrainingPlans plan = new TrainingPlans(weeksCount, daysCount);
                bundle.putParcelable("TrainingPlan", plan); // План, который нужно добавить в БД
                bundle.putParcelableArrayList("TrainingWeeksList", weeksList); // Список недель,
которые нужно добавить в БД

                daysList = new ArrayList<>();
                for (int i = 0; i < weeksList.size(); i++) {
                    for (int j = 0; j < 7; j++) {
                        daysList.add(planWeeksAndDays.get(weeksList.get(i)).get(j)); // Здесь
будут все дни недели
                    }
                }
                //bundle.putParcelableArrayList(Integer.toString(weeksList.get(i).getWeekNumber()),
daysArrayList);
                bundle.putParcelableArrayList("TrainingDaysList", daysList);

                intent.putExtra(Constants.BUNDLE_PLAN_WEEKS_DAYS, bundle);
                startActivity(intent);
                break;
            default:
                break;
        }
    }

    private void createPlanWithWeeksAndDaysCount(int weeksCount, int daysCount) {

        // Не забыть про то, что на основе weeksCount и daysCount нужно будет создавать счетчик
        // а заеетм блокировать список выбора

```

					ПГУ 1.090301.01.001 ПЗ	Лист
						61
Изм.	Лист	№ докум.	Подпись	Дата		

```

allDaysOfPlan.setText("Дней всего:" + daysCount);
distributedDays.setText("Распределено:" + daysCount);

//weeksDataHeader = new ArrayList<>();
weeksList = new ArrayList<>();
planWeeksAndDays = new HashMap<>();
//weeksAndDaysDataHeaders = new HashMap<>();

for (int i = 0; i < weeksCount; i++) {

    List<TrainingDays> daysList = new ArrayList<>();
    //List<String> daysHeader = new ArrayList<>();

    TrainingWeeks week = new TrainingWeeks(i + 1);
    weeksList.add(week);

    for (int j = 0; j < 7; j++) {

        daysList.add(new TrainingDays(j + 7 * i + 1));
        //daysHeader.add("Day " + (j + 7 * i + 1));

    }

    planWeeksAndDays.put(week, daysList);
    //weeksAndDaysDataHeaders.put(weeksDataHeader.get(i), daysHeader);
}

listAdapter = new ExpandableListAdapter(this, weeksList, planWeeksAndDays);
listView.setAdapter(listAdapter);
registerForContextMenu(listView);
}

@Override
public void onCreateContextMenu(ContextMenu menu, View v, ContextMenu.ContextMenuInfo
menuInfo) {
    super.onCreateContextMenu(menu, v, menuInfo);

    ExpandableListView.ExpandableListContextMenuInfo info =
(ExpandableListView.ExpandableListContextMenuInfo) menuInfo;

    int type = ExpandableListView.getPackedPositionType(info.packedPosition);
    //int group = ExpandableListView.getPackedPositionGroup(info.packedPosition);
    //int child = ExpandableListView.getPackedPositionChild(info.packedPosition);

    // Only create a context menu for child items
    if (type == 1) {
        MenuInflater inflater = getMenuInflater();
        inflater.inflate(R.menu.context_menu, menu);
        Log.d(TAG, "menu created");
    }
}

@Override
public boolean onContextItemSelected(MenuItem item) {
    ExpandableListView.ExpandableListContextMenuInfo info =
(ExpandableListView.ExpandableListContextMenuInfo) item getMenuInfo();
    int groupPos = 0, childPos = 0;
    int type = ExpandableListView.getPackedPositionType(info.packedPosition);
    if (type == ExpandableListView.PACKED_POSITION_TYPE_CHILD) {
        // Здесь мы получили позицию элемента (дня в нашем случае)
        // и теперь, выбрав определенный пункт, нужно занести выбранное значение в поле type
        объекта
        groupPos = ExpandableListView.getPackedPositionGroup(info.packedPosition);
        childPos = ExpandableListView.getPackedPositionChild(info.packedPosition);
    }

    Log.d(TAG, "item select...");
}

```

					ПГУ 1.090301.01.001 ПЗ	Лист
						62
Изм.	Лист	№ докум.	Подпись	Дата		

```

        // По задумке, придется поменять только соответствующие данные в дне
        // ну и сразу устанавливаем в textView, чтобы не пролистывать, чтобы тип обновился
        switch (item.getItemId()) {
            case R.id.run_training_option:
                TextView textView = info.targetView.findViewById(R.id.labelListItemType);
                textView.setText("Бег");

        planWeeksAndDays.get(weeksList.get(groupPos)).get(childPos).setTrainingDayType("Run");
                Log.d(TAG, "run set text");
                break;
            case R.id.gym_training_option:
                TextView textView2 = info.targetView.findViewById(R.id.labelListItemType);
                textView2.setText("Упражнения");

        planWeeksAndDays.get(weeksList.get(groupPos)).get(childPos).setTrainingDayType("Gym");
                Log.d(TAG, "gym set text");
                break;
            case R.id.rest_training_option:
                TextView textView3 = info.targetView.findViewById(R.id.labelListItemType);
                textView3.setText("Отдых");

        planWeeksAndDays.get(weeksList.get(groupPos)).get(childPos).setTrainingDayType("Rest");
                Log.d(TAG, "rest set text");
                break;
            default:
                break;
        }

        return super.onContextItemSelected(item);
    }
}

```

					ПГУ 1.090301.01.001 ПЗ	Лист
						63
Изм.	Лист	№ докум.	Подпись	Дата		

ПРИЛОЖЕНИЕ Б
РИСУНКИ И ДИАГРАММЫ

					<i>ПГУ 1.090301.01.001 ПЗ</i>	<i>Лист</i>
						64
<i>Изм.</i>	<i>Лист</i>	<i>№ докум.</i>	<i>Подпись</i>	<i>Дата</i>		

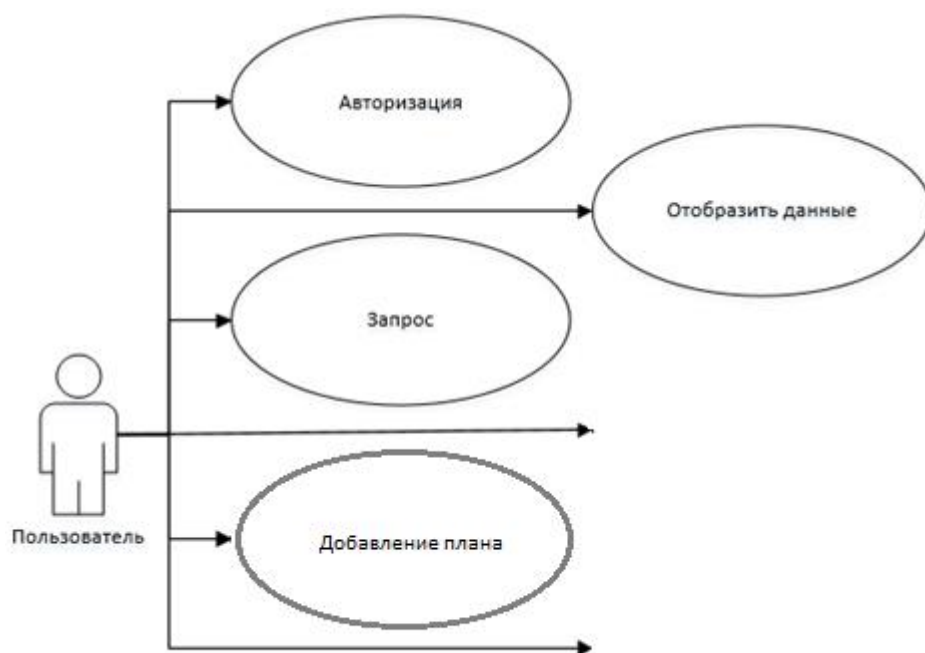


Рисунок 18 – Диаграмма вариантов использования

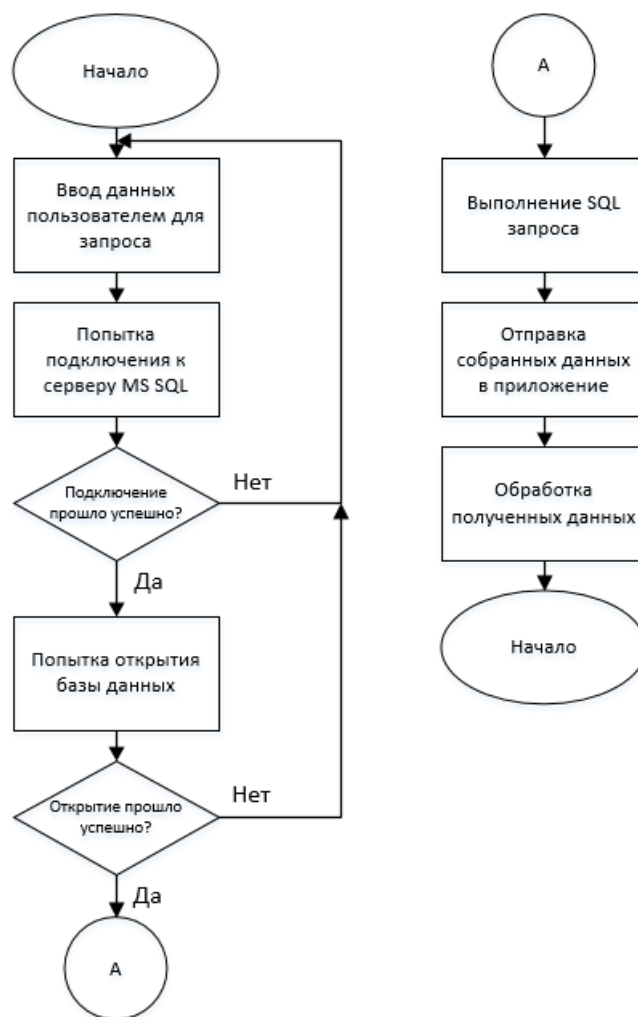


Рисунок 19 – Алгоритм обработки запроса

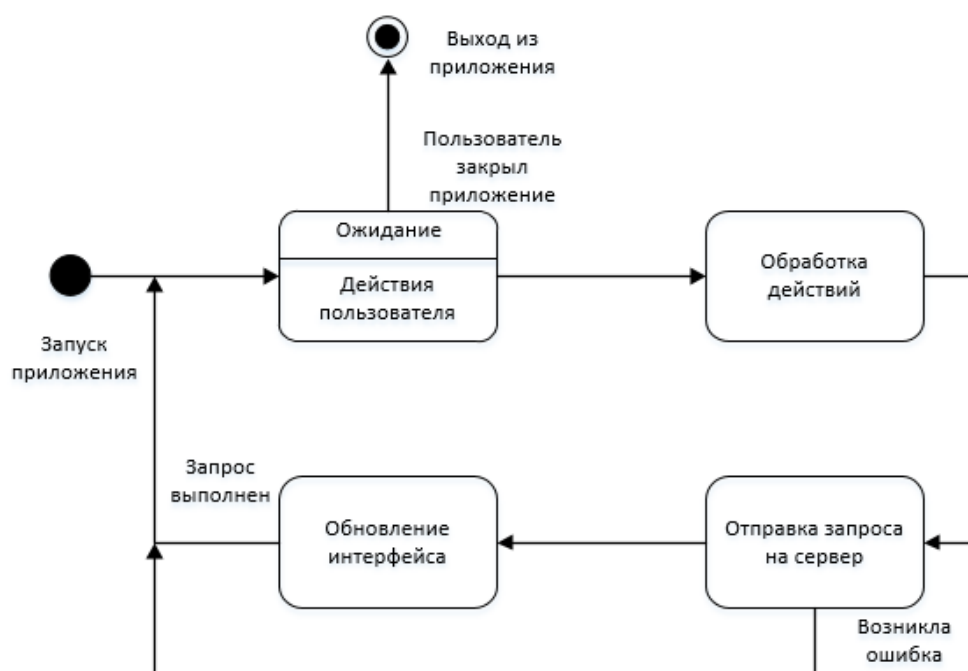


Рисунок 20 – Диаграмма состояний приложения

ПРИЛОЖЕНИЕ В
ПРЕЗЕНТАЦИЯ

					<i>ПГУ 1.090301.01.001 ПЗ</i>	<i>Лист</i>
						67
<i>Изм.</i>	<i>Лист</i>	<i>№ докум.</i>	<i>Подпись</i>	<i>Дата</i>		

Тема работы: информационная система контроля тренировки спортсменов

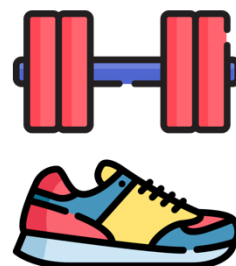
Научный руководитель: к.т.н. доцент Трокоз Д. А.

Выполнил: студент гр. 15ВВ1 Агеев Сергей

Активация Windows
Чтобы активировать Windows, перейдите в раздел "Параметры"

Рисунок 21 – Презентация – вводный слайд

Цель работы?



Цель работы – разработка информационной системы контроля тренировок спортсменов, позволяющая пользователю создавать персонализированные планы и отслеживать прогресс их выполнения.

Активация Windows
Чтобы активировать Windows, перейдите в раздел "Параметры"

Рисунок 22 – Презентация – цель работы

					ПГУ 1.090301.01.001 ПЗ	Лист
						68
Изм.	Лист	№ докум.	Подпись	Дата		

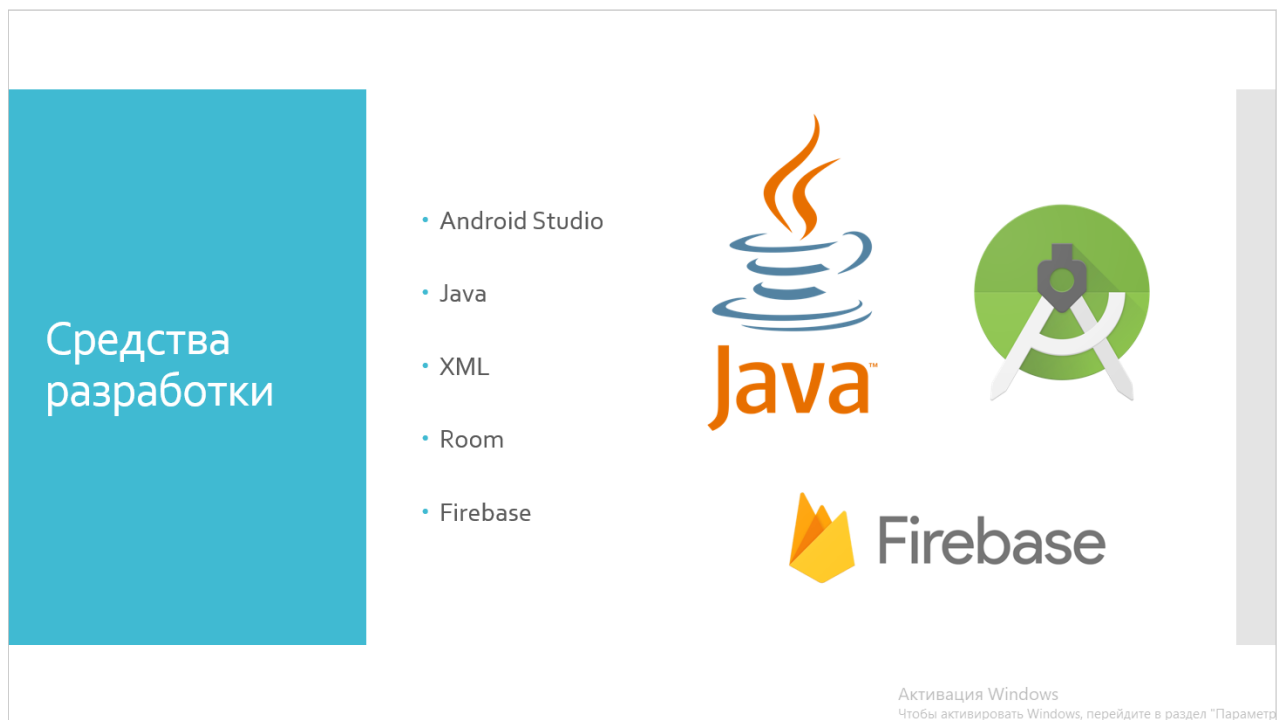


Рисунок 23 – Презентация – инструменты реализации проекта

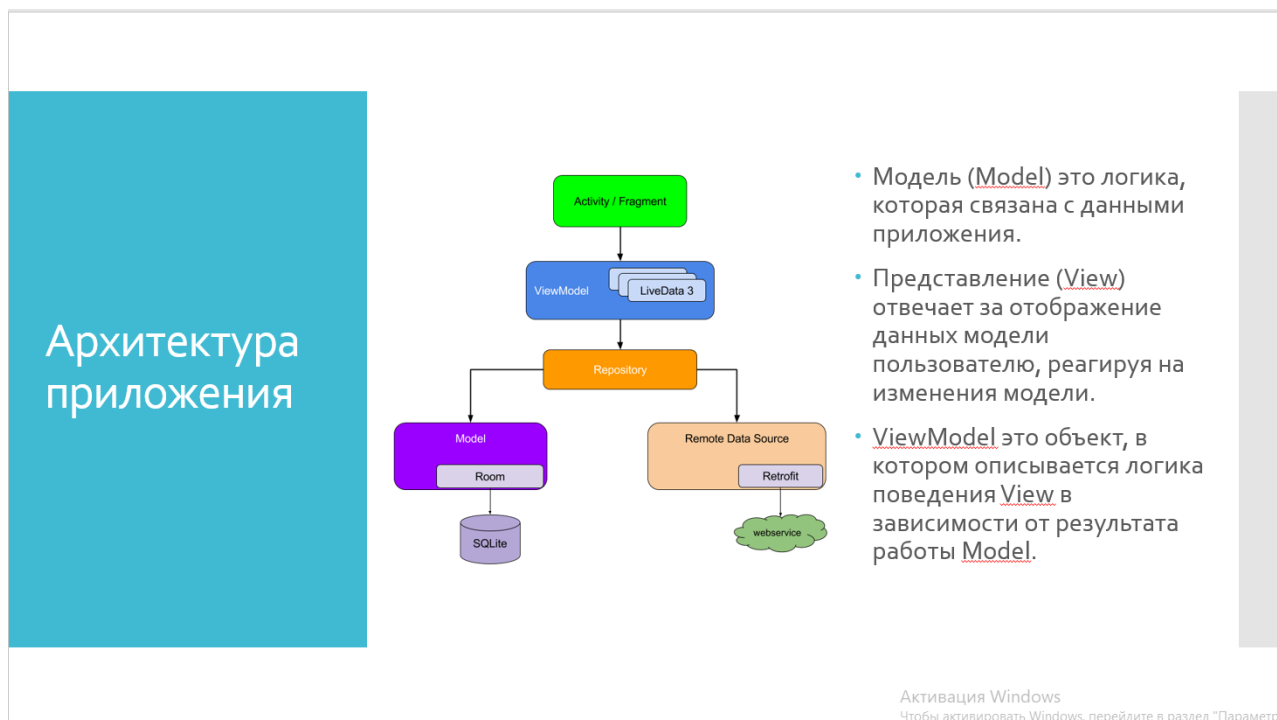


Рисунок 24 – Презентация – структура приложения

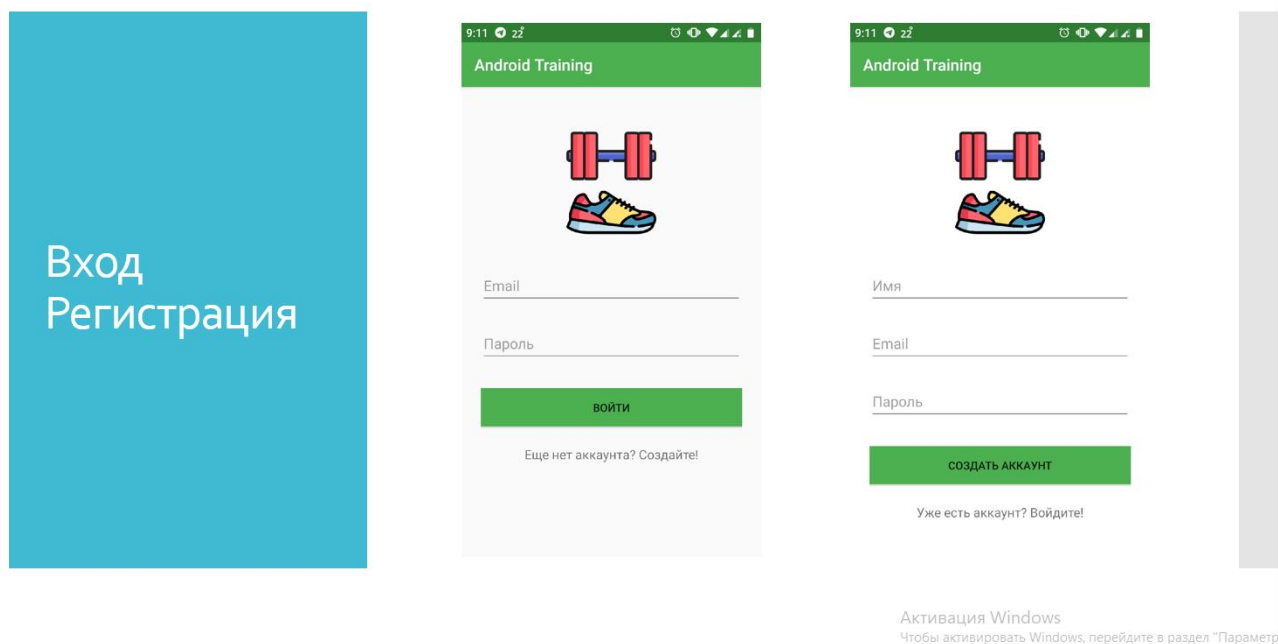


Рисунок 20 – Презентация – экран «Аутентификация»

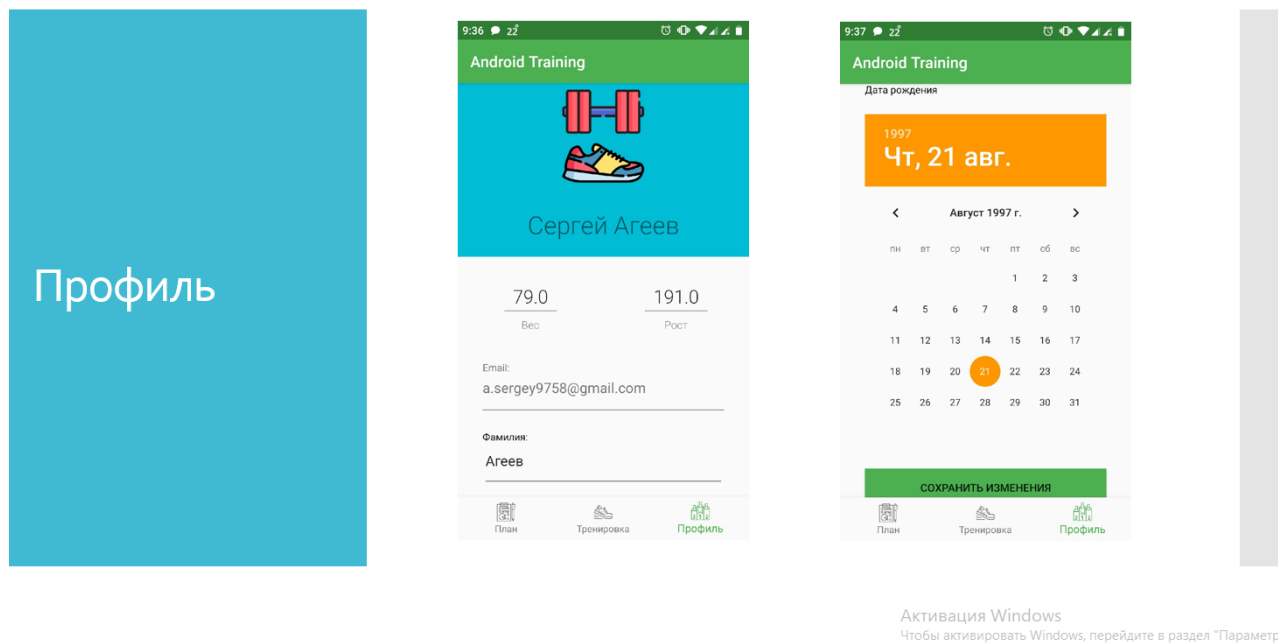


Рисунок 25 – Презентация – экран «Профиль»

					ПГУ 1.090301.01.001 ПЗ	Лист
						70
Изм.	Лист	№ докум.	Подпись	Дата		

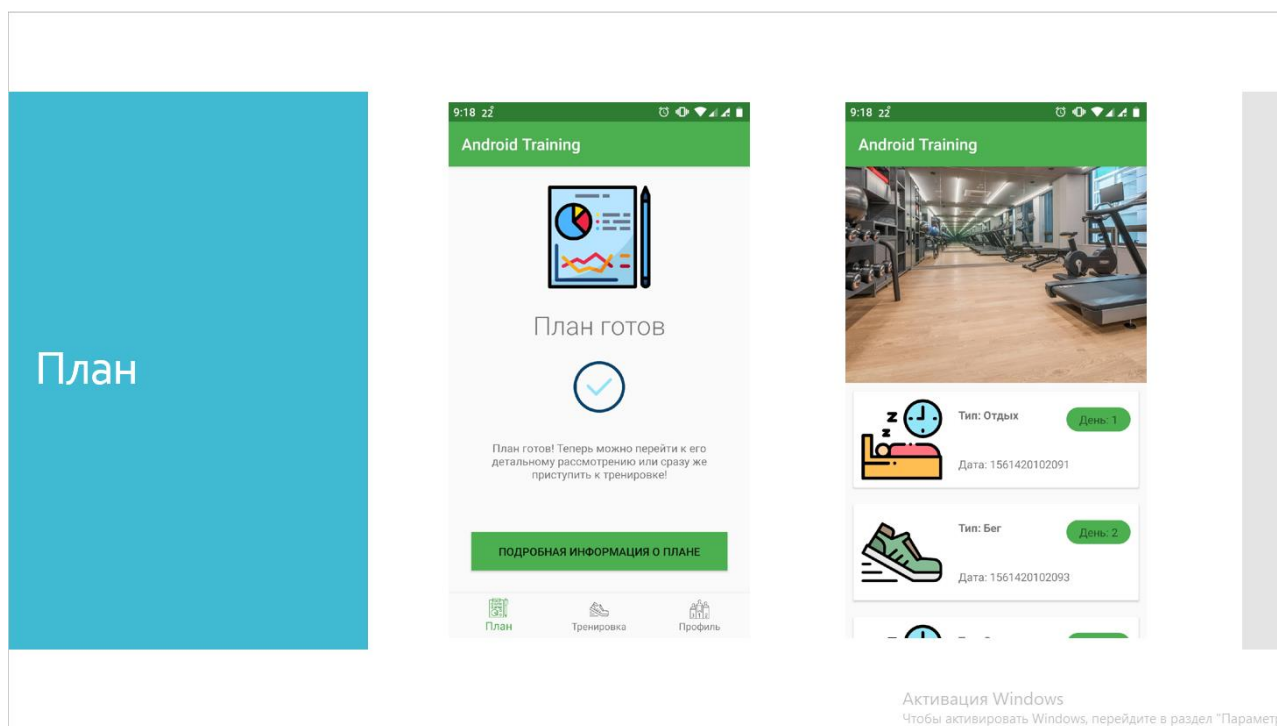


Рисунок 26 – Презентация – экран «План»

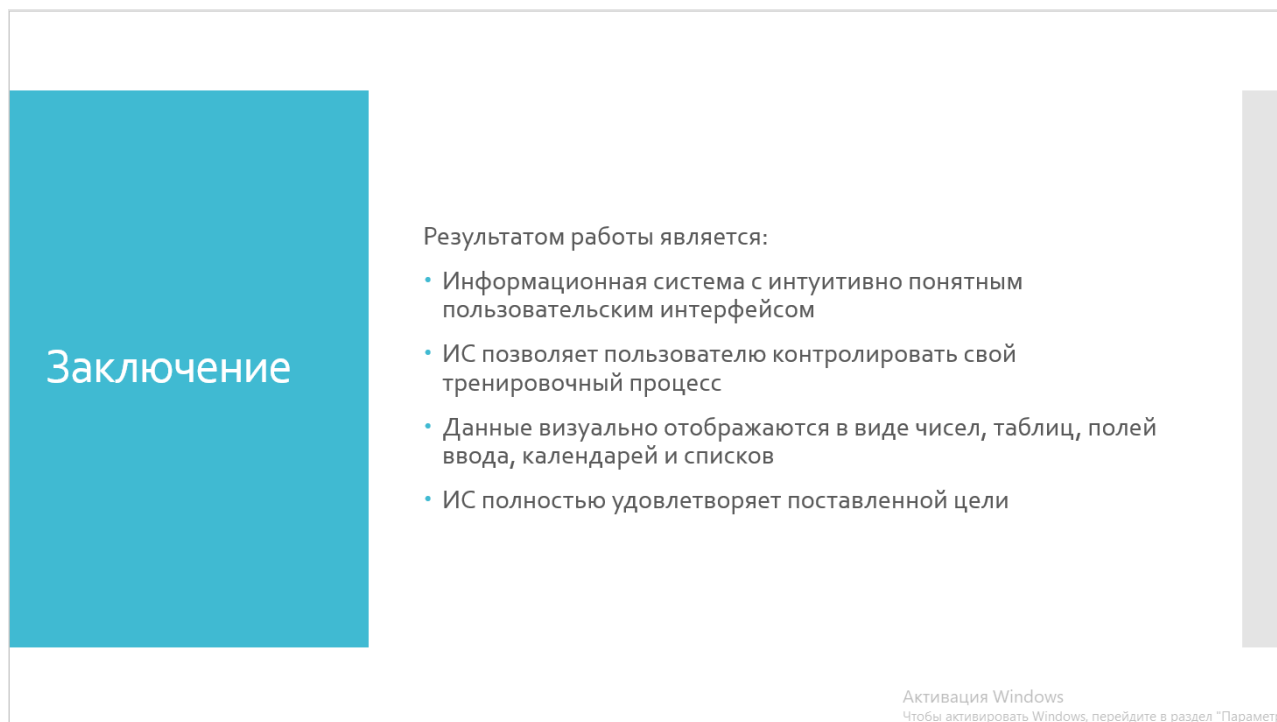
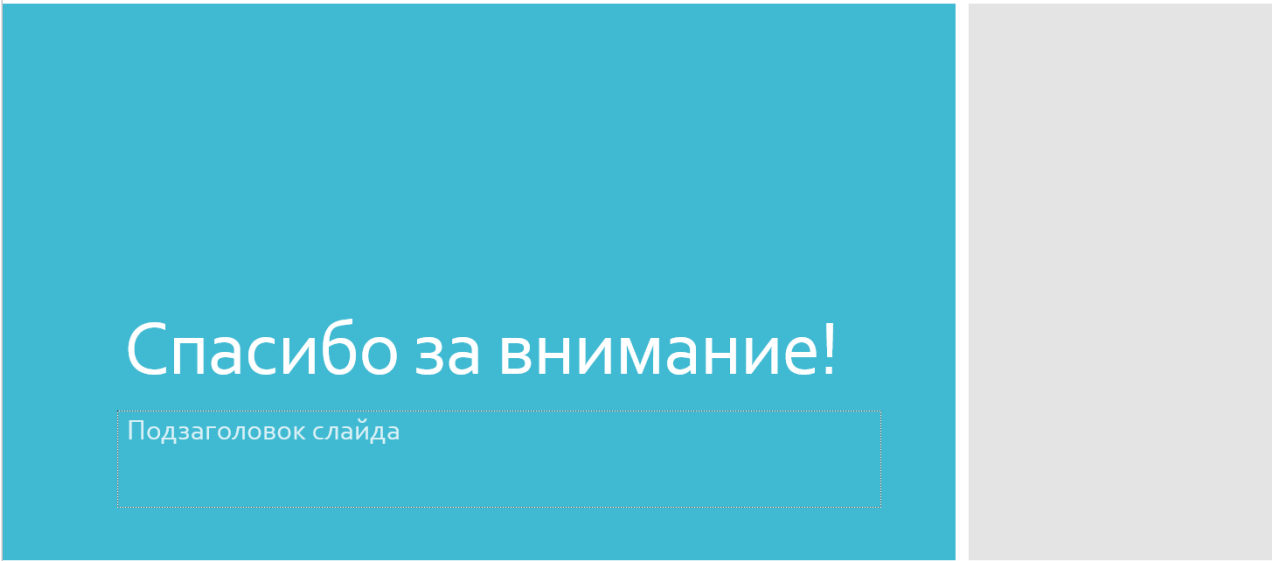


Рисунок 27 – Презентация – результаты работы

					ПГУ 1.090301.01.001 ПЗ	Лист
						71
Изм.	Лист	№ докум.	Подпись	Дата		



Активация Windows
Чтобы активировать Windows, перейдите в раздел "Параметры"

Рисунок 28 – Презентация – финальный слайд