

1
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: «Программное обеспечение средств вычислительной
техники и автоматизированных систем»

БАКАЛАВРСКАЯ РАБОТА

на тему

СТАРТАП-ПРОЕКТ. ПРИЛОЖЕНИЕ-ТРЕКЕР
«ПРИВЫЧКИ КРАСОТЫ» ДЛЯ ОС ANDROID

Студент

Папшев Никита Сергеевич
(подпись, дата) (ФИО полностью)

Руководитель

Юрова О.В.
(подпись, дата) (фамилия, инициалы)

Нормоконтролёр

Бычков А.С.
(подпись, дата) (фамилия, инициалы)

Работа допущена к защите (протокол зас. кафедры от 27.05.2024 г. № 11)

Заведующий кафедрой

М.А. Митрохин
(подпись) (фамилия, инициалы)

Работа защищена с отметкой отлично (протокол заседания ГЭК от

04.06.2024 № 6)

Секретарь ГЭК

Е.И. Гурин
(подпись) (фамилия, инициалы)

Пенза 2024

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

«Утверждаю»
Заведующий кафедрой
М.А. Митрохин
« 19 » 04 2024 г.

**ЗАДАНИЕ
НА БАКАЛАВРСКУЮ РАБОТУ**

1. Студент Папшев Никита Сергеевич гр. 20ВВП1 факультета вычислительной техники
2. Тема работы Стартап-проект. Приложение-трекер «Привычки красоты» для ОС Android
Тема утверждена приказом ПГУ №_405/о_____ от "19" _____ апреля _____ 2024 __ г.
3. Руководитель бакалаврской работы к.т.н., доцент кафедры ВТ ПГУ Юрова О.В.

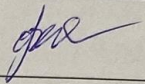
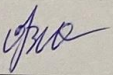
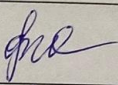
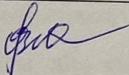
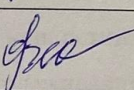
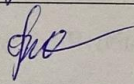
4. Задание на работу (назначение разработки, исходные данные и т.п.)
Разработать мобильное приложение «Привычки красоты», позволяющее отслеживать прогресс пользователя по выработки привычек. Составить бизнес-модель и финансовый план. Язык программирования Kotlin, среда разработки Android Studio. Программа должна обладать интуитивно понятным интерфейсом и использовать следующие технологии:

1. Firebase authentication
2. Cloud Firestore
3. Асинхронное программирование
4. Material design 3

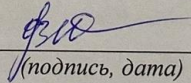
5. Перечень подлежащих разработке вопросов

Требуется произвести анализ предметной области: изучить существующие аналоги, изучить способы проектирования и разработки мобильного приложения. Выбрать язык программирования, среду разработки и базу данных.

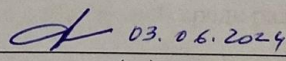
6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Изучение предметной области	45	19.04.2024 – 25.04.2024	
2	Анализ требований	35	26.04.2024 – 30.04.2024	
3	Выбор технологий, применяемых при проектировании и разработке	55	01.05.2024 – 06.05.2024	
4	Создание графического интерфейса	65	06.05.2024 – 14.05.2024	
5	Разработка мобильного приложения	84	14.05.2024 – 24.05.2024	
6	Оформление пояснительной записки	40	25.05.2024 – 31.05.2024	

Дата выдачи задания "19" апреля 2024 г.

Руководитель бакалаврской работы  Юрова О.В.
(подпись, дата)

Задание к исполнению принял студент  Папшев Н.С.
(подпись, дата)

Работу к защите допустить
Декан факультета  03.06.2024 Фионова Л.Р.
(подпись, дата)

Реферат

Бакалаврская работа содержит 118 листов, 45 рисунков, 1 формула, 6 таблица, 3 источника, 3 приложения

МОБИЛЬНОЕ ПРИЛОЖЕНИЕ, ТРЕКЕР, ТЕСТИРОВАНИЕ, FIREBASE CLOUD, FIREBASE AUTHENTICATION, ANDROID STUDIO, KOTLIN, ANDROID

Цель работы – разработать мобильное приложение, позволяющее отслеживать привычки пользователей.

Мобильное приложение должно позволять пользователям просматривать список собственных привычек, а также предоставлять инструменты для его редактирования, например, удаление или изменение данных конкретной привычки. По мимо этого должны присутствовать система отслеживания статистики выполненных целей и отправка уведомлений пользователю. Чтобы данные могли сохраняться и переноситься на другие устройства требуется реализовать регистрацию и авторизацию.

В конечном итоге было разработано мобильное приложение на языке программирования Kotlin, которое удовлетворяет всем поставленным задачам. Была реализована регистрация и сохранение данных пользователей с помощью Firebase Authentication и Firebase Cloud. Само приложение работает на устройства с ОС Android и разрабатывалось в специальной среде разработки Android Studio.

Изм.	Лист	№ докум.	Подпись	Дата	ПГУ1.090301.21.001 ПЗ		
Разраб.		Папшев Н.С.		29.05.24	Лит.	Лист	Листов
Провер.		Юрова О.В.		30.05.24		2	118
Н. Контр		Бычков А.С.		30.05.24	ФВТ, каф. ВТ гр. 20ВВП1		
Утверд.		Митрохин М.А.		30.05.24			

Стартап-проект. Приложение-трекер «Привычки красоты» для ОС Android

Содержание

Перечень сокращений и обозначений	5
Введение	6
1 Стартап-проект. Приложение-трекер «Привычки красоты» для ОС Android.7	
1.1 Анализ темы	7
1.2 Анализ аналогов	9
1.2.1 Мобильное приложение «Loop»	9
1.2.2 Мобильное приложение «Habitica»	10
1.2.3 Мобильное приложение «StickK»	13
1.2.4 Мобильное приложение «Habitshare»	14
1.2.5 Вывод	17
1.3 Постановка задачи	18
2 Выбор инструментов разработки	19
2.1 Выбор языка программирования	19
2.1.1 Язык программирования «Java»	19
2.1.2 Язык программирования «Kotlin»	19
2.1.3 Язык программирования «C++»	20
2.1.4 Вывод	20
2.2 Выбор среды разработки	20
2.3 Выбор базы данных	21
3 Архитектура приложения	22
3.1 Архитектурный паттерн «Clean Architecture»	22
4 Реализация приложения	24
4.1 Описание функционала	24
4.2 Описание работы приложения	28
4.3 Основная реализация приложения	30
4.3.1 Presentation Layer	30
4.3.2 Domain Layer	32
4.3.3 Data Layer	33

5 Тестирование	36
5.1 Авторизация.....	36
5.2 Регистрация.....	41
5.3 Взаимодействие с привычками.....	43
5.4 Просмотр статистики.....	50
5.5 Взаимодействие с пользователем.....	53
5.6 Вывод.....	55
6 Безопасность жизнедеятельности.....	56
6.1 Правила и техника безопасности при работе с ПК.	56
6.2 Правила и техника безопасности при работе со смартфоном.....	57
7 Стартап проект	59
7.1 Сегменты ЦА	59
7.2 Ключевые процессы.....	60
7.3 Каналы взаимодействия	62
7.4 Источники дохода	63
7.5 Ключевые ресурсы	64
7.6 Риски проекта	66
Заключение	67
Список используемых источников	68
Приложение А UML-диаграммы	69
Приложение Б Листинг программы	75
Приложение В Презентация.....	111

Перечень сокращений и обозначений

В данной работе применяются следующие сокращения и обозначения.

ОС – операционная система

ПК – персональный компьютер

ПО – программное обеспечение

СУБД – система управления базами данных

IDE – (от англ. Integrated Development Environment) – интегрированная среда разработки

JSON – (от англ. JavaScript Object Notation) – текстовый формат обмена данными

NoSQL – (от англ. not only SQL) – широкий класс разнородных СУБД

RPG – (role playing game) – компьютерная ролевая игра

UI – (от англ. User Interface) – пользовательский интерфейс

UML – (от англ. Unified Modeling Language) – унифицированный язык моделирования

					<i>ПГУ1.090301.21.001 ПЗ</i>	Лист
						5
Изм.	Лист	№ докум.	Подпись	Дата		

Введение

В современном мире люди все чаще начинают следить за своим здоровьем, питанием и состоянием собственного тела. Но для этого человеку приходится прилагать усилия, причем чаще всего эти усилия заключаются в систематических повторениях каких-либо действий. При должном внимании и ежедневной работе у человека вырабатываются полезные привычки. Трекер-привычек может помочь людям отслеживать своей прогресс и напоминать им о не выполненных задачах.

Для удобства пользователей в качестве трекера-привычек можно использовать мобильное приложение, так как мобильное устройство всегда находится под рукой у человека. Приложение поможет пользователям отслеживать свой прогресс к достижению собственных целей, а также будет напоминать пользователям в удобное для них время о невыполненных задачах.

Цели выпускной квалификационной работы:

- исследовать тему использования данного продукта;
- определить функциональность и интерфейс мобильного приложения;
- разработать современное мобильное приложение для учета собственных привычек;
- провести тестирование ПО;
- оценить эффективность внедрения данного продукта в малый и средний бизнес;
- предложить рекомендации по развитию и расширению функционала для частных случаев разработки данного приложения;
- проанализировать полученные результаты и сделать выводы о потенциале разработки данного приложения.

1 Стартап-проект. Приложение-трекер «Привычки красоты» для ОС Android.

1.1 Анализ темы

В наше время люди активно используют уходовую косметику чтобы следить за своим здоровьем. Вот несколько причин, почему её использование полезно:

- увлажнение и питание. Уходовая косметика содержит компоненты, которые помогают увлажнять и питать кожу, волосы и ногти;

- защита от вредных факторов. Косметические средства, такие как солнцезащитные кремы и антиоксиданты, могут помочь защитить кожу от ультрафиолетового излучения, загрязнений и других вредных факторов окружающей среды;

- стимуляция регенерации клеток. Некоторые компоненты косметики способствуют обновлению клеток кожи, что помогает сократить морщины, улучшить текстуру кожи и стимулировать её естественный процесс регенерации;

- лечение конкретных проблем. Уходовая косметика может быть специально разработана для решения конкретных проблем, таких как акне, пигментация, сухость или чувствительность кожи.

В целом, уходовая косметика играет ключевую роль в поддержании здоровья и красоты кожи, волос и ногтей, а также предотвращает различные проблемы и замедляет процессы старения.

Но зачастую люди забывают о ежедневных процедурах или откладывают их на потом и в конечном итоге забрасывают собственное лечение. Для того чтобы это исправить нужно выработать ряд полезных привычек.

С точки зрения нейрофизиологии привычка — это определенная нейронная связь и для создания этой связи нужны три компонента:

- триггер — мысль, желание;

- действие — покупка, установка приложения, пробежка;

					ПГУ1.090301.21.001 ПЗ	Лист
						7
Изм.	Лист	№ докум.	Подпись	Дата		

– результат – вознаграждение, которое приносит дофамин.

Для того чтобы сформировать привычку требуется немалое количество времени. Специалисты выделяют примерно от 18 до 254 дней на выработку привычки. Конкретный срок будет зависеть от многих факторов, например, мотивация, сложность привычки, временных ресурсов и загруженности человека.

Все больше пользователей обращается к трекерам привычек, помогающим им отслеживать свои действия. Есть несколько причин, которые могут подтолкнуть пользователя на установку такого рода приложений:

- сиюминутная награда, побуждающая повторять действия вновь и вновь;
- подкрепление мотивации;
- отслеживание прогресса;
- с трекером шанс достигнуть успеха намного выше, чем без него.

Существует два варианта трекера:

- таблицы;
- приложения.

Фактический любой трекер является таблицей, которую можно расчертить в тетради. В первой колонке вписываются привычки, а в последующих дни месяцев.

Приложения-трекеры являются более удобными аналогами бумажных записей. В них можно одновременно вести несколько таблиц, просматривать статистику и настраивать напоминания.

Трекер привычек красоты – это приложение, в которое пользователь может вносить список полезных привычек, связанных с уходом за здоровьем кожи, волос и ногтей, отслеживать выполнение поставленных целей и статистику. Тема разработки программы-трекера актуальная в наше время, когда люди начали активно следить за своим питанием, здоровьем, избавлением от вредных привычек.

1.2 Анализ аналогов

На данный момент уже представлено много качественных реализаций трекеров, в которых можно вести заметки и отмечать выполненные действия. В данном пункте рассмотрены такие популярные приложения как:

- Loop;
- Habitica;
- StickK;
- Habitshare.

Проведен анализ и выявлены слабые и сильные стороны каждого из приложений.

1.2.1 Мобильное приложение «Loop»

Интерфейс приложения представлен на рисунке 1 и 2.

Удобное и простое приложение для отслеживания и формирования здоровых привычек, разработанное только для устройств на базе операционной системы Android.

Достоинства:

- отслеживание привычек. Позволяет создать список привычек, которые пользователь захочет развивать;
- статистика. Построение графика, который показывает успехи пользователя;
- оповещения. Можно настроить напоминания, чтобы получать уведомления о выполнении привычек;
- удобство. У приложения минималистичный интерфейс, который легок в использовании.

Недостатки:

- есть только английская локализация;
- нет установки времени и дня для выполнения задания.

					ПГУ1.090301.21.001 ПЗ	Лист
						9
Изм.	Лист	№ докум.	Подпись	Дата		

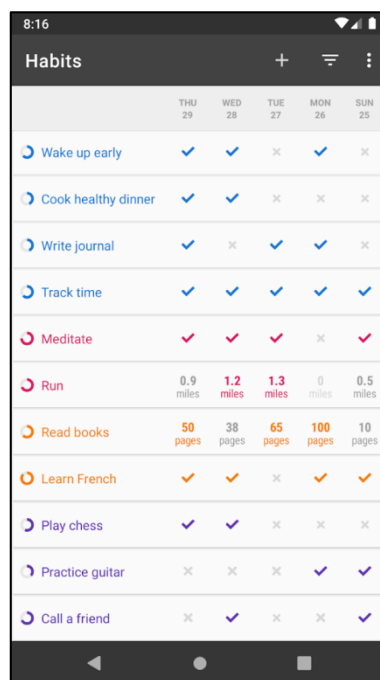


Рисунок 1 - Главная страница со списком привычек



Рисунок 2 - Страница со статистикой конкретной привычки

1.2.2 Мобильное приложение «Habitica»

Интерфейс приложения представлен на рисунке 3 и 4.

Трекер выполнен в стиле RPG игры, что придает отслеживанию определенный азарт. Соответственно, появляется дополнительный стимул для того, чтобы закрепить полезные привычки. В приложении есть такие игровые механики как: достижения, миссии, игровое золото, кастомизация персонажа. По сути, данное приложение является простой видеоигрой.

Достоинства:

- задания. Отслеживание привычек выполнено в виде заданий и при их выполнении пользователь (игрок) получает награду;
- игровой подход. Приложение полностью выполнено с использованием игровых механик, таким образом достижение целей и формирование привычек превращается в игру;
- групповые испытания. Пользователи могут объединяться и работать над общими целями и получать дополнительные стимулы к достижению привычек.

Недостатки:

- может не подойти тем, кто не любит игровые составляющие в приложениях;
- нет стандартных функций трекера привычек, например, календаря, статистики;
- перегруженное приложение: слишком много функций в которых нужно разбираться. Неподготовленный пользователь может запутаться в том, что он делает.

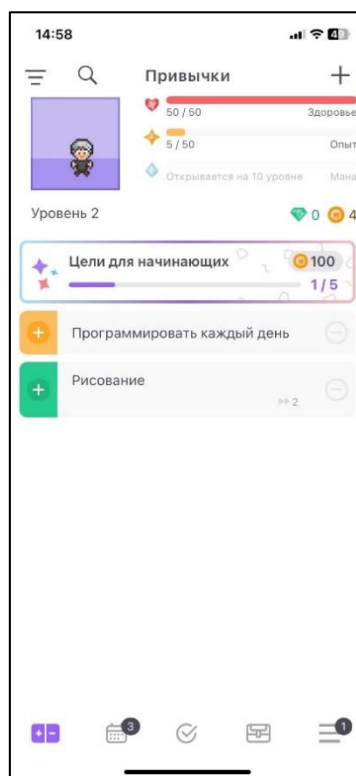


Рисунок 3 – Главная страница со списком привычек

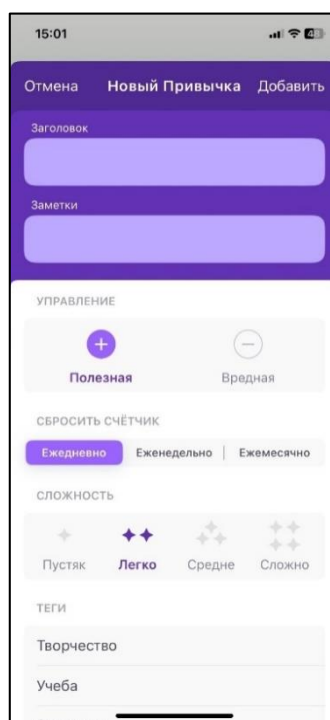


Рисунок 4 - Страница с добавлением новой привычки

1.2.3 Мобильное приложение «StickK»

Интерфейс приложения представлен на рисунке 5.

Это приложение было разработано профессорами и учеными Йельского университета, которые разработали теорию и методику борьбы с прокрастинацией и ленью. Главной фишкой трекера является договор с обязательствами, который пользователь заключает с самим собой. Также для подтверждения правдивости отчетов можно выбрать человека, который будет следить за вами и выполнять роль судьи.

Достоинства:

- оригинальная методика мотивации. Пользователи могут устанавливать финансовые стимулы, связанные с достижением своих целей;
- мониторинг прогресса. Можно отслеживать свой прогресс в достижении целей, отмечая выполненные действия;
- поддержка сообщества. В StickK можно находить единомышленников, которые тоже стремятся к достижению схожих целей.

Недостатки:

- только английская локализация;
- может не подойти тем, кто не любит контракты, договоры и прочий официоз.

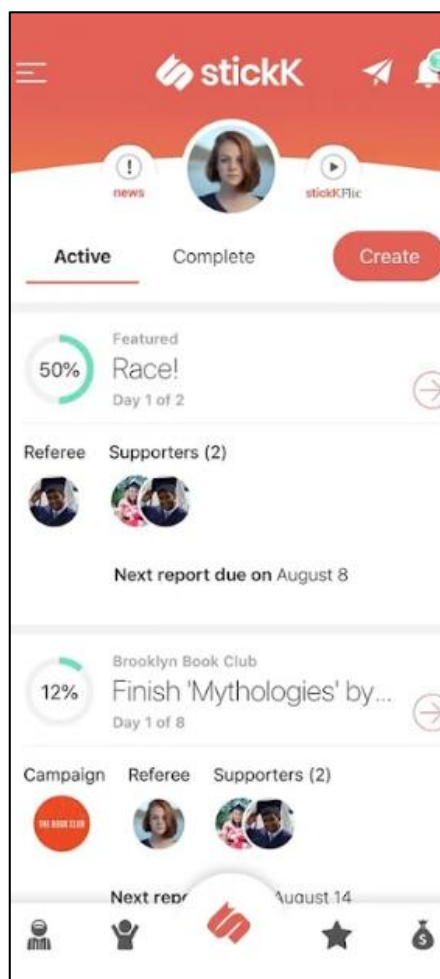


Рисунок 5 - Главная страница со списком привычек

1.2.4 Мобильное приложение «Habitshare»

Интерфейс приложения представлен на рисунке 6 и 7.

Данное приложение включает в себя как трекер привычек, так и своеобразную социальную сеть. В Habitshare можно отслеживать привычки друзей, делиться своими успехами и обмениваться сообщениями с другими пользователями.

Достоинства:

- отслеживание привычек. Позволяет создать список привычек и отслеживать их выполнение;
- статистика. Приложение позволяет анализировать свой прогресс на основе графиков;

– оповещения. Habitshare отправляет пользователям оповещения, чтобы помочь им оставаться на правильном пути;

– удобство. Приложение простое и не требует дополнительных знаний при использовании;

– поддержка сообщества. В приложении можно общаться с другими людьми, лайкать прогресс пользователей и делиться своими успехами.

Недостатки:

В приложении отсутствует русская локализация.

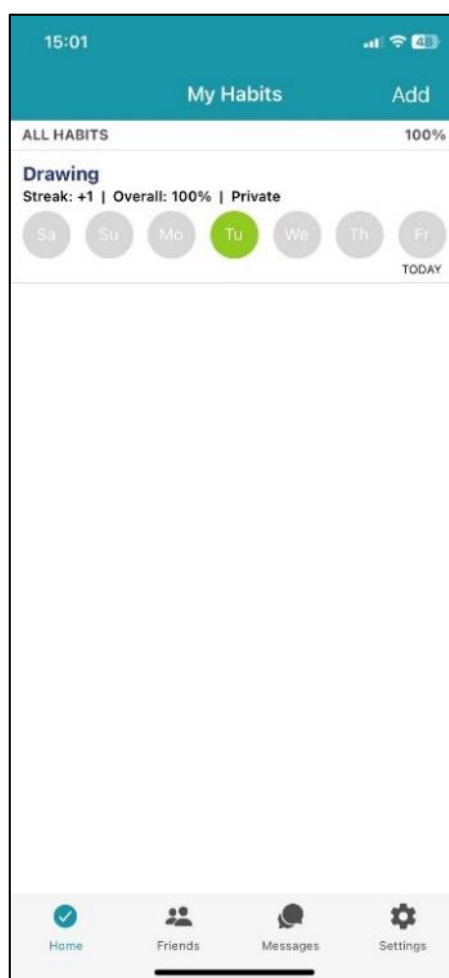


Рисунок 6 – Главная страница со списком привычек



Рисунок 7 - Страница отдельной привычки

1.2.5 Вывод

Таблица 1 – Достоинства и недостатки приложений-трекеров

	Loop	Habitica	StickK	Habitshare
Русификация	-	+	-	-
Статистика	+	-	-	+
Удобство	+	-	-	+
Отслеживание привычек	+	+	+	+
Напоминания	+	+	+	+
Оригинальная концепция	-	+	+	-
Социальная поддержка	-	+	+	+

Проанализировав существующие приложения можно отметить, что базовый функционал трекера у аналогов почти что не меняется, а меняется только то, как преподносят его. Также важно понимать, что приложение — это только удобный способ для выработки привычки. Оно должно быть простым, гибким и удобным для самого пользователя, поэтому он сам в праве выбирать более подходящее приложение для себя. Можно заметить, что приложений, специализированных на отслеживании приема косметических средств на рынке не представлено. Поэтому создание трекера, способного отслеживать использование косметических и уходовых средств, является оправданным.

1.3 Постановка задачи

Необходимо разработать приложение-трекер для телефонов на базе операционной системы Android, позволяющее вырабатывать привычки для ухода за собственным телом на основе заранее проведенного анкетирования, а также отслеживать прогресс использования косметических средств. Согласно статистике, Android является неоспоримым лидером по количеству смартфонов и занимает 70% от глобального рынка мобильных устройств [1]. Именно поэтому для разработки была выбрана эта операционная система.

В приложении должно быть реализовано:

- регистрация пользователя;
- настройку и добавление уведомлений;
- добавление привычек с выбором типа привычки;
- редактирование привычки (изменение названия, описания и времени);
- просмотр статистики;
- интуитивно понятный интерфейс с использованием Material 3.

Минимальные системные требования:

- операционная система Android 11;
- наличие интернета.

Основная цель проекта – выработка привычек для ухода за своим здоровьем за счет формирования нейронных связей мозга и получения дофамина за получение вознаграждения в трекере.

2 Выбор инструментов разработки

Для того чтобы разработать качественный продукт и оптимизировать процесс разработки следует выбрать правильные инструменты. Это помогает избежать проблем при производстве и использовании продукта, а также позволяет сократить время и средства на его создание.

2.1 Выбор языка программирования

Android является одной из самых популярных операционных систем для мобильных устройств. Поэтому очень важно выбрать правильный язык программирования, который будет наиболее эффективным инструментом и в полном объеме поможет разработать безопасное и удобное приложение.

Языки, которые позволяют писать приложения для телефонов под управлением Android представлены ниже:

- Java;
- Kotlin;
- C++.

Чтобы решить какой же язык подходит больше всего проведем анализ и выявим особенности каждого из них.

2.1.1 Язык программирования «Java»

Java является официальным языком для разработки программного обеспечения под Android, а также языком который используется в качестве основного в Android Studio. Кроме того, Java имеет огромное сообщество разработчиков и большое количество библиотек и фреймворков, что делает этот язык хорошим вариантом для разработки мобильного приложения [2].

2.1.2 Язык программирования «Kotlin»

Kotlin, как и Java, также является официальным языком для программирования программного обеспечения для мобильных устройств, но с недавних пор стал предпочтительнее в разработке нежели Java.

Kotlin является оберткой над Java, но с лучшей типизацией, компактным синтаксисом, наличием корутин, поддержкой функционального программирования, что позволяет создавать программы эффективнее, быстрее и компактнее. Сейчас Kotlin имеет большое сообщество, которое готово помогать и делиться опытом с начинающими разработчиками. Исходя из всего выше сказанного, Kotlin является отличным выбором и идеально подойдет для разработки мобильного приложения [3].

2.1.3 Язык программирования «C++»

Из-за своих возможностей и высокой производительности C++ также плотно вошел в мир мобильной разработке, но уже для более узконаправленных приложений таких как 3D-игры или для работы с низкоуровневыми компонентами Android. Но из-за высокой сложности языка, отсутствия сборщика мусора и небезопасных указателей разрабатывать на нем программы становится проблематичной задачей.

2.1.4 Вывод

Проанализировав все варианты было принято решение выбрать Kotlin в качестве основного языка программирования для разработки приложения-трекера, потому что, используя его получится написать компактное, безопасное и не затратное приложение.

2.2 Выбор среды разработки

Рассмотрим самую популярную и поддерживаемую IDE для разработки Android приложений - Android Studio, которая была специально разработана и признана официальной средой для программирования программного обеспечения для мобильных устройств. Так как эта IDE была создана на основе IntelliJ IDEA она имеет такой же удобный функционал для редактирования кода, отслеживания файлов в проекте, встроенный эмулятор мобильного устройства и интуитивно понятный интерфейс [4].

Также для разработки мобильных приложений могут использоваться такие IDE как IntelliJ, Eclipse и даже Visual Studio, но именно для разработки приложения на базе операционной системы Android они подходят в меньшей степени, поэтому было решено выбрать Android Studio.

2.3 Выбор базы данных

В качестве базы данных была выбрана Cloud Firestore. Это облачная NoSQL база данных входящая в состав огромной платформы Firebase от Google. Cloud Firestore привлекательна тем что позволяет синхронизировать данные в реальном времени, удобно и масштабируемо хранить данные в виде коллекций и документов, а также предоставляет офлайн-доступ к данным, если у пользователя не доступен интернет [5].

В Firestore данные хранятся в виде пары «ключ-значение» в JSON файлах, что позволяет хранить и обрабатывать данные просто и быстро. Такой формат хранения идеально подходит для сохранения данных о привычках конкретного пользователя.

Также в проект был интегрирован Authentication, еще один сервис от Firebase. Он позволяет регистрировать аккаунты пользователей с помощью разных способов, а также сохранять их, что в комбинации с Cloud Firestore позволяет создать удобную и функциональную систему для пользователя.

3 Архитектура приложения

3.1 Архитектурный паттерн «Clean Architecture»

Для того чтобы разработать хорошо тестируемое, расширяемое и надежное приложение, требуется разделять части программы на какие-то логические блоки называемыми слоями. Именно про данный принцип проектирования приложений написал Роберт К. Мартин в своей статье [6]. Так он разделяет структуру приложения на 4 компонента:

- сущности;
- сценарии использования;
- адаптеры интерфейсов;
- фреймворки и драйверы.

Сущности по своей сути являются бизнес объектами, которые содержат в себе правила и понятия для работы всего приложения.

Сценарии использования требуются для того чтобы определить бизнес-логику приложения. Благодаря сценариям сущности могут взаимодействовать между собой и выполнять операции с задачами, связанными конкретно с данным приложением.

Адаптеры интерфейсов нужны для того чтобы преобразовывать один вид информации в другой. Именно они являются прослойкой между внутренней частью приложения и его логикой и внешней частью, например, слоем пользовательского интерфейса.

Последний слой можно легко изменить или полностью заменить при устаревании, потому что именно на этом слое хранятся все внешние компоненты, такие как базы данных, UI, фреймворки.

Разделение кода на отдельные части позволяет добиться независимости от внешних компонентов, легкой тестируемости, простой заменимостью элементов,

а также помогает обеспечить долгосрочную поддержку проекта и его безопасность.

Также следует уточнить что Google для разработки приложений рекомендует использовать архитектуру, основанную на «чистой архитектуре» [7]. Поэтому исходя из всего вышесказанного было принято решение использовать данную архитектуру в разработке проекта.

					ПГУ1.090301.21.001 ПЗ	Лист
						23
Изм.	Лист	№ докум.	Подпись	Дата		

4 Реализация приложения

Перед началом разработки ПО следует разобраться в архитектуре и функционале разрабатываемого приложения. Так нам потребуется рассмотреть такие важные аспекты разработки приложения как описание работы приложения, функционал, взаимодействие с сервером, а также общую структуру приложения.

4.1 Описание функционала

В данной программе предусмотрена регистрация пользователь, поэтому чтобы воспользоваться полным функционалом, каждый пользователь должен зарегистрировать свой аккаунт, либо войти в уже имеющийся. Только после регистрации ему станут доступны все возможности приложения такие как:

- добавление привычек;
- редактирование привычек;
- удаление привычек;
- настройка оповещений;
- отметка выполненных действий;
- просмотр статистики.

Данные варианты использования представлены на диаграмме на рисунке 8.

Диаграмма вариантов использования

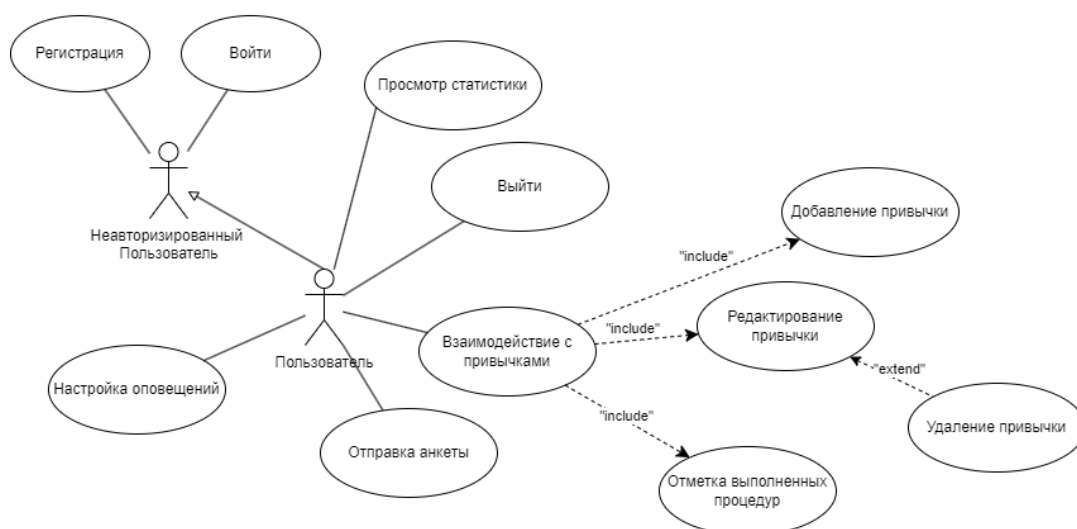


Рисунок 8 – UML диаграмма вариантов использования для пользователей

Чтобы более точно проанализировать взаимодействие пользователя с приложением составим диаграммы деятельности, представленные на рисунках 9-11.

Так пользователь начинает взаимодействие с приложением через его запуск, после чего клиент проверяет был ли до этого авторизован пользователь или нет. Если да, то сразу же выполняется вход и открывается главный экран со списком привычек (рисунок 10), если все же пользователь не был авторизован, то ему предлагается либо ввести свои данные, либо зарегистрироваться (рисунок 9).

После успешного входа пользователь может воспользоваться всем функционалом приложения (рисунок 11) таким как: редактирование, добавление и удаление привычек. Чтобы добавить или изменить привычку следует ввести ее название, тип, время напоминания и описание, после чего она добавится в общий список.

Пользователь также может выйти из аккаунта или в любой момент времени прекратить работу приложения

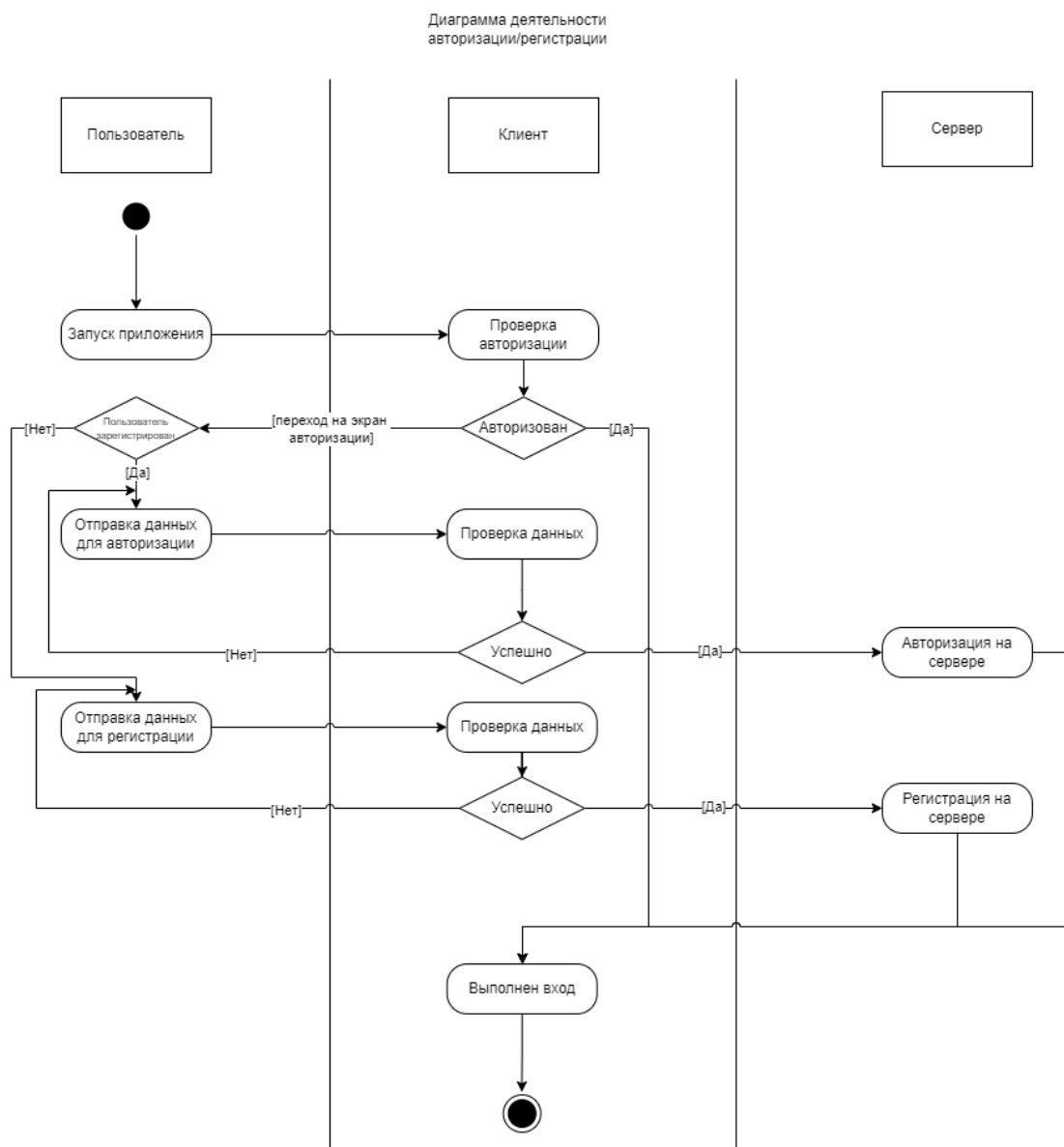


Рисунок 9 – UML диаграмма деятельности для
авторизации/регистрации

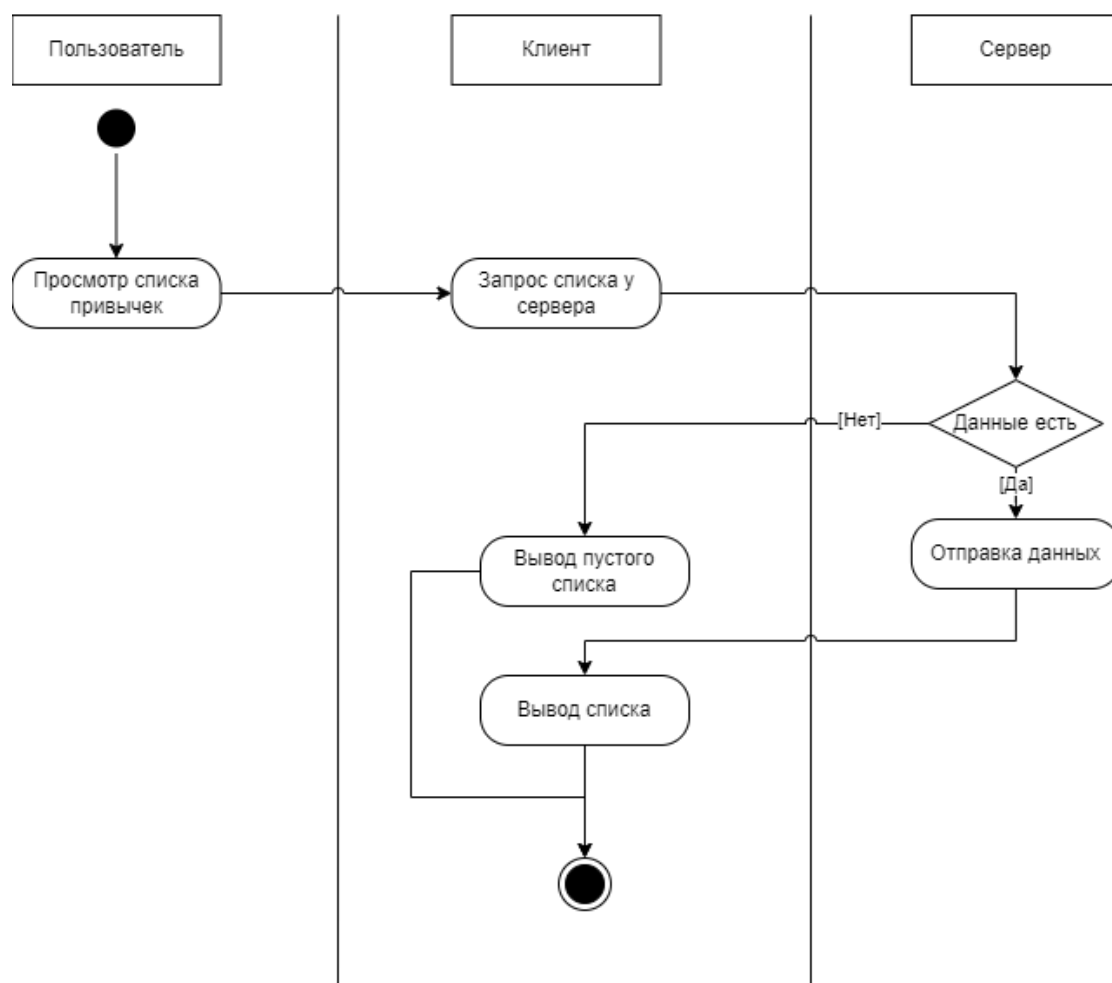


Рисунок 10 – UML диаграмма деятельности для просмотра списка привычек

Диаграмма деятельности взаимодействия с одной привычкой

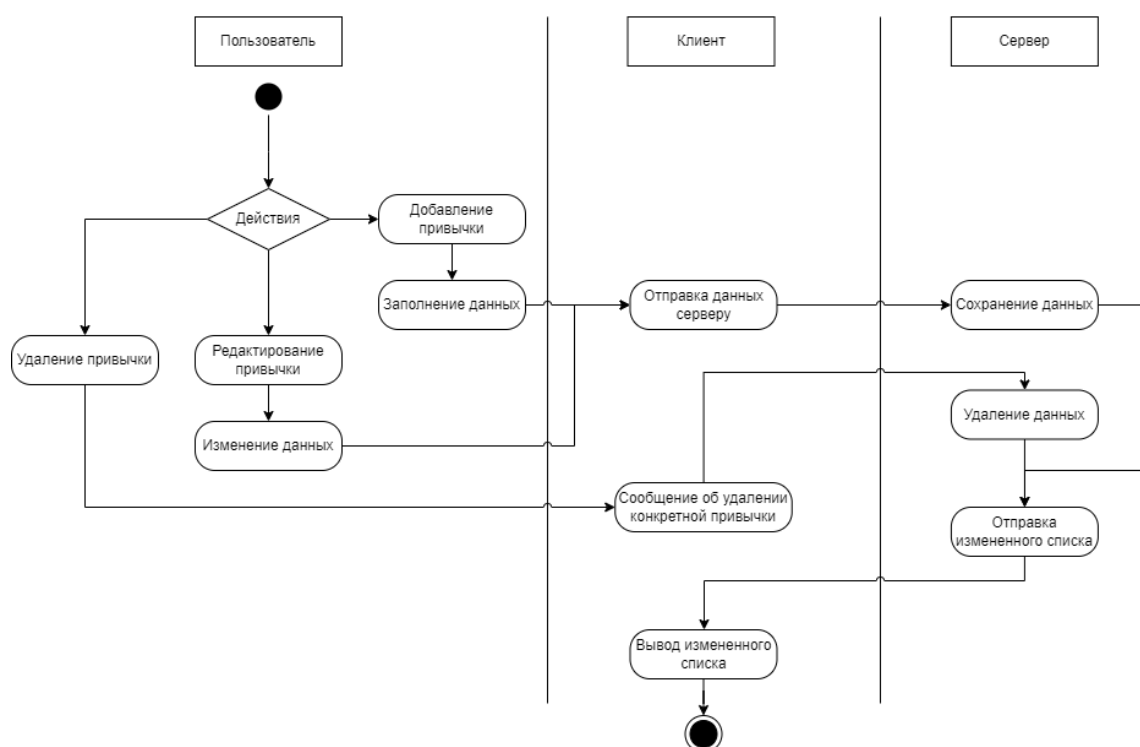


Рисунок 11 – UML диаграмма деятельности взаимодействия с одной привычкой

4.2 Описание работы приложения

Пользователь взаимодействует только с клиентской частью приложения. Для того чтобы начать пользоваться ПО требуется запустить его на подходящей версии Android, после чего зайти в свой аккаунт или зарегистрироваться, то есть создать новый. После этого клиент сравнит данные с данными сервера и при успешной авторизации отобразит для пользователя его список привычек. Пользователь, инициируя любое взаимодействие с привычками, отправляет данные клиенту. Клиент в свою очередь отправит или изменит (удалит) данные на сервере и после успешной операции отобразит для пользователя обновленный UI.

Все выполняемые действия как пользователя, так и клиента с сервером изображены на диаграмме последовательности на рисунке 12. А физическое

расположение каждого из компонентов данной системы представлено на рисунке 13.

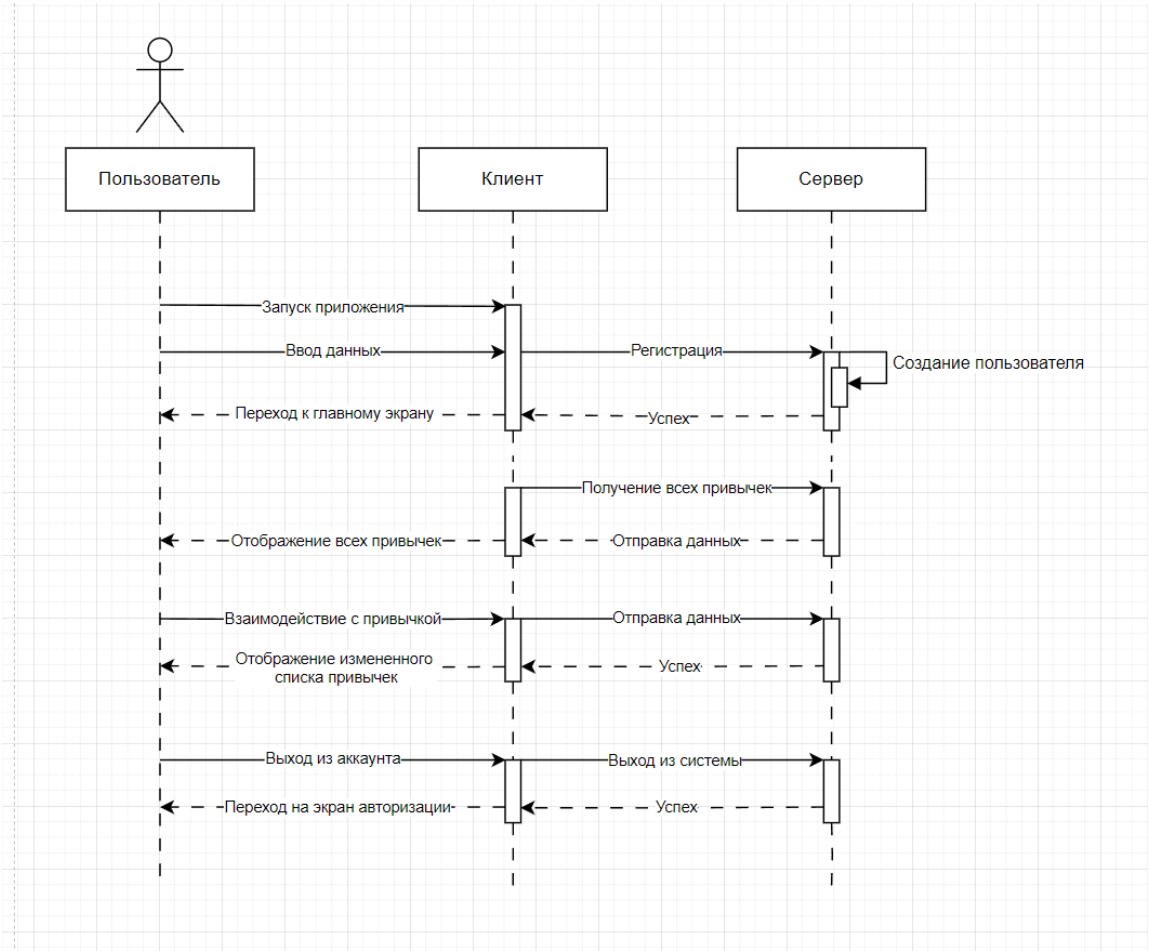


Рисунок 12 – UML диаграмма последовательности приложения

Диаграмма развертывания

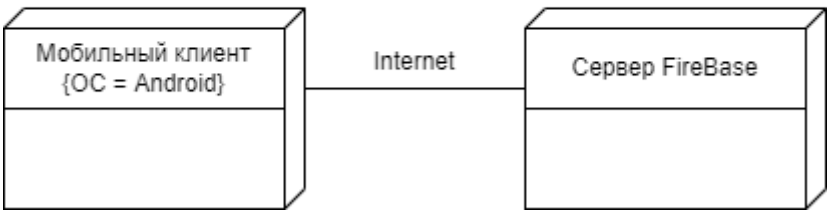


Рисунок 13 – UML диаграмма развертывания

4.3 Основная реализация приложения

Так как была выбрана «Чистая архитектура», то следует поделить приложение на несколько частей и рассмотреть их по отдельности.

Приложение было поделено на такие слоя как:

- presentation Layer;
- domain Layer;
- data Layer.

В слое представлений (Presentation Layer) находятся все классы, которые отвечают за обработку, создание и отображение UI. На данном слое будут находиться Activity и ViewModel. Activity требуется для отображения пользовательского интерфейса, а ViewModel будет прослойкой между слоем представлений и слоем домена.

Следующий слой, слой домена будет отвечать за бизнес логику приложения. Он будет включать в себя Use Cases и Entity. Классы Use Cases будут отвечать за выполнение определенной логики в приложении.

Последний слой отвечает за управление данными и будет проводить всю работу по взаимодействию с базой данных и сервером. Этот слой будет содержать Repositories, классы, которые будут предоставлять методы для получения данных из различных источников.

Посмотрим на конкретный пример реализации приложения на основе авторизации пользователя.

4.3.1 Presentation Layer

На данном уровне у нас будет только два класса это LoginActivity и LoginViewModel. LoginActivity создает весь интерфейс и хранит внутри себя объект класса LoginViewModel, который постоянно наблюдает за изменением данных. LoginViewModel содержит две переменные при изменении которых вызывается обновление UI, также если пользователь уже был авторизован то происходит переход на главный экран.

Листинг 1. Класс LoginActivity:

```
viewModel.loginState.observe(this@LoginActivity, Observer {
    val loginState = it ?: return@Observer
    if (loginState.isLogin) {
        val intent = Intent(this@LoginActivity,
            HabitsActivity::class.java)
        startActivity(intent)
        finish()
    }
    btnSignIn.isEnabled = loginState.isValidate
    binding.inputLayoutEmail.error = loginState.errorEmailMessage
    binding.inputLayoutPass.error = loginState.errorPasswordMessage
})
viewModel.result.observe(this@LoginActivity, Observer {
    val loginResult = it ?: return@Observer
    if (loginResult.error != null) {
        Toast.makeText(applicationContext, loginResult.error,
            Toast.LENGTH_SHORT).show()
    }
    if (loginResult.success != null) {
        val intent = Intent(this@LoginActivity,
            HabitsActivity::class.java)
        startActivity(intent)
        finish()
    }
})
```

Переменная LoginState требуется для того чтобы следить за состоянием пользовательского интерфейса.

Листинг 2. Класс LoginState:

```
data class LoginState (
    val emailInput: String = "",
    val passwordInput: String = "",
    val isEmailValidate: Boolean = false,
    val isPassValidate: Boolean = false,
    val isValidate: Boolean = false,
    val errorPasswordMessage: String? = null,
    val errorEmailMessage: String? = null,
    val isLogin: Boolean = false
)
```

Как только данные меняются в полях отрабатывает UseCase, который выполняет бизнес логику. Как только UseCase отработает с положительным результатом нам станет доступна кнопка «Войти». По ее нажатию выполняется метод signIn() класса LoginViewModel

Листинг 3. Метод LoginViewModel signIn():

					ПГУ1.090301.21.001 ПЗ	Лист
						31
Изм.	Лист	№ докум.	Подпись	Дата		

```

fun signIn(email: String, password: String) {
    viewModelScope.launch {
        accountControlRepository.signIn(email, password)
        if (accountControlRepository.hasUser()) {
            _Result.value = Result(success = "Completed")

        } else {
            _Result.value = Result(error = "Error
Completed")
            _loginState.value = LoginState()
        }
    }
}

```

Эта функция делегирует ответственность уже более высокому уровню архитектуры.

Также в LoginViewModel есть метод который вызывается при изменении данных и в котором будут срабатывать UseCase.

Листинг 4. Метод LoginViewModel onInputChange():

```

fun onInputChange(email: String, pass: String) {
    _loginState.value = LoginState(emailInput = email,
passwordInput = pass)
    checkInput()
}

```

4.3.2 Domain Layer

На этом уровне находится ValidateLoginInputUseCase, который вызывается в слое представлений при изменении данных. Он проверяет введенные данные на правильность с помощью специального Enum класса ValidateInputType.

Листинг 5. Enum класс ValidateInputType:

```

enum class ValidateInputType {
    EmptyPass,
    EmptyEmail,
    NotEmail,
    SmallPass,
    EmptyConfirm,
    NotConfirm,
    Validate
}

```

Листинг 6. Класс ValidateLoginInputUseCase:

```

class ValidateLoginInputUseCase {
    operator fun invoke(email: String, password: String) :
ValidateInputType {
        if (email.isEmpty()) {

```

```

        return ValidateInputType.EmptyEmail
    }
    if (!Patterns.EMAIL_ADDRESS.matcher(email).matches()) {
        return ValidateInputType.NotEmail
    }
    if (password.isEmpty()) {
        return ValidateInputType.EmptyPass
    }
    if (password.length < 6) {
        return ValidateInputType.SmallPass
    }

    return ValidateInputType.Validate
}
}

```

Так как слои должны быть не связными, то класс домена выполняет функцию некоторого узла между слоем преставлений и слоем данных. Именно поэтому в данном слое также находится интерфейс AccountControlRepository, который потребуется для реализации класса работы с базой данных в слое данных.

Листинг 7. Интерфейс AccountControlRepository:

```

interface AccountControlRepository {
    var user: User?
    fun hasUser(): Boolean
    suspend fun signIn(email: String, password: String)
    suspend fun signUp(email:String, password: String)
    fun signOut()
}

```

4.3.3 Data Layer

В этом классе происходит работа с базой данных Firestore. Вся работа с ней происходит в специальном классе AccountControlRepositoryImpl, который является реализацией интерфейса AccountControlRepository. Именно здесь находится метод менно здесь находится метод signIn(), который используется для авторизации пользователя в приложении.

Листинг 8. Класс AccountControlRepositoryImpl:

```

class AccountControlRepositoryImpl : AccountControlRepository {
    override var user: User? = null
    get() = User(Firebase.auth.currentUser?.uid,
Firebase.auth.currentUser?.email)
    override fun hasUser(): Boolean {
        return Firebase.auth.currentUser != null
    }
}

```

```

        override suspend fun signIn(email: String, password:
String) {
            Firebase.auth.signInWithEmailAndPassword(email,
password).await()
        }
        override suspend fun signUp(email: String, password:
String) {
            Firebase.auth.createUserWithEmailAndPassword(email,
password).await()
        }
        override fun signOut() {
            Firebase.auth.signOut()
            user = null
        }
    }
}

```

Чтобы совсем не запутаться в классах и их методах посмотрим на рисунок 14, на котором представлена диаграмма классов.

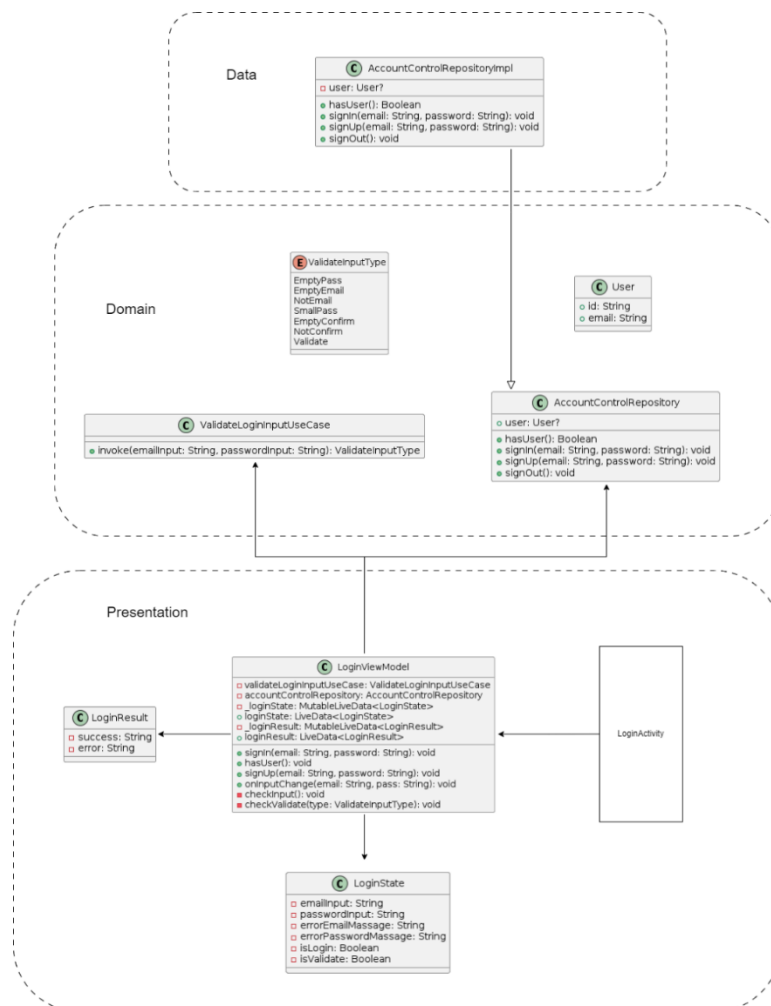


Рисунок 14 - Диаграмма классов для авторизации

Теперь благодаря такой реализации приложение можно легко тестировать и расширять приложение.

5 Тестирование

После того как получилось реализовать первый прототип приложения, следует заняться его тестированием. Этот этап является одним из важнейших этапов при разработке ПО, так как помогает выявить проблемы и ошибки в коде, а также улучшить пользовательский опыт взаимодействия.

В данном пункте будут протестированы такие программные моменты как:

- регистрация;
- авторизация;
- добавление, редактирование и удаление привычки;
- просмотр статистики;
- взаимодействие с пользователем.

5.1 Авторизация

В первую очередь стоит открыть клиентское приложение, где мы увидим окно авторизации для пользователей изображенное на рисунке 15. На данном окне представлены 2 редактируемых поля для ввода почты и пароля соответственно, не кликабельная кнопка «Войти», а также текстовое кликабельное поле для регистрации.

					ПГУ1.090301.21.001 ПЗ	Лист
						36
Изм.	Лист	№ докум.	Подпись	Дата		

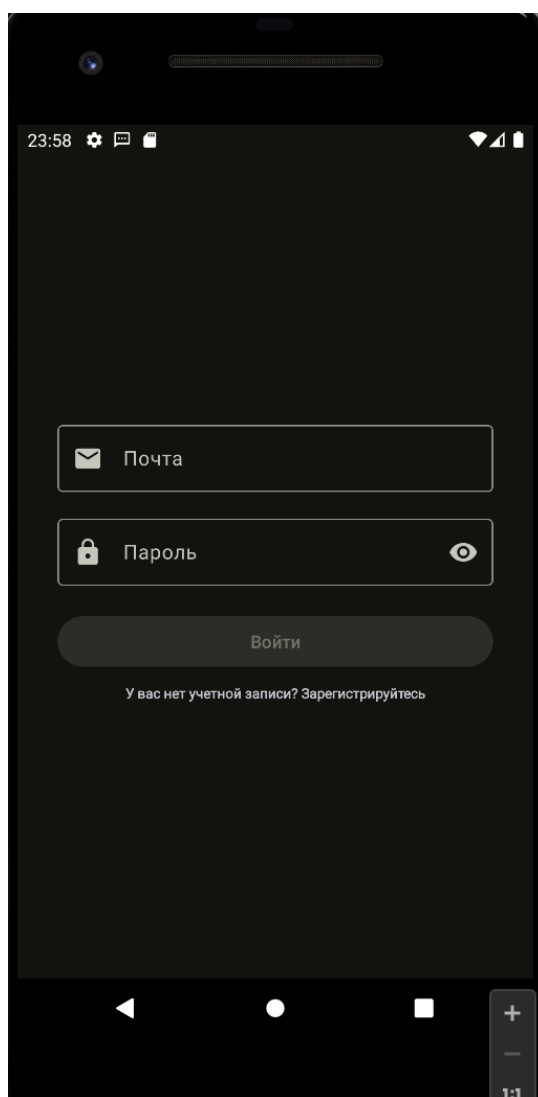


Рисунок 15 – Стартовое окно авторизации

После того как пользователь попытается ввести какие-то данные в одно из полей, появится сообщение об ошибке, а также визуальное выделение выбранного поля. Сообщение будет содержать информацию о правильности введенных данных. Например, на рисунке 16 пользователь ввел неправильный формат почты.

					ПГУ1.090301.21.001 ПЗ	Лист
						37
Изм.	Лист	№ докум.	Подпись	Дата		

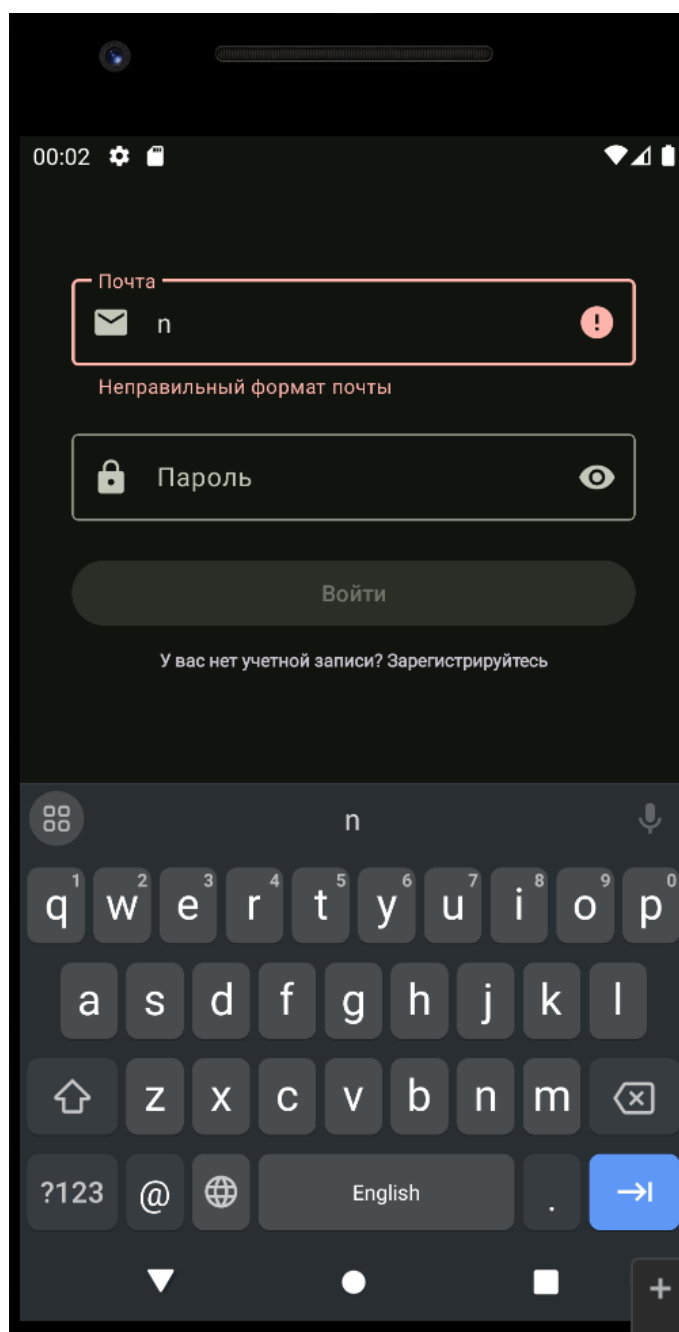


Рисунок 16 – Сообщение об ошибке при вводе данных

Также приложение может сообщить информацию о том, что поле пустое или содержит слишком маленький пароль как представлено на рисунках 17 и 18 соответственно.

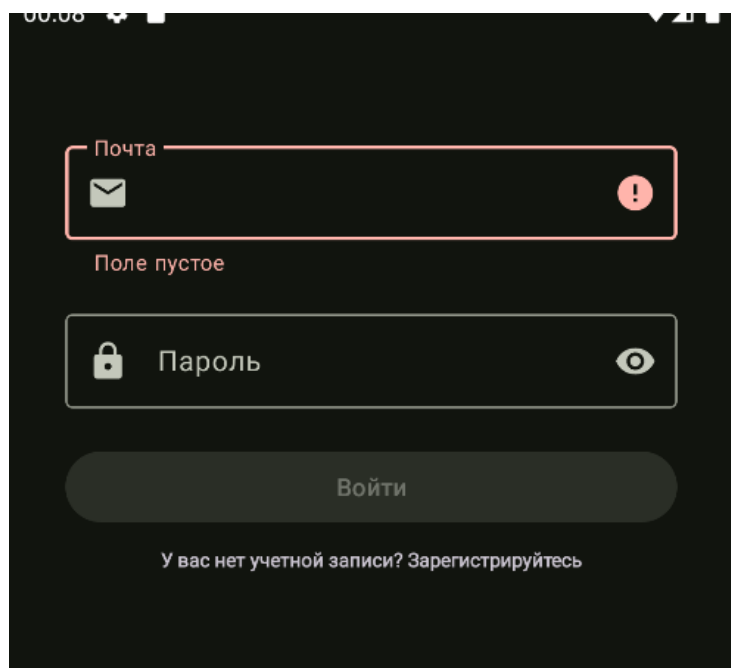


Рисунок 17 – Ошибка. Поле пустое

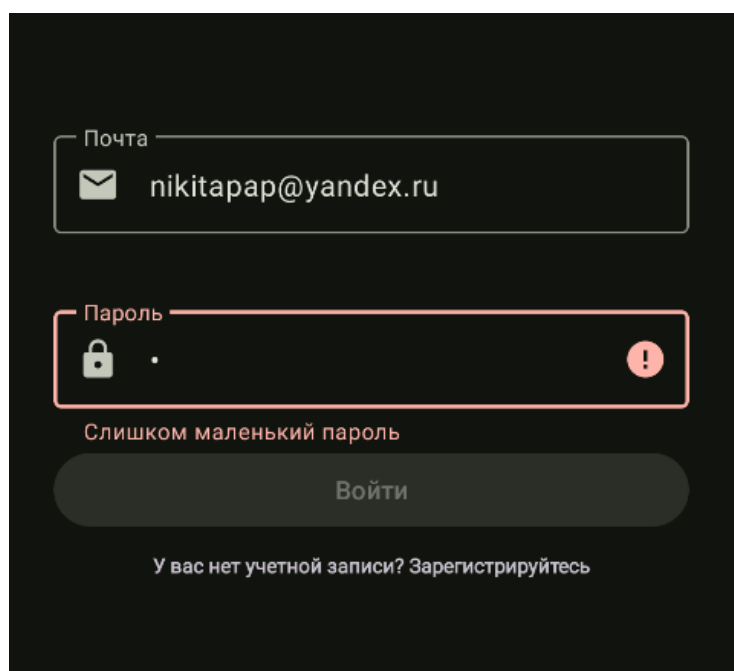


Рисунок 18 – Ошибка. Слишком маленький пароль

Если же пользователь введет достоверные данные, кнопка «Войти» станет доступной, после чего пользователь сможет войти в аккаунт, рисунок 19.

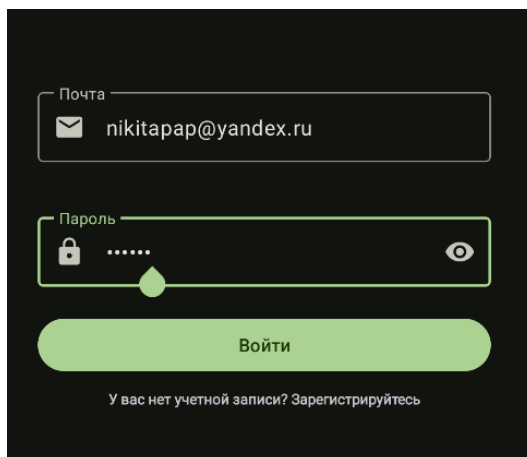


Рисунок 19 - Достоверность данных

После успешной авторизации пользователь попадает на главное окно приложения как представлено на рисунке 20.

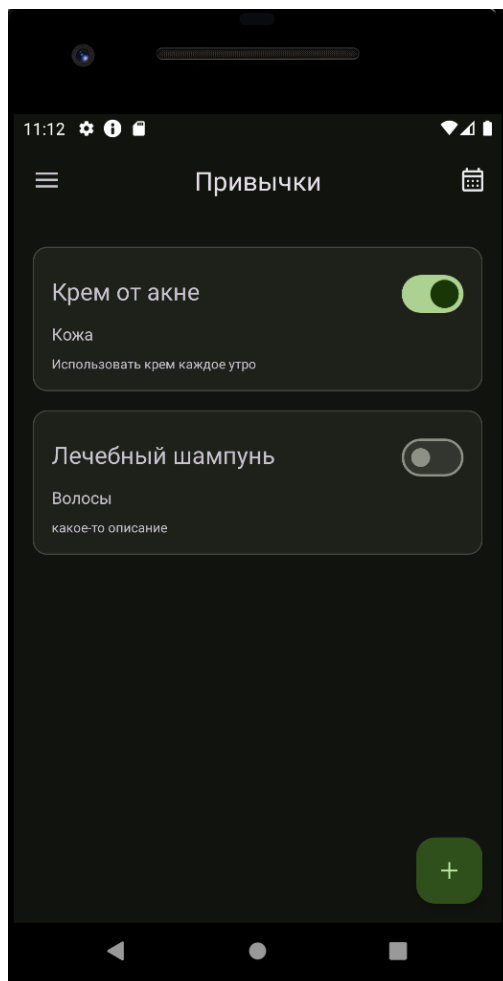
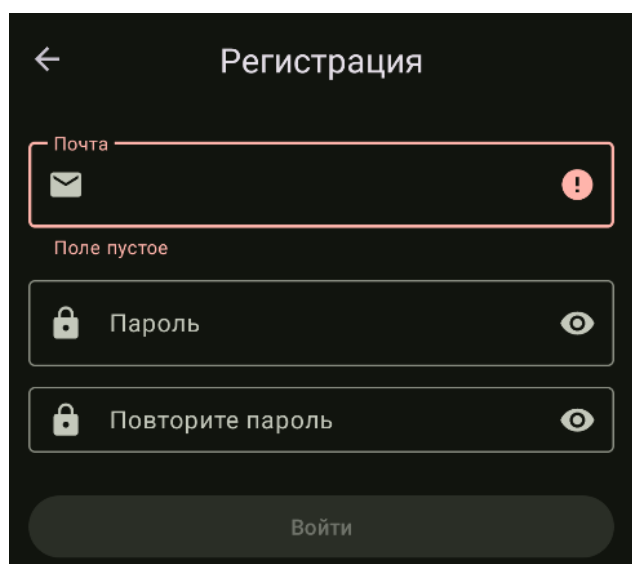


Рисунок 20 - Главное окно приложения

5.2 Регистрация

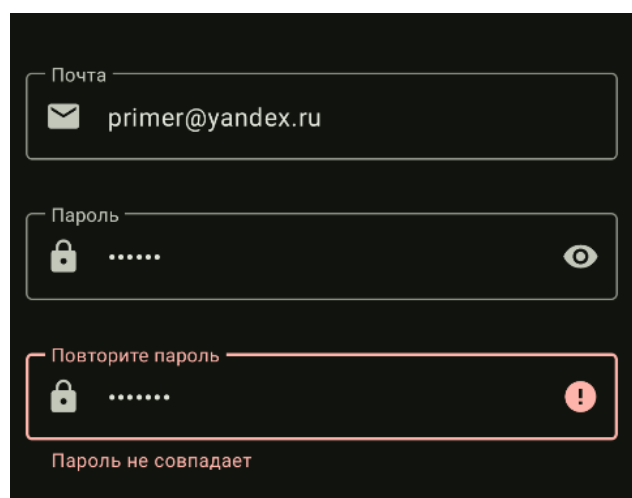
Если у пользователя не существовало аккаунта, то он может зарегистрироваться, нажав на текстовое поле ниже главной кнопки. После этого его перекинет на другую активность, в которой попросят заполнить нужные поля, рисунок 21.



The screenshot shows a registration form titled "Регистрация" with a back arrow. It contains three input fields: "Почта" (Email) with an envelope icon and a red exclamation mark icon; "Пароль" (Password) with a lock icon and an eye icon; and "Повторите пароль" (Repeat password) with a lock icon and an eye icon. Below the email field, the text "Поле пустое" (Field is empty) is displayed. At the bottom is a "Войти" (Login) button.

Рисунок 21 – Окно регистрации

В данном окне у последнего поля ввода появилась новая проверка на то чтобы пароль совпадал, пример проявления данной ошибки показан на рисунке 22.



The screenshot shows the same registration form as Figure 21, but with the "Почта" field filled with "primer@yandex.ru" and the "Пароль" field filled with ".....". The "Повторите пароль" field is also filled with "....." and has a red exclamation mark icon. Below this field, the error message "Пароль не совпадает" (Password does not match) is displayed.

Рисунок 22 - Ошибка. Пароль не совпадает

Как и в предыдущем окне с авторизацией все поля должны быть строго заполнены иначе кнопка «Войти» не будет доступна. Чтобы проверить корректность создания нового аккаунта проанализируем изменение в базе данных аккаунтов. Так на рисунке 23 до регистрации представлено всего 3 аккаунта.

Identifier	Providers	Created ↓	Signed In	User UID
nikita@yandex.ru	✉	May 16, 2024	May 16, 2024	uvvQ8h7GzJQBmAfYQDnhszz...
iska@gmail.com	✉	May 11, 2024	May 15, 2024	GBNG9iJwlFe75y4ZJMRWS9C...
nikitapap@yandex.ru	✉	May 9, 2024	May 17, 2024	m4fMOYaSlSSkRh9ZyQrdlcbV...

Рисунок 23 - Количество аккаунтов до регистрации

Но как только пользователь нажмет на кнопку «Войти», то на сервере сразу же создастся новый аккаунт (рисунок 22), а сам пользователь перейдет на экран главного окна, рисунок 25.

<div>Search by email address, phone number, or user UID</div> <div>Add user ↻ ⋮</div>				
Identifier	Providers	Created ↓	Signed In	User UID
nikita@yandex.ru	✉	May 16, 2024	May 16, 2024	uvvQ8h7GzJQBmAfYQDnhszz...
primer@yandex.ru	✉	May 15, 2024	May 15, 2024	MjERhai26qNb5XVclc05h5fTh...
iska@gmail.com	✉	May 11, 2024	May 15, 2024	GBNG9iJwlFe75y4ZJMRWS9C...
nikitapap@yandex.ru	✉	May 9, 2024	May 17, 2024	m4fMOYaSlSSkRh9ZyQrdlcbV...
Rows per page: 50 1 – 4 of 4 < >				

Рисунок 24 – Количество аккаунтов после регистрации

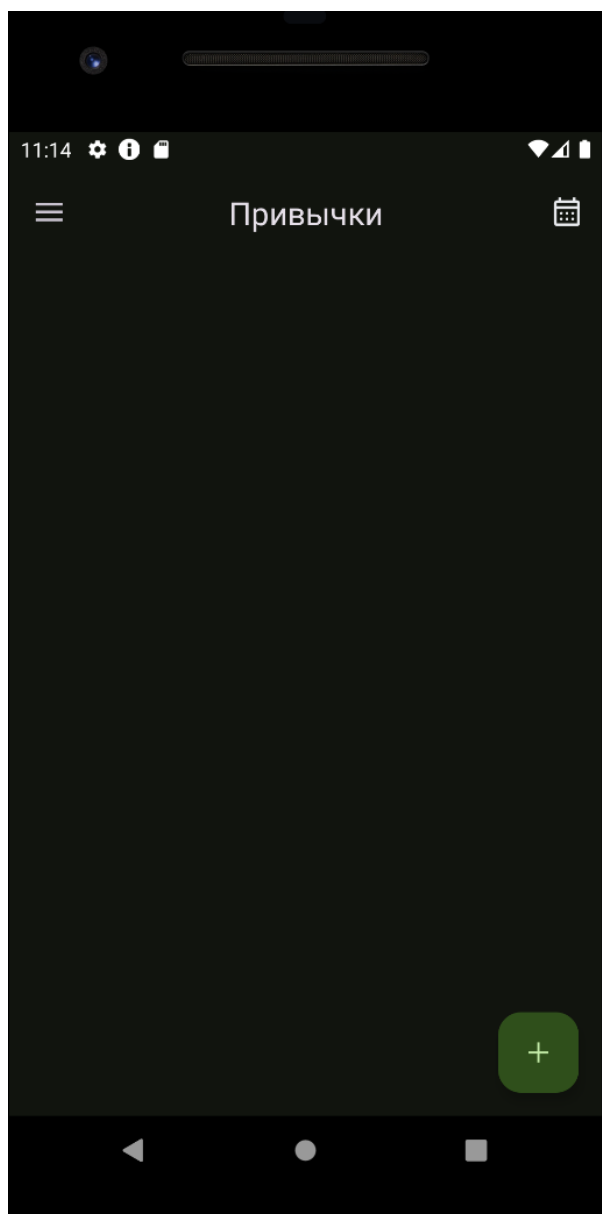


Рисунок 25 – Главное окно для нового пользователя

5.3 Взаимодействие с привычками

Чтобы добавить новую привычку следует нажать на кнопку в правом нижнем углу, после чего пользователя перекинёт на экран создания привычки, рисунок 26.

					ПГУ1.090301.21.001 ПЗ	Лист
						43
Изм.	Лист	№ докум.	Подпись	Дата		

Рисунок 26 – Экран создания привычки

На данном экране присутствует несколько редактируемых полей, кнопка «Добавить», а также кнопка вернуться назад в верхнем левом углу.

Пользователь может ввести название привычки, выбрать ее тип как на рисунке 27, выбрать время что представлено на рисунке 28 и добавить описание, рисунок 29.

Рисунок 27 Выбор типа привычки

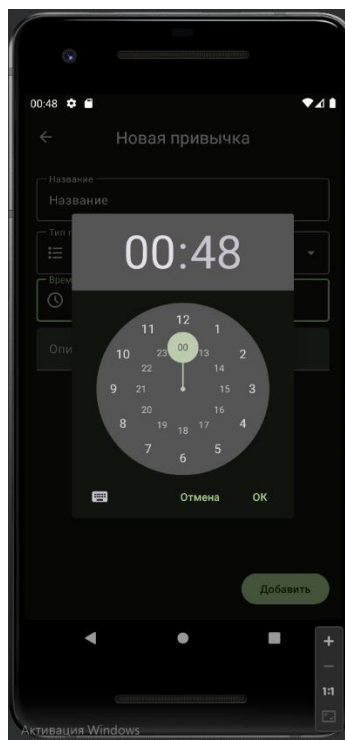


Рисунок 28 - Установка времени напоминания

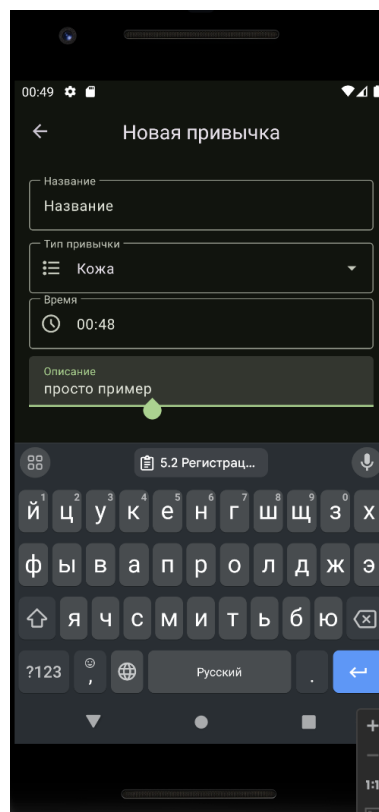


Рисунок 29 - Добавление описания

					ПГУ1.090301.21.001 ПЗ	Лист
						45
Изм.	Лист	№ докум.	Подпись	Дата		

После того как пользователь нажмет на кнопку «Добавить», на главном экране появится новая привычка рисунок 30.

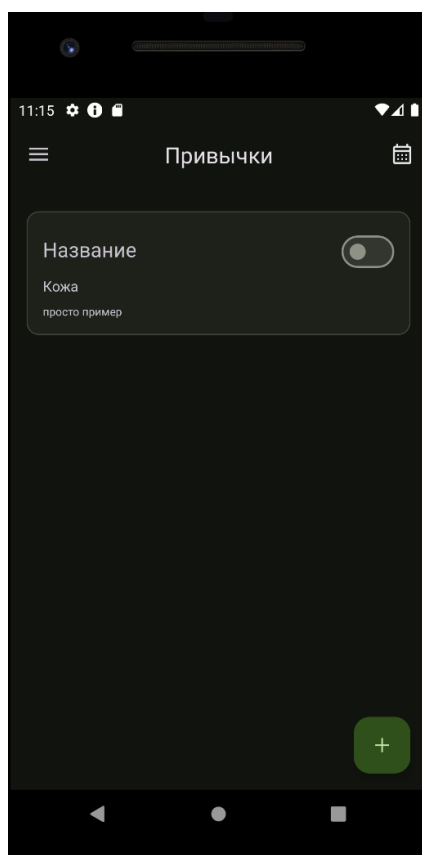


Рисунок 30 - Добавлена новая привычка

Если наступит выбранное время, то на телефон придет уведомление, которое напомнит пользователю о выполнении процедур. Данное уведомление изображено на рисунке 31.

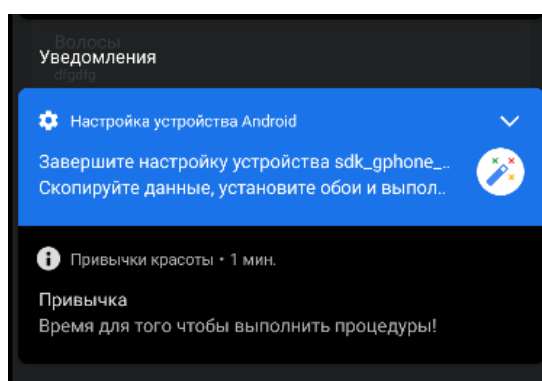


Рисунок 31 - Уведомление

Чтобы проверить добавление привычки на сервер зайдём в Firestore Database и заметим, что наши данные сохранились на сервере, рисунок 32.

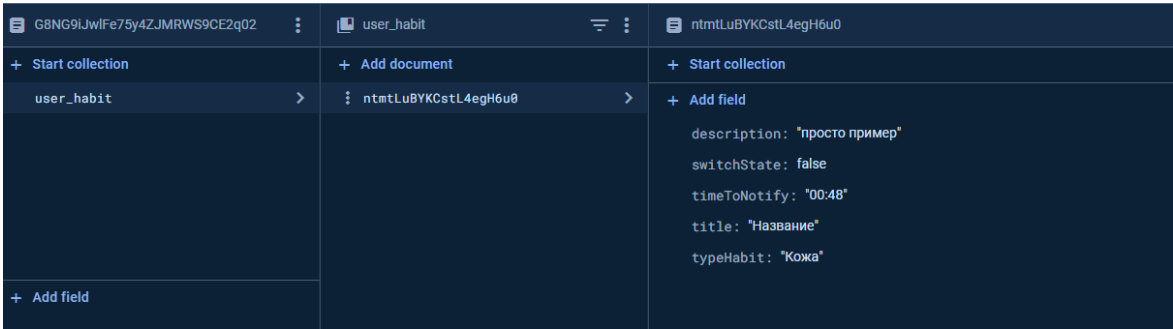


Рисунок 32 - Сохраненные данные

Для того чтобы отредактировать или удалить уже имеющуюся привычку требуется ненадолго зажать пальцем карточку с этой привычкой, после чего откроется экран редактирования, что можно увидеть на рисунке 33.

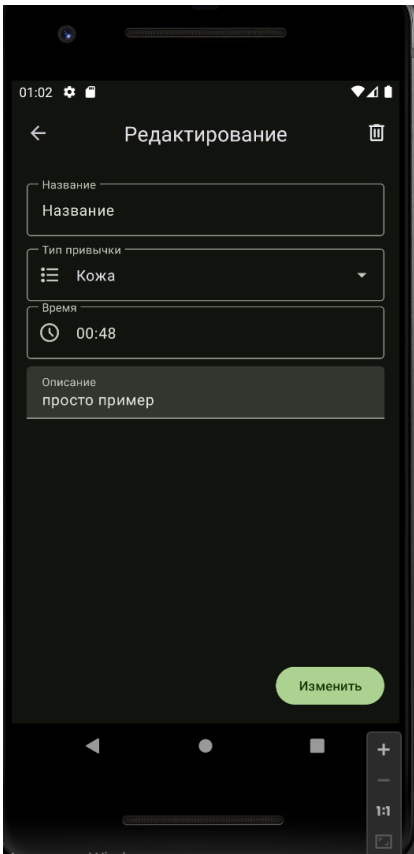


Рисунок 33 - Экран редактирования

Попробуем поменять одно из полей и сохраним результат, рисунок 34.

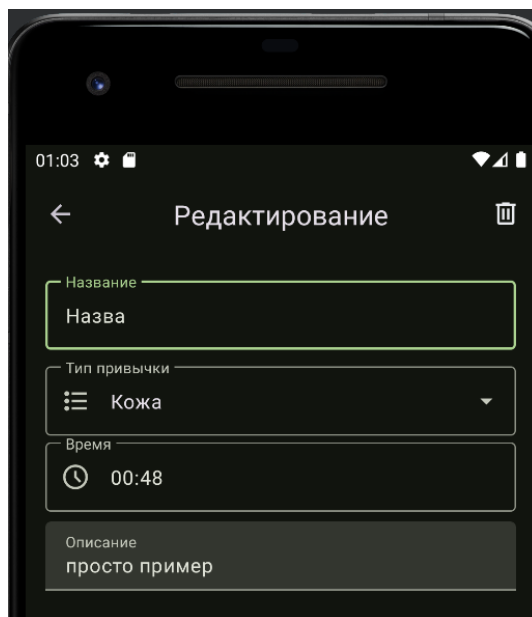


Рисунок 34 - Изменение одного из полей

После этого на рисунке 35, где изображен главный экран, можно заметить уже измененную привычку.

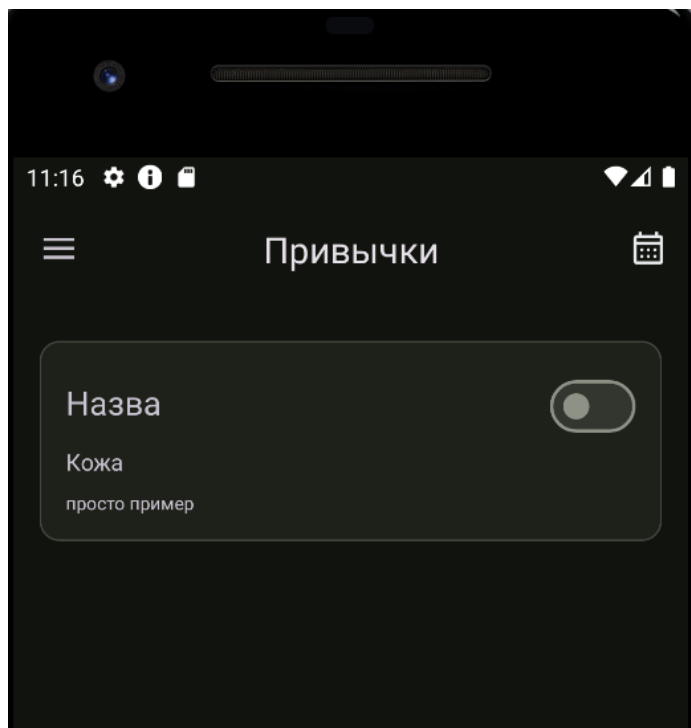


Рисунок 35 - Список с измененной привычкой

Также проверим изменилась ли информация на сервере, на рисунке 36 видно, что название поменялось, так что все отработало правильно.

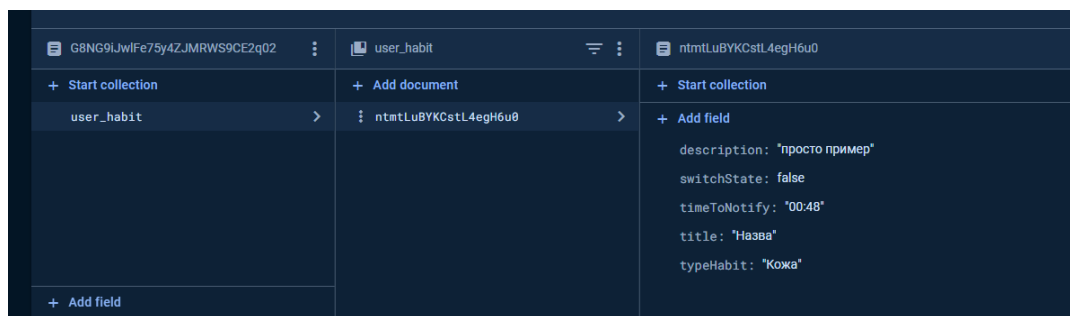


Рисунок 36 – Изменения на сервере

Последнее что стоит проверить в данном пункте это удаление привычки. Чтобы удалить привычку пользователь снова должен нажать нужную ему карточку и выбрать иконку корзины, как показано на рисунке 37.

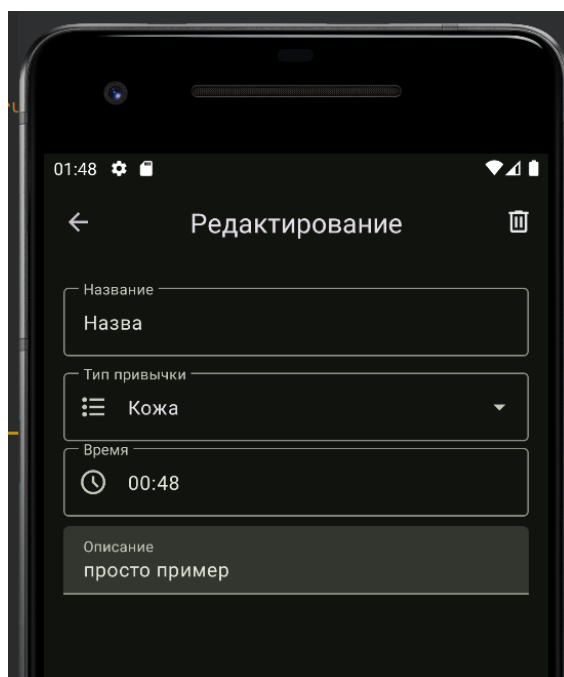


Рисунок 37 - Иконка корзины

После удаления на рисунке 38 видно, что привычки больше нет

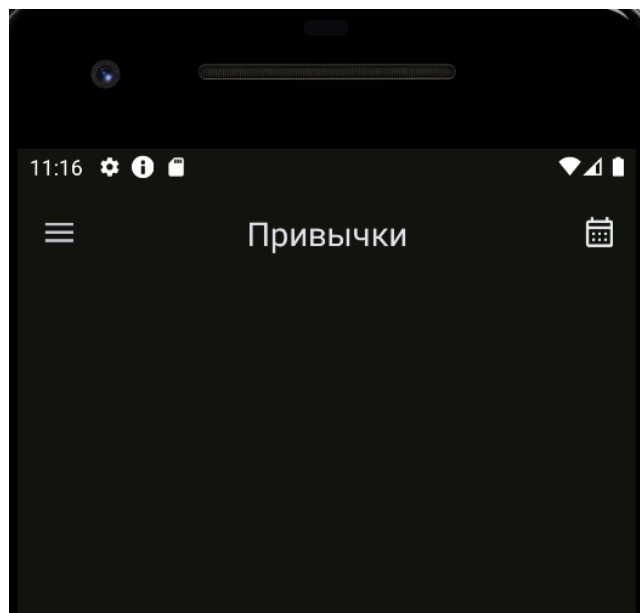


Рисунок 38 - Список привычек после удаления привычки

Последняя проверка заключается в просмотре файлов сервера. Как видно на рисунке 39, на сервере нет ни одной привычки, точно также, как и у пользователя в приложении.

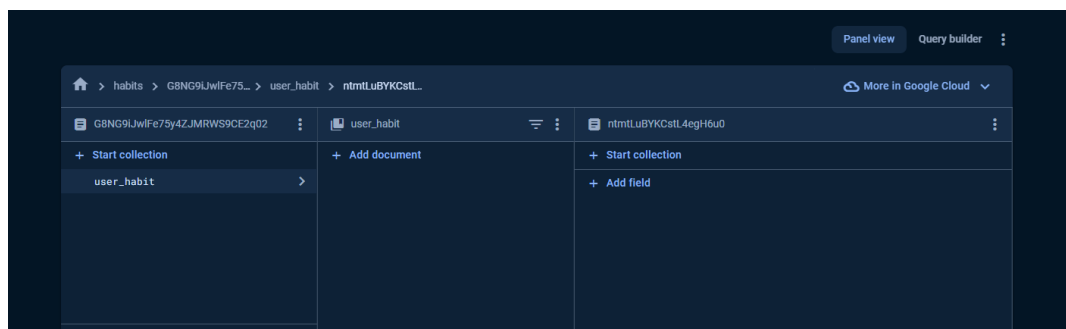


Рисунок 39 - Отсутствие привычек на сервере

5.4 Просмотр статистики

Если на главном экране пользователь нажмет на значок календаря в правом верхнем углу, то в диалоговом окне откроется календарь, рисунок 40.

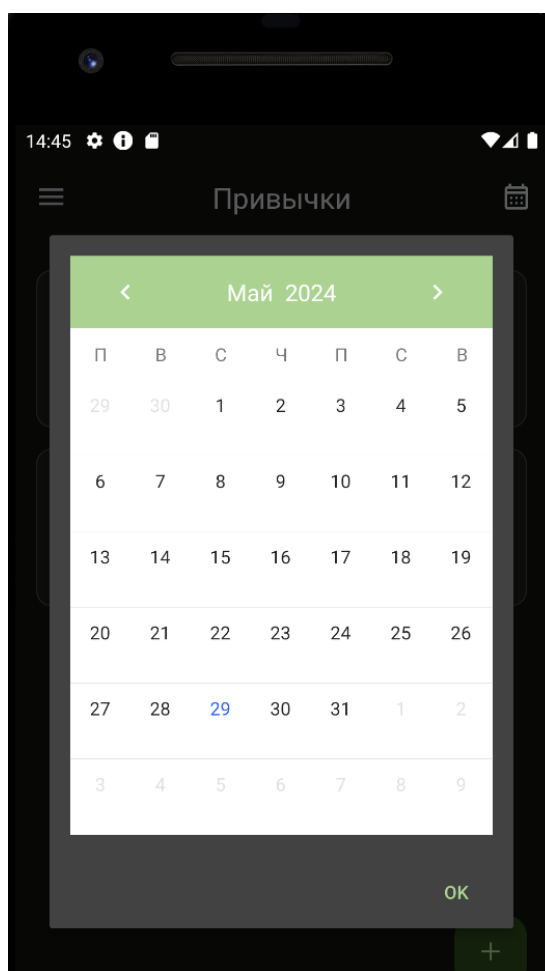


Рисунок 40 - Календарь в диалоговом окне

Если пользователь нажмет на тумблер одной из привычек как показано на рисунке 41, то в календаре появится отметка о том, что сегодня были выполнены действия, рисунок 42.

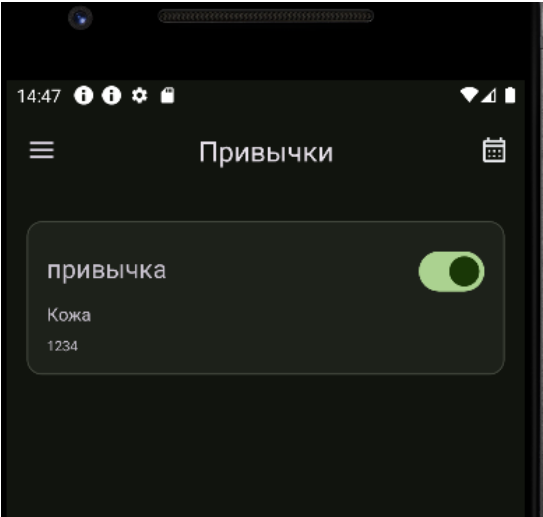


Рисунок 41 - Отметка привычки

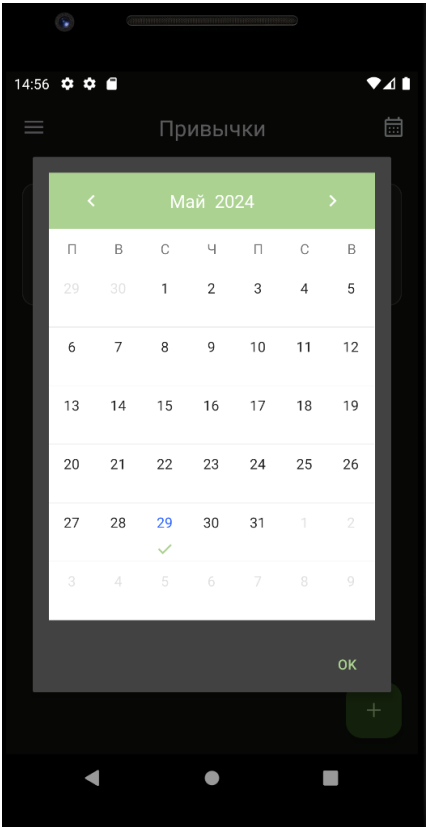


Рисунок 42 - Отметка на календаре

5.5 Взаимодействие с пользователем

В данном пункте будут рассмотрены такие взаимодействия между авторизовавшимся пользователем и приложением как: выход из аккаунта и отправка данных на сервер о себе.

Чтобы открыть меню взаимодействия стоит выдвинуть окно навигации на главной странице. Это делается либо с помощью нажатия иконки в верхнем правом углу, либо с помощью вытягивания меню из левого края телефона, рисунок 43.

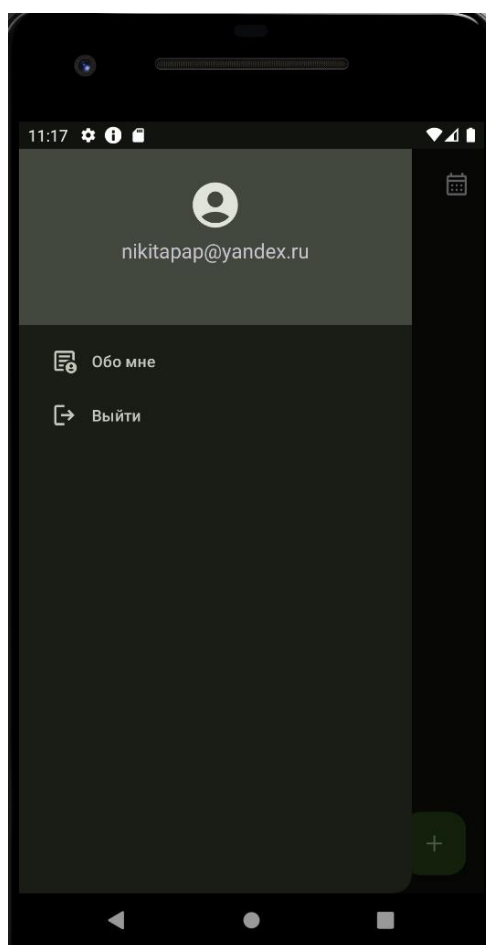


Рисунок 43 - Окно навигации приложения

После того как пользователь нажмет на пункт меню «Обо мне» он перейдет на окно для заполнения формы, рисунок 44.

Рисунок 44 - Окно для заполнения анкеты

После отправки данных пользователь переходит на главный экран, а на сервере появляется дополнительная коллекция «user_about» в которой будут храниться анкеты пользователя. Также сама анкета сохраняется с временной меткой, что представлено на рисунке 45.

+ Start collection	+ Add document	+ Start collection
user_about >	8NghIpqjrgdZhyw3L1EY >	+ Add field
user_habit		age: 22
		desc: "применяю лекарство от прыщей, а также лосьон для волос"
		height: 178
		timestamp: May 29, 2024 at 2:28:27 PM UTC+3
		weight: 59
+ Add field		

Рисунок 45 - Анкета пользователя на сервере

Если же пользователь нажмет на пункт меню «Выйти» представленный на рисунке 43, то он сразу же окажется на стартовом экране с авторизацией.

5.6 Вывод

В момент создания приложения было выявлено достаточное количество ошибок. Но протестировав ПО получилось исправить и обработать большую часть из них, это позволило сделать более надежное и отказоустойчивое приложение.

					ПГУ1.090301.21.001 ПЗ	Лист
						55
Изм.	Лист	№ докум.	Подпись	Дата		

6 Безопасность жизнедеятельности

Чтобы максимально обезопасить жизни разработчиков и обычных людей, которые проводят за компьютером достаточно много времени, были разработаны специальные правила и нормативы по обеспечению безопасности за ПК и мобильными устройствами. Пренебрежение данными правилами может влечь за собой негативные последствия как для здоровья пользователя, так и для его имущества.

В данном пункте будут рассмотрены основные правила и инструкции по работе человека с компьютером и мобильным телефоном.

6.1 Правила и техника безопасности при работе с ПК.

Человек при взаимодействии с вычислительной техникой может подвергаться таким вредным и опасным воздействиям как:

- поражение электрическим током;
- электромагнитное излучение;
- опасность пожара;
- повышенный уровень шума;
- механические повреждения.

Для того чтобы снизить данное влияние требуется соблюдать ряд определенных правил.

В первую очередь нужно обеспечить рабочее помещение достаточным уровнем освещения, которое не будет оставлять бликов на мониторе и светить прямо в глаза рабочему. Нужно учитывать, что освещение должно быть, как естественным, так и искусственным, но не слишком ярким.

Также требуется обзавестись удобной и эргономичной офисной мебелью. Стул должен регулироваться по высоте и иметь регулировку спинки, чтобы ноги могли быть выпрямлены вперед и ступни полностью стояли на полу, а также для того чтобы работник мог опереться в спинку стула. Разработчик должен сидеть от

монитора на уровне вытянутой руки, а монитор должен быть наклонен примерно на 30 градусов. Все эти правила по безопасности позволят обеспечить правильную осанку, а также снизить нагрузку на позвоночник, руки и голову.

В целях безопасности не следует пользоваться компьютером или любой техникой при наличии внешних повреждений. Необходимо следить на своем рабочем месте за исправным состоянием техники и не оставлять ее включенной на долго без присмотра людей. Рабочие места должны быть установлены вдоль стен так, чтобы монитор задней стенкой был обращен к стене. Это поможет снизить количество вредных электромагнитных излучений [8].

Чтобы снизить как физическую, так и психологическую нагрузку требуется периодически отдыхать от работы. При 8 часовой смене общая продолжительность перерывов должна составлять от 30 до 90 минут. Также для профилактики заболевания глаз и опорно-двигательного аппарата работнику следует проводить короткие упражнения, нацеленные на снятия нагрузок с собственного тела.

6.2 Правила и техника безопасности при работе со смартфоном.

Так как основным устройством, где будет работать приложение, является телефон, то следует также проанализировать и учесть все правила безопасности по обращению с ним.

Не следует пытаться вскрывать, разбирать или модифицировать устройство самостоятельно. Также не следует подвергать устройство физическим нагрузкам (ронять, сгибать, бросать, протыкать) или воздействию огня или воды.

Если мобильное или зарядное устройство неисправно использовать его строго запрещено. Также следует пользоваться только оригинальными зарядными устройствами и аксессуарами.

Мобильное устройство постоянно выделяет радиоизлучение, поэтому в целях безопасности не стоит класть телефон близко к жизненно важным органам, а также при разговоре периодически менять положение смартфона относительно головы. Также нужно учитывать, что данных и качественных исследований относительно пагубного влияния радиоизлучения пока что нет [9].

Последнее на что следует обратить внимание это на чрезмерное злоупотребление использованием телефона может привести к зависимости, а также нарушить сон и концентрацию.

					ПГУ1.090301.21.001 ПЗ	Лист
						58
Изм.	Лист	№ докум.	Подпись	Дата		

7 Стартап проект

Инициатор проекта планирует разработку мобильного приложения для устройств на ОС Android.

Разместить бизнес планируется в г.Пенза в помещении, соответствующему всем нормам и требованиям РК.

Данный проект призван упростить рутинные задачи по выполнению однотипных ежедневных действий в целях выработки полезных привычек. Также целью деятельности участников проекта является извлечение дохода для улучшения материального благосостояния его участников.

Целевой аудиторией данного проекта будут являться малые и средние бизнесы связанный со здоровьем, к примеру фитнес-центры, косметические компании, а также компании, которые специализируются на предоставлении медицинских услуг (оптика, фармацевтика).

7.1 Сегменты ЦА

Целевой аудиторией будут являться разные по своей сути компании, поэтому чтобы более эффективно взаимодействовать с каждой из них, следует учитывать их потребности и особенности. Можно рассмотреть каждый бизнес в отдельности чтобы четче понимать их предпочтения и предложить индивидуальное коммерческое предложение.

Фитнес центры и спортзалы нацелены на то чтобы клиенты как можно чаще посещали занятия и тренировки в рамках абонимента. Мобильное приложение поможет решить вопрос с мотивацией клиентов добавляя дополнительный стимул для посещения спортзала. Также при индивидуальном рассмотрении можно было бы предложить доработку продукта, например, добавление классического трекера, который будет отслеживать количество шагов и сжигаемых каллорий.

Для косметических компаний также важна мотивация клиентов. Чем мотивированнее клиент, тем потенциально больше прибыли он сможет принести

компании. Также стоит учитывать, что и сама компания может анализировать потребности клиентов и их привычки для того чтобы улучшить качество продукции. При индивидуальном рассмотрении такого рода компаниям можно было бы предложить внедрение дополнительного анкетирования и разделения привычек на типы.

Компании, специализирующиеся на предоставлении медицинских услуг, также могут быть заинтересованы в данном приложении. Пользователям было бы удобно отслеживать прием лекарственных средств, а компаниям в индивидуальном плане можно было бы предложить добавить функционал записи на прием к врачу.

Таким образом для каждого сегмента ЦА будут предложены индивидуальные предложения, которые будут учитывать их потребности. Стоимость разработки мобильного приложения будет меняться в зависимости от объема и специфики требований каждого заказчика.

Перед запуском проекта важно найти ключевых партнеров для ведения бизнеса. Партнерами могут являться как поставщики технического оборудования, так и непосредственно компании.

Поставщиками оборудования могут стать компании, которые предоставляют аренду компьютеров, серверов и другого оборудования, которое потребуется для нормального функционирования мобильного приложения.

Малый и средний бизнес станет основным клиентом данного продукта. Сотрудничество с ними позволит предоставить им удобное и эффективное решение для повышения мотивации и выработки полезных привычек у пользователей. Также данная взаимовыгодная сделка поможет им сэкономить время, финансы и ресурсы.

7.2 Ключевые процессы

Ниже перечислены основные ключевые процессы при разработке стартап-проекта:

					ПГУ1.090301.21.001 ПЗ	Лист
						60
Изм.	Лист	№ докум.	Подпись	Дата		

- поиск и создание команды;
- поиск средств;
- разработка ПО;
- поиск каналов сбыта и взаимодействие с целью продажи;
- поддержка проданного продукта.

Для разработки такой системы нужно правильно подобрать команду, на начальном этапе в нее должны входить: разработчики, бухгалтер, менеджер по продажам. В нашем случае разработчики найдены из числа единомышленников. А бухгалтер и менеджер по продажам были найдены на сайте с резюме.

Этапы разработки ПО описаны в пункте 4.

Поиск каналов сбыта осуществляется с помощью коммерческих предложений. Коммерческое предложение имеет несколько значимых достоинств, которые помогут привлечь покупателей и отличить этот проект от конкурентов.

Во-первых, большинство приложений, которые представлены на рынке не имеют русской локализации.

Во-вторых, предлагается индивидуальный подход к каждой компании, что позволяет создать более удобное мобильное приложение для конкретных пользователей.

В-третьих, высокая популярность здорового образа жизни, может привлечь внимание огромной аудитории к данному проекту, что повлечет за собой высокий спрос.

Можно отметить, что использование передовых и популярных технологий, таких как мобильные приложения, может повысить имидж компании и показать ее инновационный подход к бизнесу. Клиенты могут воспринимать компанию как современную, внимательную к новым тенденциям и готовую предложить им удобные и современные способы взаимодействия.

Ниже в таблице 2 приведены услуги, предложенные в коммерческом предложении.

Таблица 2 – Наименование предложенных услуг

Наименование	Цена реализации
Разработка под компанию	50 000р
Поддержка	10 000р/мес
Доработка продукта	20 000р

7.3 Каналы взаимодействия

Для более эффективной демонстрации функционала и ценности мобильного приложения нужно, путем обзвона, согласовать личные встречи с руководителями малых и средних бизнесов. На таких встречах можно показать полный спектр возможностей продукта и ответить на вопросы потенциальных клиентов.

После заключения сделки, наша команда будет отвечать за поставку и внедрение мобильного приложения. Это включает установку и настройку приложения в соответствии с потребностями и предпочтениями клиента. Четкое планирование и организация этапов внедрения помогут сэкономить время и обеспечить плавный переход к полноценному использованию системы. Также нужно обеспечить поддержку клиентов в течение нескольких месяцев после покупки продукта. Это включает в себя решение технических проблем, обновление и доработку системы в соответствии с потребностями клиента, а также консультации и обучение пользователей по использованию приложением. Быстрый и качественный ответ на запросы клиентов поможет создать положительное впечатление от сотрудничества. Чтобы поддерживать долгосрочные отношения с клиентами, нужно стремиться регулярно предоставлять обновления и новости о продукте через электронные рассылки.

7.4 Источники дохода

Продукт будет иметь следующие источники доходов:

– продажа продукта. Доход будет получаться за продажу и внедрение мобильного приложения для каждой компании. Стоимость одной сделки приведена в таблице 2;

– поддержка. После внедрения приложения требуется предоставлять услугу поддержки заказчикам в течении нескольких месяцев. Плата за данный источник дохода будет составлять 10 000 рублей ежемесячно;

– доработка. Компания может попросить модифицировать, улучшить или доработать данное мобильное приложение. Стоимость данной услуги представлена в таблице 2.

Для более точного подсчета количества прибыли будем рассчитывать доход для конкретной сферы бизнеса, например, косметических компаний. Так согласно источнику 2ГИС в Пензе их насчитывается порядка 249. Исходя из предположения, что около 20% из них готовы приобрести мобильное приложение, можно оценить, что 50 компаний станут будущими клиентами. Также стоит учитывать, что 50% захотят уникальный продукт, что повысит доход за счет доработки приложения.

Доход высчитывается по формуле

$$Д = К * Ц , \quad (1)$$

где Д – доход;

К – количество потребителей;

Ц – установленная цена.

Доход от разработки и внедрения будет определяться

$$50\,000\text{р} * 50\text{пр} = 2\,500\,000\text{р}$$

Доход от поддержки будет высчитываться

$$10\,000\text{р} * 50\text{пр} = 500\,000\text{р/мес}$$

Доход от модификации приложения высчитывается

$$20\,000\text{р} * 25\text{пр} = 500\,000\text{р}$$

Таким образом, общий ожидаемый доход в г. Пенза составит 3 000 000 рублей единоразово и 500 000р ежемесячно. Для наглядности в таблице 3 представлена прогнозируемая программа производства по месяцам для коммерческих учреждений по г. Пенза.

Таблица 3 - Прогнозируемая программа производства

Наименование услуги	1 мес.	2 мес.	3 мес.	4 мес.	5 мес.
Разработка	5	12	14	10	9
Поддержка	5	17	31	35	40
Доработка продукта	2	9	16	16	19

7.5 Ключевые ресурсы

Для запуска, функционирования и развития стартап-проекта потребуются некоторые ключевые ресурсы. Такими ресурсами будут являться:

- разработчики ПО для мобильных устройств на базе ОС Android;
- менеджеры по продажам, обладающие навыками продажи и умением вести переговоры с потенциальными клиентами;
- техническое оборудование (компьютеры, серверы и программное обеспечение) для разработки и обслуживания приложений;
- финансовые ресурсы для финансирования разработки, маркетинг, продаж и операционных расходов бизнеса.

Финансирование проекта планируется осуществить как за счет собственных средств инициатора проекта, так и за счет государственной поддержки «Гранты для молодых предпринимателей». В таблице 4 представлена программа финансирования.

Таблица 4 - Источники финансирования

Источник финансирования	Сумма
Собственные средства	400 000р
Гос. поддержка	500 000р
Всего	900 000р

Приведенные выше ключевые ресурсы можно разделить на финансовые и людские. Подробно направление использования средств указано в таблице 5 и 6.

Таблица 5 - Финансовые ключевые ресурсы

Направление использования средств	Кол-во, шт	Цена, руб.	Сумма, руб
Аренда сервера (6 мес)	1	20 000	20 000
Аренда помещения для серверов (6 мес)	1	24 000	24 000
Закупка ПК	2	60 000	120 000
Комунальные услуги (6 мес)	6	5 000	30 000
Закупка другого оборудования	1	50 000	50 000

Таблица 6 - Людские ключевые ресурсы

Направление использования средств	Кол-во, шт	Цена, руб.	Сумма, руб
З/п разработчику (6 мес)	12	30 000	360 000
З/п бухгалтеру (6 мес)	6	20 000	120 000
З/п менеджеру (6 мес)	6	30 000	180 000

7.6 Риски проекта

Для того чтобы успешно реализовать стартап-проект требуется провести анализ всех рисков, которые могут негативно повлиять на его исход.

Возможными рисками являются, например, недостаточный спрос на предлагаемое мобильное приложение, проблемы с финансированием проекта, технические трудности при внедрении проекта и нестабильность экономической ситуации.

Важно проанализировать и предусмотреть меры по решению или минимизированию негативных последствий рисков.

					ПГУ1.090301.21.001 ПЗ	Лист
						66
Изм.	Лист	№ докум.	Подпись	Дата		

Заключение

Результатом выполнения данной выпускной квалификационной работы можно считать успешно разработанное мобильное приложение «Привычки красоты», полностью соответствующее заданию. Также были выполнены следующие задачи:

- полный анализ темы и аналогов;
- выбор инструментов разработки;
- анализ архитектуры разрабатываемого приложения;
- непосредственная разработка приложения;
- тестирование ПО;
- составление плана стартап-проекта.

В результате было создано полностью протестированное приложение для мобильных устройств написанное на языке программирования Kotlin. Также в ходе разработки были получены навыки работы с Firebase Cloud и Firebase Authentication. Таким образом результаты показали, что приложение полностью соответствует всем поставленным задачам, имеет удобный пользовательский интерфейс и выполняет свою работу корректно.

В перспективе приложение можно будет улучшить, добавив новый функционал и доработав пользовательский интерфейс для более комфортного использования со стороны пользователей. Данное мобильное приложение готово к использованию в бизнесе, связанном с уходовой косметикой, и поможет улучшить качество жизни пользователей за счет выработки полезных привычек.

Список используемых источников

1. gs.statcounter.com: [Электронный ресурс]. URL: <https://gs.statcounter.com/os-market-share/mobile/worldwide> (дата обращения: 22.04.2024)
2. ru.wikipedia.org: [Электронный ресурс]. URL: <https://ru.wikipedia.org/wiki/Java> (дата обращения: 25.04.2024)
3. kotlinlang.com: [Электронный ресурс]. URL: <https://kotlinlang.org/> (дата обращения: 25.04.2024)
4. developer.android.com: [Электронный ресурс]. URL: <https://developer.android.com/studio> (дата обращения: 30.04.2024)
5. firebase.google.com: [Электронный ресурс]. URL: <https://firebase.google.com/> (дата обращения: 03.05.2024)
6. blog.cleancoder.com: [Электронный ресурс]. URL: <https://blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html> (дата обращения: 6.05.2024)
7. developer.android.com: [Электронный ресурс]. URL: <https://developer.android.com/topic/architecture> (дата обращения: 8.05.2024)
8. oxrana-bez.ru: [Электронный ресурс]. URL: <https://oxrana-bez.ru/stati/tehnika-bezopasnosti-pri-rabote-s-personalnym-kompyuterom/> (дата обращения: 18.05.2024)
9. stupinoadm.ru: [Электронный ресурс]. URL: <https://stupinoadm.ru/novosti/detail/mery-bezopasnosti-pri-ispolzovanii-mobilnykh-ustroystv/> (дата обращения: 18.05.2024)

Приложение А

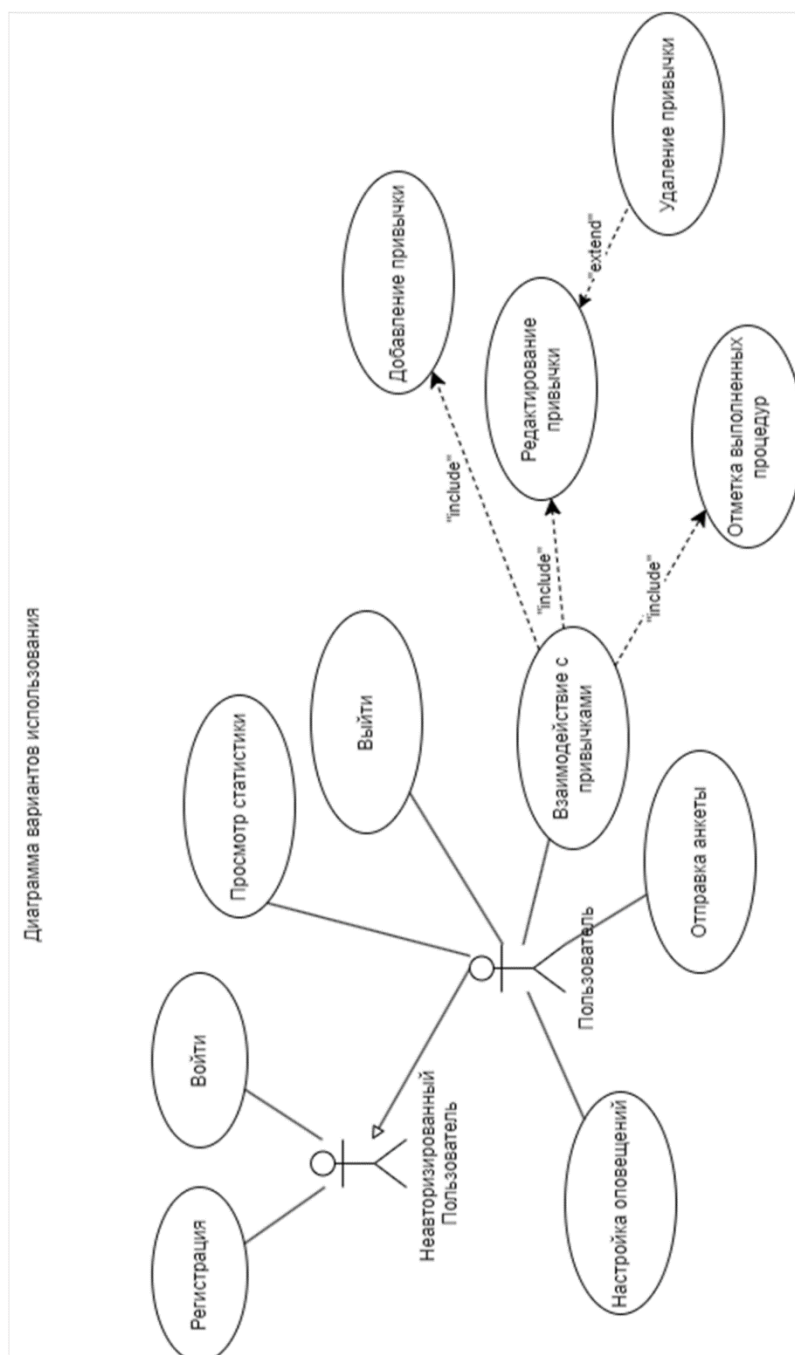
(Обязательное)

Стартап-проект. Приложение-трекер «Привычки красоты» для ОС Android

UML – диаграммы

					<i>ПГУ1.090301.21.001 ПЗ</i>	Лист
						69
Изм.	Лист	№ докум.	Подпись	Дата		

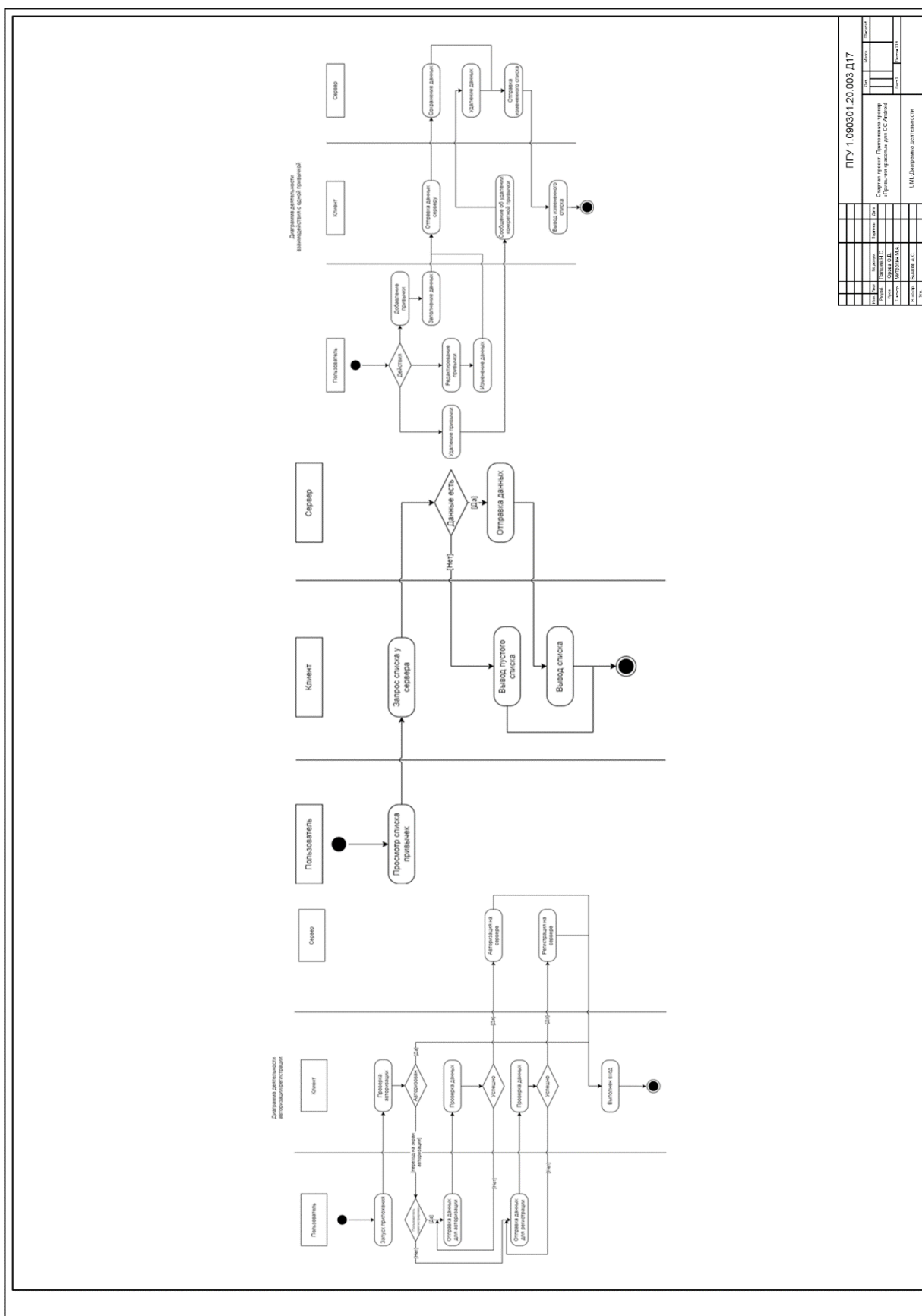
Приложение А.1-Диаграмма прецедентов

[illegible]

Изм.	Лист	№ докум.	Подпись	Дата

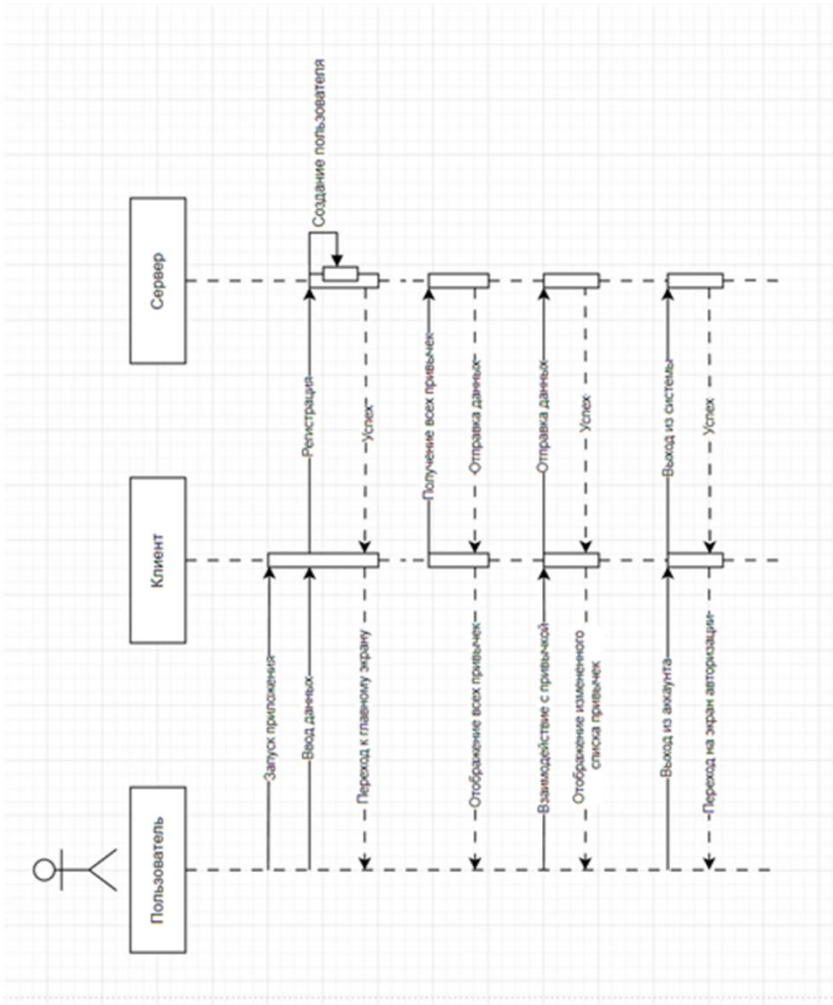
ПГУ1.090301.21.001 ПЗ

Приложение А.2-Диаграмма деятельности

[illegible]

Изм.	Лист	№ докум.	Подпись	Дата

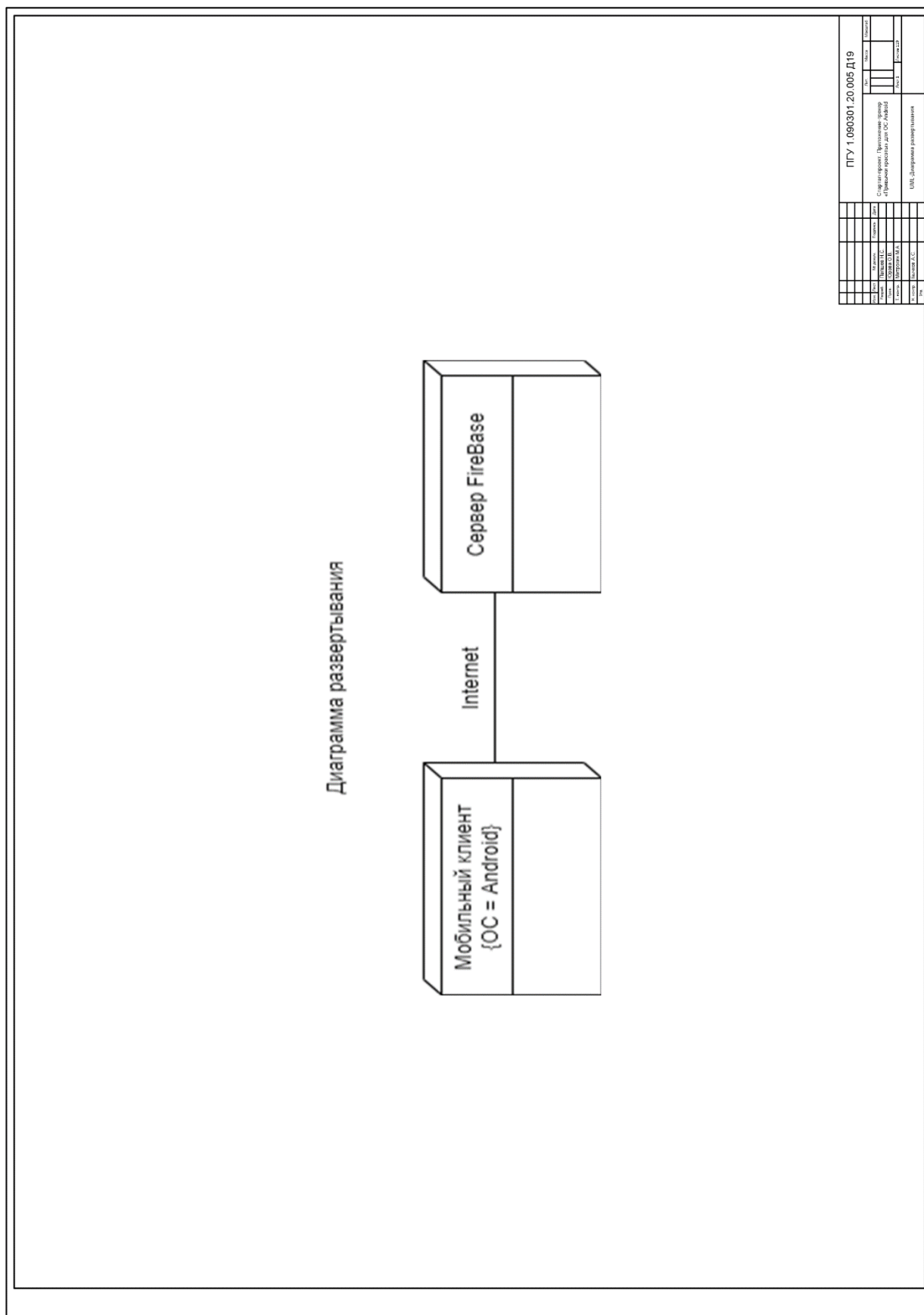
Приложение А.3-Диаграмма последовательностей



ПГУ 1.090301.20.004 Д14									
Имя	Фамилия	Пол	Дата	Время	Место	Страна	Город	Улица	Почта
Иван	Иванов	М	12.12.2020	14:30	Москва	Россия	Москва	ул. Ленина	ivan@yandex.ru
ИМ. Диаграмма последовательностей									

Изм.	Лист	№ докум.	Подпись	Дата

Приложение А.4-Диаграмма развертывания



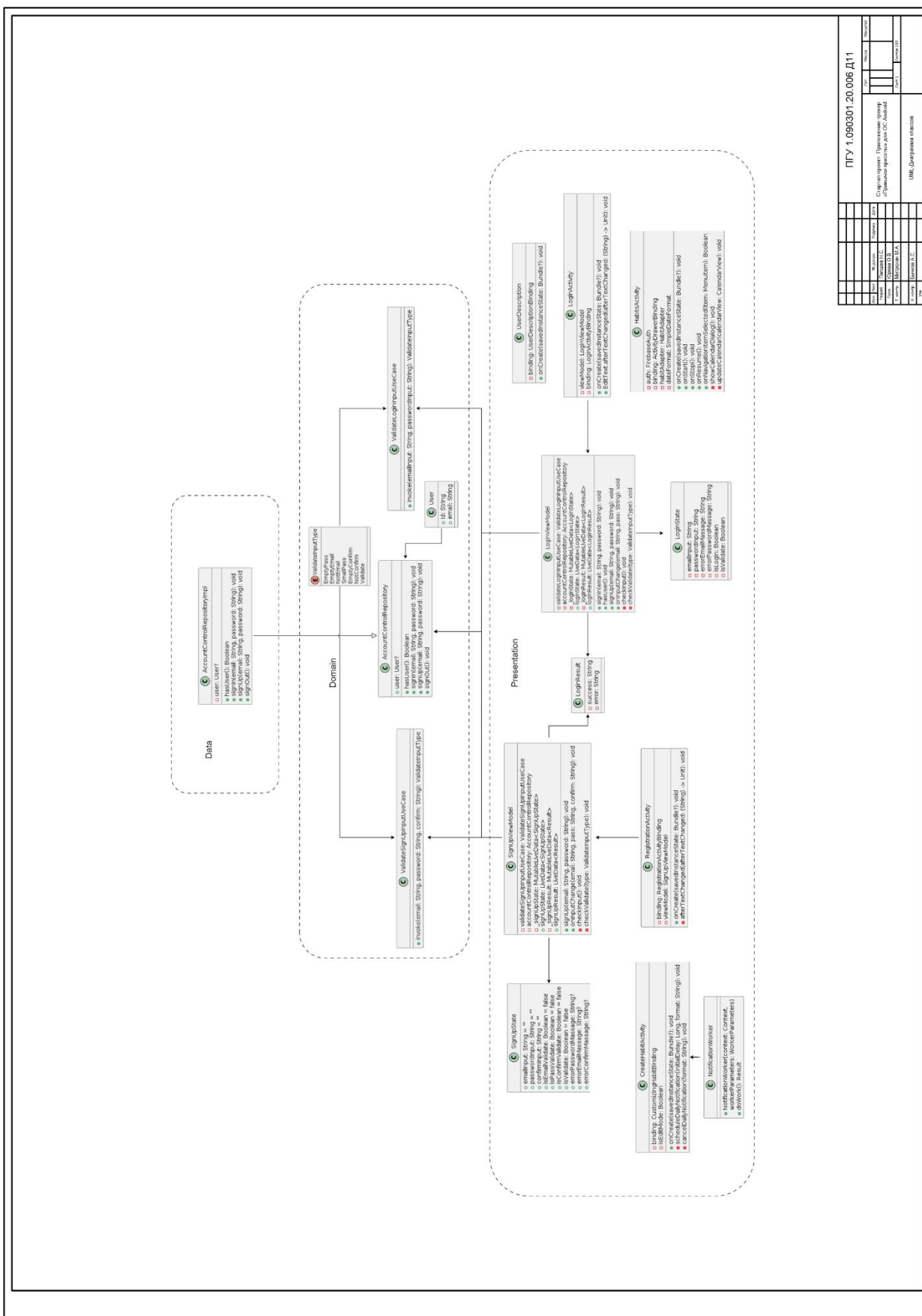
Изм.	Лист	№ докум.	Подпись	Дата

ПГУ1.090301.21.001 ПЗ

Лист

73

Приложение А.5-Диаграмма классов



Приложение Б

(Обязательно)

Стартап-проект. Приложение-трекер «Привычки красоты» для ОС Android

Листинг программы

					ПГУ1.090301.21.001 ПЗ	Лист
						75
Изм.	Лист	№ докум.	Подпись	Дата		

Файл «LoginActivity.kt»

```
package com.example.diplom.presintation.login

import android.content.Intent
import android.os.Bundle
import android.text.Editable
import android.text.TextWatcher
import android.view.inputmethod.EditorInfo
import android.widget.EditText
import android.widget.Toast
import androidx.activity.viewModels
import androidx.appcompat.app.AppCompatActivity
import androidx.lifecycle.Observer
import com.example.diplom.presintation.habitsList.HabitsActivity
import com.example.diplom.presintation.signUp.RegistrationActivity
import com.example.diplom.databinding.LoginActivityBinding
import dagger.hilt.android.AndroidEntryPoint

@AndroidEntryPoint
class LoginActivity : AppCompatActivity() {
    private val viewModel: LoginViewModel by viewModels()
    private lateinit var binding: LoginActivityBinding
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding = LoginActivityBinding.inflate(layoutInflater)
        setContentView(binding.root)

        val email = binding.idEditEmail
        val password = binding.idEditPassword
        val btnSignIn = binding.btnAuthorization
        val btnSignUp = binding.signUp

        viewModel.hasUser()

        viewModel.loginState.observe(this@LoginActivity,
Observer {
            val loginState = it ?: return@Observer
            if (loginState.isLogin) {
                val intent = Intent(this@LoginActivity,
HabitsActivity::class.java)
                startActivity(intent)
                finish()
            }

            btnSignIn.isEnabled = loginState.isValidate

            binding.inputLayoutEmail.error =
loginState.errorMessage
        })
    }
}
```

```

        binding.inputLayoutPass.error =
loginState.errorPasswordMassage
    })

    viewModel.result.observe(this@LoginActivity, Observer {
        val loginResult = it ?: return@Observer

        if (loginResult.error != null) {
            Toast.makeText(applicationContext,
loginResult.error, Toast.LENGTH_SHORT).show()
        }

        if (loginResult.success != null) {
            val intent = Intent(this@LoginActivity,
HabitsActivity::class.java)
            startActivity(intent)
            finish()
        }
    })

    email.afterTextChanged {
        viewModel.onInputChange(email.text.toString(),
password.text.toString())
    }

    password.apply {
        afterTextChanged {
            viewModel.onInputChange(email.text.toString(),
password.text.toString())
        }

        setOnEditorActionListener { _, actionId, _ ->
            when(actionId) {
                EditorInfo.IME_ACTION_DONE ->
                    viewModel.signIn(
                        email.text.toString(),
                        password.text.toString()
                    )
            }
            false
        }
    }

    btnSignIn.setOnClickListener {
        viewModel.signIn(email.text.toString(),
password.text.toString())
    }

    btnSignUp.setOnClickListener {
        val intent = Intent(this,
RegistrationActivity::class.java)

```

```

        startActivity(intent)
    }
}
private fun EditText.afterTextChanged(afterTextChanged:
(String) -> Unit) {
    this.addTextChangedListener(object : TextWatcher {
        override fun afterTextChanged(editable: Editable?) {
            afterTextChanged.invoke(editable.toString())
        }

        override fun beforeTextChanged(s: CharSequence,
start: Int, count: Int, after: Int) {}

        override fun onTextChanged(s: CharSequence, start:
Int, before: Int, count: Int) {}
    })
}
}

```

Файл «LoginViewModel.kt»

```

package com.example.diplom.presintation.login

import androidx.lifecycle.LiveData
import androidx.lifecycle.MutableLiveData
import androidx.lifecycle.ViewModel
import androidx.lifecycle.viewModelScope
import com.example.diplom.domain.models.ValidateInputType
import
com.example.diplom.domain.repository.AccountControlRepository
import
com.example.diplom.domain.useCase.ValidateLoginInputUseCase
import com.example.diplom.presintation.models.Result
import com.example.diplom.presintation.models.LoginState
import dagger.hilt.android.lifecycle.HiltViewModel
import kotlinx.coroutines.launch
import javax.inject.Inject
@HiltViewModel
class LoginViewModel @Inject constructor(
    private val validateLoginInputUseCase:
ValidateLoginInputUseCase,
    private val accountControlRepository:
AccountControlRepository
) : ViewModel() {

    private val _loginState = MutableLiveData<LoginState>()
    val loginState: LiveData<LoginState> = _loginState

    private val _Result = MutableLiveData<Result>()
    val result: LiveData<Result> = _Result
    fun signIn(email: String, password: String) {
        viewModelScope.launch {

```

```

        accountControlRepository.signIn(email, password)
        if (accountControlRepository.hasUser()) {
            _Result.value = Result(success = "Completed")

        } else {
            _Result.value = Result(error = "Error
Completed")
            _loginState.value = LoginState()
        }
    }
}
fun hasUser() {
    if (accountControlRepository.hasUser()) {
        _loginState.value = LoginState(isLogin = true)
    } else {
        _loginState.value = LoginState(isLogin = false)
    }
}
fun onInputChange(email: String, pass: String) {
    _loginState.value = LoginState(emailInput = email,
passwordInput = pass)
    checkInput()
}
private fun checkInput() {
    val validateResult = _loginState.value?.let {
        validateLoginInputUseCase(
            it.emailInput,
            it.passwordInput
        )
    }
    if (validateResult != null) {
        checkValidate(validateResult)
    }
}
private fun checkValidate(type: ValidateInputType) {
    when(type) {
        ValidateInputType.NotEmail ->
            _loginState.value = LoginState(errorMessage = "Неправильный формат почты", isValid = false)

        ValidateInputType.EmptyEmail ->
            _loginState.value = LoginState(errorMessage = "Поле пустое", isValid = false)

        ValidateInputType.EmptyPass ->
            _loginState.value =
LoginState(errorMessage = "Поле пустое", isValid =
false)

        ValidateInputType.SmallPass ->
            _loginState.value =

```

```

LoginState(errorPasswordMassage = "Слишком маленький пароль",
isValidate = false)

        ValidateInputType.Validate ->
            _loginState.value = LoginState(errorEmailMassage
= null, errorPasswordMassage = null, isValidate = true)
            else -> {

        }
    }
}
}
}

```

Файл «HabitsActivity.kt»

```

package com.example.diplom.presintation.habitsList

import android.content.Intent
import android.os.Bundle
import android.view.MenuItem
import android.widget.TextView
import androidx.appcompat.app.ActionBarDrawerToggle
import androidx.appcompat.app.AlertDialog
import androidx.appcompat.app.AppCompatActivity
import androidx.core.view.GravityCompat
import com.applandeo.materialcalendarview.CalendarView
import com.applandeo.materialcalendarview.EventDay
import
com.example.diplom.presintation.habit_crud.CreateHabitActivity
import com.example.diplom.domain.models.Habit
import com.example.diplom.HabitAdapter
import com.example.diplom.R
import com.example.diplom.databinding.ActivityDrawerBinding
import com.example.diplom.presintation.login.LoginActivity
import
com.example.diplom.presintation.user_description.UserDescription
import com.firebase.ui.firestore.FirestoreRecyclerOptions
import com.google.android.material.navigation.NavigationView
import com.google.firebase.auth.FirebaseAuth
import com.google.firebase.auth.ktx.auth
import com.google.firebase.firestore.FirebaseFirestore
import com.google.firebase.ktx.Firebase
import java.text.SimpleDateFormat
import java.util.ArrayList
import java.util.Calendar
import java.util.Locale

class HabitsActivity : AppCompatActivity(),
    NavigationView.OnNavigationItemSelectedListener {
    private lateinit var auth: FirebaseAuth
    private lateinit var binding: ActivityDrawerBinding
    private lateinit var habitAdapter: HabitAdapter

```

```

        private val dateFormat = SimpleDateFormat("yyyy-MM-dd",
Locale.getDefault())
        override fun onCreate(savedInstanceState: Bundle?) {
            super.onCreate(savedInstanceState)
            binding = ActivityDrawerBinding.inflate(layoutInflater)
            setContentView(binding.root)
            //setSupportActionBar(binding.appBarDrawer.toolbar)

            val drawer = binding.drawerLayout
            val toggle = ActionBarDrawerToggle(this, drawer,
binding.appBarDrawer.toolbar,
                R.string.navigation_drawer_open,
                R.string.navigation_drawer_close
            )
            drawer.addDrawerListener(toggle)
            toggle.syncState()
            binding.navView.setNavigationItemSelectedListener(this)

            binding.appBarDrawer.toolbar.setOnMenuItemClickListener
{
                when(it.itemId) {
                    R.id.calendar -> {
                        showCalendarDialog()
                        true
                    }
                    else -> false
                }
            }

            auth = Firebase.auth
            val user = auth.currentUser

            if (user == null)
            {
                val intent = Intent(this@HabitsActivity,
LoginActivity::class.java)
                startActivity(intent)
                finish()
            }

            val headerView = binding.navView.getHeaderView(0)
            val headerTitle =
headerView.findViewById<TextView>(R.id.textView)
            headerTitle.text = user?.email

            binding.appBarDrawer.floatingActionButton.setOnClickListener {
                intent = Intent(this@HabitsActivity,
CreateHabitActivity::class.java)
                startActivity(intent)
            }
        }
    }
}

```

```

    }
    var query =
        FirebaseFirestore.getInstance().collection("habits").document(us
        er!!.uid).collection("user_habit").orderBy("title")
    var options =
        FirestoreRecyclerOptions.Builder<Habit>().setQuery(query,
        Habit::class.java).build()
    habitAdapter = HabitAdapter(options, this)

    binding.appBarDrawer.recyclerView.adapter = habitAdapter
}

override fun onStart() {
    super.onStart()
    habitAdapter.startListening()
}

override fun onStop() {
    super.onStop()
    habitAdapter.stopListening()
}

override fun onResume() {
    super.onResume()
    habitAdapter.notifyDataSetChanged()
}

override fun onNavigationItemSelected(item: MenuItem):
Boolean {
    when(item.itemId) {
        R.id.nav_exit -> {
            FirebaseAuth.getInstance().signOut()
            val intent = Intent(this@HabitsActivity,
            LoginActivity::class.java)
            startActivity(intent)
            finish()
        }
        R.id.nav_about_me -> {
            val intent = Intent(this@HabitsActivity,
            UserDescription::class.java)
            startActivity(intent)
        }
    }
    binding.drawerLayout.closeDrawer(GravityCompat.START)
    return true
}

private fun showCalendarDialog() {
    val dialogView =
        layoutInflater.inflate(R.layout.dialog_calendar, null)
    val calendarView =

```

```

dialogView.findViewById<CalendarView>(R.id.calendarView)

        val dialog = AlertDialog.Builder(this)
            .setView(dialogView)
            .setPositiveButton("OK") { dialog, _ ->
dialog.dismiss() }
            .create()

        updateCalendar(calendarView)

        dialog.show()
    }

    private fun updateCalendar(calendarView: CalendarView) {
        val sharedPreferences =
getSharedPreferences("marked_dates", MODE_PRIVATE)
        val markedDates = sharedPreferences.all.keys

        val events = ArrayList<EventDay>()
        for (date in markedDates) {
            if (sharedPreferences.getBoolean(date, false)) {
                val calendar = Calendar.getInstance()
                calendar.time = dateFormat.parse(date)!!
                events.add(EventDay(calendar, R.drawable.done))
            }
        }
        calendarView.setEvents(events)
    }
}

```

Файл «CreateHabitActivity.kt»

```

package com.example.diplom.presintation.habit_crud

import android.app.TimePickerDialog
import android.content.Intent
import android.os.Bundle
import android.util.Log
import androidx.appcompat.app.AppCompatActivity
import androidx.work.ExistingPeriodicWorkPolicy
import androidx.work.PeriodicWorkRequestBuilder
import androidx.work.WorkManager
import com.example.diplom.R
import com.example.diplom.domain.models.Habit
import com.example.diplom.databinding.CustomizingHabitBinding
import com.example.diplom.presintation.habitsList.HabitsActivity
import
com.example.diplom.presintation.notification.NotificationWorker
import com.google.firebase.auth.FirebaseAuth
import com.google.firebase.firestore.FirebaseFirestore
import java.text.SimpleDateFormat
import java.util.Calendar

```

```

import java.util.concurrent.TimeUnit

class CreateHabitActivity : AppCompatActivity() {
    private lateinit var binding: CustomizingHabitBinding
    private var isEditMode = false
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding =
CustomizingHabitBinding.inflate(layoutInflater)
        setContentView(binding.root)

        binding.toolbar.menu.findItem(R.id.delete).isVisible =
false

binding.editTitle.setText(intent.extras?.getString("title") ?:
"")

binding.editDescription.setText(intent.extras?.getString("descri
ption") ?: "")

binding.autoCompleteText.setText(intent.extras?.getString("typeH
abit") ?: "")

binding.timePicker.setText(intent.extras?.getString("timeToNotif
y") ?: "")
        val docId = intent.extras?.getString("id")
        if(!docId.isNullOrEmpty()) {
            isEditMode = true
        }

        if(isEditMode) {
            binding.toolbar.title = "Редактирование"
            binding.addButton.text = "Изменить"
            binding.toolbar.menu.findItem(R.id.delete).isVisible
= true
        }

        binding.toolbar.setOnMenuItemClickListener {
            when(it.itemId) {
                R.id.delete -> {
                    val user =
FirebaseAuth.getInstance().currentUser
                    if (!docId.isNullOrEmpty()) {

cancelDailyNotification(binding.timePicker.text.toString())
                    var documentReference =

FirebaseFirestore.getInstance().collection("habits").document(us
er!!.uid).collection("user_habit").document(docId)

```

```

        documentReference.delete()
    }
    val intent = Intent(this,
HabitsActivity::class.java)
    startActivity(intent)
    finish()
    true
}
else -> false
}
}

binding.toolbar.setNavigationOnClickListener {
    val intent = Intent(this,
HabitsActivity::class.java)
    startActivity(intent)
    finish()
}

binding.timePicker.setOnClickListener {

    val cal = Calendar.getInstance()
    val timePickerDialog =
TimePickerDialog.OnTimeSetListener { _, hour, minute ->
        cal.set(Calendar.HOUR_OF_DAY, hour)
        cal.set(Calendar.MINUTE, minute)
        cal.set(Calendar.SECOND, 0)
        val format =
SimpleDateFormat("HH:mm").format(cal.time)
        binding.timePicker.setText(format)
        scheduleDailyNotifcation(cal.timeInMillis -
System.currentTimeMillis(), format)
    }
    val timePicker =
TimePickerDialog(this@CreateHabitActivity, timePickerDialog,
cal.get(Calendar.HOUR_OF_DAY), cal.get(Calendar.MINUTE), true)
    timePicker.show()
}
binding.addButton.setOnClickListener{
    val title = binding.editTitle.text.toString()
    if (title == null || title.isEmpty()) {
        binding.titleInputLayout.error = "Напишите
название"

        return@setOnClickListener

    }

    val typeHabit =
binding.autoCompleteText.text.toString()
    if (typeHabit == null || typeHabit.isEmpty()) {
        binding.typeInputLayout.error = "Выберите тип
привычки"
    }
}

```

```

        return@setOnClickListener
    }

    val time = binding.timePicker.text.toString()
    val description =
binding.editDescription.text.toString()
    val habit = Habit(title = title, typeHabit =
typeHabit, timeToNotify = time, description = description)
    val user = FirebaseAuth.getInstance().currentUser
    var documentReference = if (isEditMode &&
!docId.isNullOrEmpty()) {

FirebaseFirestore.getInstance().collection("habits").document(us
er!!.uid).collection("user_habit").document(docId)
    } else {

FirebaseFirestore.getInstance().collection("habits").document(us
er!!.uid).collection("user_habit").document()
    }

    documentReference.set(habit)
    val intent = Intent(this,
HabitsActivity::class.java)
    startActivity(intent)
    finish()
    }
}

private fun scheduleDailyNotification(initialDelay: Long,
format: String) {
    val uniqueWorkName = "DailyNotification_${format}"
    val dailyWorkRequest =
PeriodicWorkRequestBuilder<NotificationWorker>(1, TimeUnit.DAYS)
        .setInitialDelay(initialDelay,
TimeUnit.MILLISECONDS)
        .build()

    WorkManager.getInstance(this).enqueueUniquePeriodicWork(
        uniqueWorkName,
        ExistingPeriodicWorkPolicy.REPLACE,
        dailyWorkRequest
    )
}

private fun cancelDailyNotification(format: String) {
    WorkManager.getInstance(this).cancelUniqueWork("DailyNotificatio
n_${format}")
}
}

```

Файл «LoginState.kt»

					ПГУ1.090301.21.001 ПЗ	Лист
						86
Изм.	Лист	№ докум.	Подпись	Дата		

```
package com.example.diplom.presintation.models

data class LoginState (
    val emailInput: String = "",
    val passwordInput: String = "",
    val isEmailValidate: Boolean = false,
    val isPassValidate: Boolean = false,
    val isValiddate: Boolean = false,
    val errorPasswordMessage: String? = null,
    val errorEmailMessage: String? = null,
    val isLogin: Boolean = false
)
```

Файл «Result.kt»

```
package com.example.diplom.presintation.models

data class Result (
    val success: String? = null,
    val error: String? = null
)
```

Файл «SignUpState.kt»

```
package com.example.diplom.presintation.models

data class SignUpState(
    val emailInput: String = "",
    val passwordInput: String = "",
    val confirmInput: String = "",
    val isEmailValidate: Boolean = false,
    val isPassValidate: Boolean = false,
    val isConfirmValidate: Boolean = false,
    val isValiddate: Boolean = false,
    val errorPasswordMessage: String? = null,
    val errorEmailMessage: String? = null,
    val errorConfirmMessage: String? = null
)
```

Файл «NotificationWorker.kt»

```
package com.example.diplom.presintation.notification

import android.app.NotificationChannel
import android.app.NotificationManager
import android.content.Context
import androidx.core.app.NotificationCompat
import androidx.work.Worker
import androidx.work.WorkerParameters

class NotificationWorker(context: Context, workerParameters:
WorkerParameters) : Worker(context, workerParameters) {
    override fun doWork(): Result {
        val notificationManager =
```

```

applicationContext.getSystemService(Context.NOTIFICATION_SERVICE
) as NotificationManager

        if (android.os.Build.VERSION.SDK_INT >=
android.os.Build.VERSION_CODES.O) {
            val channel = NotificationChannel("notify",
"Notification Channel", NotificationManager.IMPORTANCE_DEFAULT)

notificationManager.createNotificationChannel(channel)
        }

        val builder =
NotificationCompat.Builder(applicationContext, "notify")
            .setSmallIcon(android.R.drawable.ic_dialog_info)
            .setContentTitle("Привычка")
            .setContentText("Время для того чтобы выполнить
процедуры!")
            .setPriority(NotificationCompat.PRIORITY_DEFAULT)

        notificationManager.notify((System.currentTimeMillis() /
1000).toInt(), builder.build())

        return Result.success()
    }
}

```

Файл «RegistrationActivity.kt»

```

package com.example.diplom.presintation.signUp

import android.content.Intent
import android.os.Bundle
import android.text.Editable
import android.text.TextUtils
import android.text.TextWatcher
import android.util.Patterns
import android.view.inputmethod.EditorInfo
import android.widget.EditText
import android.widget.Toast
import androidx.activity.viewModels
import androidx.appcompat.app.AppCompatActivity
import androidx.lifecycle.Observer
import com.example.diplom.presintation.habitsList.HabitsActivity
import
com.example.diplom.databinding.RegistrationActivityBinding
import com.google.firebase.auth.FirebaseAuth
import com.google.firebase.auth.ktx.auth
import com.google.firebase.ktx.Firebase
import dagger.hilt.android.AndroidEntryPoint

@AndroidEntryPoint

```

```

class RegistrationActivity : AppCompatActivity() {
    private lateinit var binding: RegistrationActivityBinding
    private val viewModel: SignUpViewModel by viewModels()
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding =
RegistrationActivityBinding.inflate(layoutInflater)
        setContentView(binding.root)

        binding.idToolbar.setNavigationOnClickListener {
            onBackPressed()
        }

        val email = binding.idEditEmail
        val password = binding.idEditPassword
        val confirmPassword = binding.idEditConfirmPassword
        val login = binding.enterButton

        viewModel.signUpState.observe(this, Observer {
            val state = it ?: return@Observer

            login.isEnabled = state.isValidDate

            binding.editInputLayout.error =
state.errorMessage

            binding.passInputLayout.error =
state.errorPasswordMessage

            binding.confirmInputLayout.error =
state.errorConfirmMessage
        })

        viewModel.signUpResult.observe(this, Observer {
            val result = it ?: return@Observer

            if (result.error != null) {
                Toast.makeText(applicationContext, result.error,
Toast.LENGTH_SHORT).show()
            }

            if (result.success != null) {
                val intent = Intent(this@RegistrationActivity,
HabitsActivity::class.java)
                startActivity(intent)
                finish()
            }
        })

        email.afterTextChanged {
            viewModel.onInputChange(email.text.toString(),

```

```

password.text.toString(), confirmPassword.text.toString())
    }

    password.afterTextChanged {
        viewModel.onInputChange(email.text.toString(),
password.text.toString(), confirmPassword.text.toString())
    }

    confirmPassword.apply {
        afterTextChanged {
            viewModel.onInputChange(email.text.toString(),
password.text.toString(), confirmPassword.text.toString())
        }

        setOnEditorActionListener { _, actionId, _ ->
            when(actionId) {
                EditorInfo.IME_ACTION_DONE ->
                    viewModel.signUp(
                        email.text.toString(),
                        confirmPassword.text.toString()
                    )
            }
            false
        }
    }

    login.setOnClickListener {
        viewModel.signUp(email.text.toString(),
confirmPassword.text.toString())
    }
}

private fun EditText.afterTextChanged(afterTextChanged:
(String) -> Unit) {
    this.addTextChangedListener(object : TextWatcher {
        override fun afterTextChanged(editable: Editable?) {
            afterTextChanged.invoke(editable.toString())
        }

        override fun beforeTextChanged(s: CharSequence,
start: Int, count: Int, after: Int) {}

        override fun onTextChanged(s: CharSequence, start:
Int, before: Int, count: Int) {}
    })
}
}

```

Файл «SignUpViewModel.kt»

```

package com.example.diplom.presintation.signUp

import androidx.lifecycle.LiveData

```

					ПГУ1.090301.21.001 ПЗ	Лист
						90
Изм.	Лист	№ докум.	Подпись	Дата		

```

import androidx.lifecycle.MutableLiveData
import androidx.lifecycle.ViewModel
import androidx.lifecycle.viewModelScope
import com.example.diplom.domain.models.ValidateInputType
import
com.example.diplom.domain.repository.AccountControlRepository
import
com.example.diplom.domain.useCase.ValidateSignUpInputUseCase
import com.example.diplom.presentation.models.Result
import com.example.diplom.presentation.models.SignUpState
import dagger.hilt.android.lifecycle.HiltViewModel
import kotlinx.coroutines.launch
import javax.inject.Inject

@HiltViewModel
class SignUpViewModel @Inject constructor(
    private val validateSignUpInputUseCase:
ValidateSignUpInputUseCase,
    private val accountControlRepository:
AccountControlRepository
) : ViewModel() {
    private val _signUpState = MutableLiveData<SignUpState>()
    val signUpState: LiveData<SignUpState> = _signUpState

    private val _signUpResult = MutableLiveData<Result>()
    val signUpResult : LiveData<Result> = _signUpResult

    fun signUp(email: String, password: String) {
        viewModelScope.launch {
            accountControlRepository.signUp(email, password)
            if (accountControlRepository.hasUser()) {
                _signUpResult.value = Result(success =
"Completed")

            } else {
                _signUpResult.value = Result(error = "Error
Completed")
            }
        }
    }

    fun onInputChange(email: String, pass: String, confirm:
String) {
        _signUpState.value = SignUpState(emailInput = email,
passwordInput = pass, confirmInput = confirm)
        checkInput()
    }

    private fun checkInput() {
        val validateResult = _signUpState.value?.let {
            validateSignUpInputUseCase(

```

```

        it.emailInput,
        it.passwordInput,
        it.confirmInput
    )
}
if (validateResult != null) {
    checkValidate(validateResult)
}
}

private fun checkValidate(type: ValidateInputType) {
    when(type) {
        ValidateInputType.NotEmail ->
            _signUpState.value =
SignUpState(errorEmailMessage = "Неправильный формат почты",
isValidate = false)

        ValidateInputType.EmptyEmail ->
            _signUpState.value =
SignUpState(errorEmailMessage = "Поле пустое", isValidate =
false)

        ValidateInputType.EmptyPass ->
            _signUpState.value =
SignUpState(errorPasswordMessage = "Поле пустое", isValidate =
false)

        ValidateInputType.SmallPass ->
            _signUpState.value =
SignUpState(errorPasswordMessage = "Слишком маленький пароль",
isValidate = false)

        ValidateInputType.EmptyConfirm ->
            _signUpState.value =
SignUpState(errorConfirmMessage = "Поле пустое", isValidate =
false)

        ValidateInputType.NotConfirm ->
            _signUpState.value =
SignUpState(errorConfirmMessage = "Пароль не совпадает",
isValidate = false)

        ValidateInputType.Validate ->
            _signUpState.value =
SignUpState(errorEmailMessage = null, errorPasswordMessage =
null, errorConfirmMessage = null, isValidate = true)
    }
}
}

```

Файл «HabitAdapter.kt»

```

package com.example.diplom

import android.content.Context
import android.content.Intent
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import androidx.appcompat.app.AppCompatActivity
import androidx.recyclerview.widget.RecyclerView
import com.example.diplom.domain.models.Habit
import com.example.diplom.databinding.HabitTemplateBinding
import
com.example.diplom.presintation.habit_crud.CreateHabitActivity
import com.firebase.ui.firestore.FirestoreRecyclerAdapter
import com.firebase.ui.firestore.FirestoreRecyclerOptions
import com.google.android.material.materialswitch.MaterialSwitch
import com.google.firebase.auth.FirebaseAuth
import com.google.firebase.firestore.FirebaseFirestore
import java.text.SimpleDateFormat
import java.util.Date
import java.util.Locale

class HabitAdapter(options: FirestoreRecyclerOptions<Habit>,
context: Context) : FirestoreRecyclerAdapter<Habit,
HabitAdapter.HabitViewHolder>(
options) {
    private val context = context
    private val dateFormat = SimpleDateFormat("yyyy-MM-dd",
Locale.getDefault())
    class HabitViewHolder(item: View):
RecyclerView.ViewHolder(item) {
        private val binding = HabitTemplateBinding.bind(item)
        fun bind(habit: Habit) = with(binding) {
            title.text = habit.title
            typeHabit.text = habit.typeHabit
            description.text = habit.description
            switchStatus.isChecked = habit.switchState
            switchStatus.setOnCheckedChangeListener {
buttonView, isChecked ->
                habit.switchState = isChecked
            }
        }
    }

    override fun onCreateViewHolder(parent: ViewGroup, viewType:
Int): HabitViewHolder {
        val view =
LayoutInflater.from(parent.context).inflate(R.layout.habit_templ
ate, parent, false)
        return HabitViewHolder(view)
    }

```

					ПГУ1.090301.21.001 ПЗ	Лист
						93
Изм.	Лист	№ докум.	Подпись	Дата		

```

        override fun onBindViewHolder(holder: HabitViewHolder,
position: Int, model: Habit) {
            holder.bind(model)

holder.itemView.findViewById<MaterialSwitch>(R.id.switchStatus).
setOnCheckedChangeListener { buttonView, isChecked ->
            val user = FirebaseAuth.getInstance().currentUser
            if
(!this.snapshots.getSnapshot(position).id.isNullOrEmpty()) {
                model.switchState = isChecked
                val currentDate = dateFormat.format(Date())
                val sharedPreferences =
holder.itemView.context.getSharedPreferences("marked_dates",
                    AppCompatActivity.MODE_PRIVATE
                )
                val editor = sharedPreferences.edit()
                editor.putBoolean(currentDate, true)
                editor.apply()
                var documentReference =

FirebaseFirestore.getInstance().collection("habits").document(us
er!!.uid).collection("user_habit").document(this.snapshots.getSn
apshot(position).id)
                    documentReference.set(model)
                }
            }
            holder.itemView.setOnLongClickListener {
                val intent = Intent(context,
CreateHabitActivity::class.java)
                intent.putExtra("title", model.title)
                intent.putExtra("typeHabit", model.typeHabit)
                intent.putExtra("description", model.description)
                intent.putExtra("timeToNotify", model.timeToNotify)
                intent.putExtra("id",
this.snapshots.getSnapshot(position).id)

                context.startActivity(intent)
                true
            }
        }
    }
}

```

Файл «Habit.kt»

```

package com.example.diplom.domain.models

data class Habit(var title: String = "",
                var typeHabit: String = "",
                var timeToNotify: String = "",
                var description: String = "",

```

```

        var switchState: Boolean = false
    )

```

Файл «ValidateInputType.kt»

```
package com.example.diplom.domain.models
```

```
enum class ValidateInputType {
    EmptyPass,
    EmptyEmail,
    NotEmail,
    SmallPass,
    EmptyConfirm,
    NotConfirm,
    Validate
}
```

Файл «User.kt»

```
package com.example.diplom.domain.models
```

```
data class User(
    val id: String? = "",
    val email: String? = ""
)
```

Файл «ValidateLoginInputUseCase.kt»

```
package com.example.diplom.domain.useCase
```

```
import android.util.Patterns
import com.example.diplom.domain.models.ValidateInputType
```

```
class ValidateLoginInputUseCase {
    operator fun invoke(email: String, password: String) :
    ValidateInputType {
        if (email.isEmpty()) {
            return ValidateInputType.EmptyEmail
        }
        if (!Patterns.EMAIL_ADDRESS.matcher(email).matches()) {
            return ValidateInputType.NotEmail
        }
        if (password.isEmpty()) {
            return ValidateInputType.EmptyPass
        }
        if (password.length < 6) {
            return ValidateInputType.SmallPass
        }

        return ValidateInputType.Validate
    }
}
```

Файл «ValidateSignUpInputUseCase.kt»

```

package com.example.diplom.domain.useCase

import android.util.Patterns
import com.example.diplom.domain.models.ValidateInputType

class ValidateSignUpInputUseCase {
    operator fun invoke(email: String, password: String,
confirm: String) : ValidateInputType {
        if (email.isEmpty()) {
            return ValidateInputType.EmptyEmail
        }
        if (!Patterns.EMAIL_ADDRESS.matcher(email).matches()) {
            return ValidateInputType.NotEmail
        }
        if (password.isEmpty()) {
            return ValidateInputType.EmptyPass
        }
        if (confirm.isEmpty()) {
            return ValidateInputType.EmptyConfirm
        }
        if (password.length < 6) {
            return ValidateInputType.SmallPass
        }
        if (password != confirm) {
            return ValidateInputType.NotConfirm
        }
        return ValidateInputType.Validate
    }
}

```

Файл «AccountControlRepository.kt»

```

package com.example.diplom.domain.repository

import com.example.diplom.domain.models.User
import com.google.firebase.ktx.Firebase
import kotlinx.coroutines.flow.Flow

interface AccountControlRepository {
    var user: User?
    fun hasUser(): Boolean
    suspend fun signIn(email: String, password: String)
    suspend fun signUp(email:String, password: String)
    fun signOut()
}

```

Файл «AccountControlRepositoryImpl.kt»

```

package com.example.diplom.data.repository

import android.util.Log
import com.example.diplom.domain.models.User
import

```

```

com.example.diplom.domain.repository.AccountControlRepository
import com.google.firebase.auth.ktx.auth
import com.google.firebase.ktx.Firebase
import kotlinx.coroutines.tasks.await

class AccountControlRepositoryImpl : AccountControlRepository {
    override var user: User? = null
        get() = User(Firebase.auth.currentUser?.uid,
        Firebase.auth.currentUser?.email)
    override fun hasUser(): Boolean {
        return Firebase.auth.currentUser != null
    }
    override suspend fun signIn(email: String, password: String)
    {
        Firebase.auth.signInWithEmailAndPassword(email,
        password).await()
    }
    override suspend fun signUp(email: String, password:
    String){
        Firebase.auth.createUserWithEmailAndPassword(email,
        password).await()
    }
    override fun signOut() {
        Firebase.auth.signOut()
        user = null
    }
}

```

Файл «DomainModule.kt»

```

package com.example.diplom.di

import
com.example.diplom.domain.useCase.ValidateLoginInputUseCase
import
com.example.diplom.domain.useCase.ValidateSignUpInputUseCase
import dagger.Module
import dagger.Provides
import dagger.hilt.InstallIn
import dagger.hilt.components.SingletonComponent
import javax.inject.Singleton

@Module
@InstallIn(SingletonComponent::class)
object DomainModule {
    @Provides
    @Singleton
    fun provideValidateLoginInputCase() :
    ValidateLoginInputUseCase {
        return ValidateLoginInputUseCase()
    }
    @Provides

```

```

@Singleton
fun provideValidateSignUpInputCase() :
ValidateSignUpInputUseCase {
    return ValidateSignUpInputUseCase()
}
}

```

Файл «DataModule.kt»

```

package com.example.diplom.di

import
com.example.diplom.data.repository.AccountControlRepositoryImpl
import
com.example.diplom.domain.repository.AccountControlRepository
import dagger.Module
import dagger.Provides
import dagger.hilt.InstallIn
import dagger.hilt.components.SingletonComponent
import javax.inject.Singleton
@Module
@InstallIn(SingletonComponent::class)
object DataModule {
    @Provides
    @Singleton
    fun provideAccountControlRepository() :
AccountControlRepository {
        return AccountControlRepositoryImpl()
    }
}

```

Файл «Activity_drawer.xml»

```

<?xml version="1.0" encoding="utf-8"?>
<androidx.drawerlayout.widget.DrawerLayout
xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:app="http://schemas.android.com/apk/res-auto"
xmlns:tools="http://schemas.android.com/tools"
android:id="@+id/drawer_layout"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:fitsSystemWindows="true"
tools:openDrawer="start">

    <include
        android:id="@+id/app_bar_drawer"
        layout="@layout/habits_activity"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />

    <com.google.android.material.navigation.NavigationView
        android:id="@+id/nav_view"
        android:layout_width="wrap_content"

```

```

        android:layout_height="match_parent"
        android:layout_gravity="start"
        android:fitsSystemWindows="true"
        app:headerLayout="@layout/nav_header_drawer"
        app:menu="@menu/activity_menu_drawer" />
</androidx.drawerlayout.widget.DrawerLayout>

```

Файл «customizing_habit.xml»

```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <com.google.android.material.appbar.AppBarLayout
        android:id="@+id/appBarLayout"
        android:layout_width="match_parent"
        android:layout_height="wrap_content">

        <com.google.android.material.appbar.MaterialToolbar
            android:id="@+id/toolbar"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:minHeight="?attr/actionBarSize"
            app:navigationIcon="@drawable/arrow_back"
            app:navigationIconTint="?attr/colorControlNormal"
            app:menu="@menu/change_top_app_bar_menu"
            app:title="Новая привычка"
            app:titleCentered="true" />
    </com.google.android.material.appbar.AppBarLayout>

    <androidx.constraintlayout.widget.ConstraintLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:padding="16dp">

        <com.google.android.material.textfield.TextInputLayout
            android:id="@+id/titleInputLayout"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:hint="Название"

            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="parent">

        <com.google.android.material.textfield.TextInputEditText
            android:id="@+id/editTitle"

```

```

        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:imeOptions="actionNext"
        android:inputType="text"/>
</com.google.android.material.textfield.TextInputLayout>

<com.google.android.material.textfield.TextInputLayout
    android:id="@+id/typeInputLayout"

style="@style/Widget.Material3.TextInputLayout.OutlinedBox.ExposedDropdownMenu"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="8dp"
    android:hint="Тип привычки"

app:startIconDrawable="@drawable/format_list_bulleted"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"

app:layout_constraintTop_toBottomOf="@+id/titleInputLayout">

    <AutoCompleteTextView
        android:id="@+id/autoCompleteText"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:inputType="none"
        app:simpleItems="@array/list_of_types" />
</com.google.android.material.textfield.TextInputLayout>

<com.google.android.material.textfield.TextInputLayout
    android:id="@+id/timeInputLayout"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="Время"
    app:startIconDrawable="@drawable/schedule"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"

app:layout_constraintTop_toBottomOf="@+id/typeInputLayout">

<com.google.android.material.textfield.TextInputEditText
    android:id="@+id/timePicker"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:inputType="none" />
</com.google.android.material.textfield.TextInputLayout>

<com.google.android.material.textfield.TextInputLayout
    android:id="@+id/textInputLayout6"

```

```

        style="?attr/textInputFilledStyle"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="8dp"
        android:hint="Описание"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"

app:layout_constraintTop_toBottomOf="@+id/timeInputLayout">

<com.google.android.material.textfield.TextInputEditText
    android:id="@+id/editDescription"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" />
</com.google.android.material.textfield.TextInputLayout>

<com.google.android.material.button.MaterialButton
    android:id="@+id/addButton"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/add_habit"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
</LinearLayout>

```

Файл «dialog_calendar.xml»

```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="16dp">

    <com.applandeo.materialcalendarview.CalendarView
        android:id="@+id/calendarView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        app:headerColor="@color/md_theme_primary"/>
</LinearLayout>

```

Файл «habit_template.xml»

```

<?xml version="1.0" encoding="utf-8"?>
<com.google.android.material.card.MaterialCardView
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"

```

```

        android:layout_height="wrap_content"
        android:id="@+id/cardView"

        android:backgroundTint="@color/md_theme_surfaceContainer_mediumContrast"
        android:layout_marginTop="8dp"
        android:layout_marginBottom="8dp">
        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:orientation="vertical"
            android:padding="16dp">
            <LinearLayout
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:orientation="horizontal">

            <com.google.android.material.textview.MaterialTextView
                android:id="@+id/title"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Привычка"
                android:textSize="20dp"
                android:layout_weight="1"/>

            <com.google.android.material.materialswitch.MaterialSwitch
                android:id="@+id/switchStatus"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"/>
            </LinearLayout>
            <com.google.android.material.textview.MaterialTextView
                android:id="@+id/typeHabit"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Тело" />
            <com.google.android.material.textview.MaterialTextView
                android:id="@+id/description"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_marginTop="8dp"
                android:textSize="11dp"
                android:text="Описание конкретной привычки"/>
        </LinearLayout>
    </com.google.android.material.card.MaterialCardView>

```

Файл «habits_activity.xml»

```

<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"

```

```

        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <androidx.recyclerview.widget.RecyclerView
            android:id="@+id/recyclerView"
            android:layout_width="0dp"
            android:layout_height="0dp"
            android:padding="16dp"

app:layoutManager="androidx.recyclerview.widget.LinearLayoutManager"

            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toBottomOf="@+id/appBarLayout"

        />

        <com.google.android.material.appbar.AppBarLayout
            android:id="@+id/appBarLayout"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="parent">

            <com.google.android.material.appbar.MaterialToolbar
                android:id="@+id/toolbar"
                app:menu="@menu/habit_activity_menu"
                android:layout_width="match_parent"
                android:layout_height="?attr/actionBarSize"
                app:title="ПРИВЫЧКИ"
                app:titleCentered="true"/>
            </com.google.android.material.appbar.AppBarLayout>

        <com.google.android.material.floatingactionbutton.FloatingAction
        Button
            android:id="@+id/floatingActionButton"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_margin="16dp"
            android:clickable="true"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintEnd_toEndOf="parent"
            app:srcCompat="@drawable/add" />
    </androidx.constraintlayout.widget.ConstraintLayout>

```

Файл «login_activity.xml»

					ПГУ1.090301.21.001 ПЗ	Лист
						103
Изм.	Лист	№ докум.	Подпись	Дата		

```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:gravity="center"
    android:padding="16dp">

    <androidx.constraintlayout.widget.ConstraintLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content">

        <com.google.android.material.textfield.TextInputLayout
            android:id="@+id/inputLayoutEmail"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_marginLeft="16dp"
            android:layout_marginRight="16dp"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="parent"
            app:startIconDrawable="@drawable/mail">

            <com.google.android.material.textfield.TextInputEditText
                android:id="@+id/idEditEmail"
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:hint="@string/str_input_email"
                android:imeOptions="actionNext"
                android:inputType="textEmailAddress"
                android:padding="16dp" />
        </com.google.android.material.textfield.TextInputLayout>

        <com.google.android.material.textfield.TextInputLayout
            android:id="@+id/inputLayoutPass"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_margin="16dp"
            app:startIconDrawable="@drawable/lock"
            app:endIconMode="password_toggle"
            app:errorEnabled="true"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintStart_toStartOf="parent"

            app:layout_constraintTop_toBottomOf="@+id/inputLayoutEmail">

```

```

<com.google.android.material.textfield.TextInputEditText
    android:id="@+id/idEditPassword"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:padding="16dp"
    android:hint="@string/str_input_password"
    android:imeOptions="actionDone"
    android:inputType="textPassword"

    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
</com.google.android.material.textfield.TextInputLayout>

<com.google.android.material.button.MaterialButton
    android:id="@+id/btn_authorization"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginLeft="16dp"
    android:layout_marginRight="16dp"
    android:text="@string/str_authorization"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"

app:layout_constraintTop_toBottomOf="@+id/inputLayoutPass" />

<com.google.android.material.textview.MaterialTextView
    android:id="@+id/signUp"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginLeft="16dp"
    android:layout_marginRight="16dp"
    android:fontFamily="@font/roboto_regular"
    android:gravity="center"
    android:paddingTop="10dp"
    android:paddingBottom="10dp"
    android:text="@string/str_registration"
    android:textSize="11sp"
    android:textStyle="bold"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"

app:layout_constraintTop_toBottomOf="@+id/btn_authorization" />
</androidx.constraintlayout.widget.ConstraintLayout>
</LinearLayout>

```

Файл «nav_header_drawer.xml»

```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"

```

```

xmlns:app="http://schemas.android.com/apk/res-auto"
android:id="@+id/header_title"
android:layout_width="match_parent"
android:layout_height="180dp"
android:background="@color/md_theme_outlineVariant"
android:gravity="center"
android:orientation="vertical">

<ImageView
    android:id="@+id/imageView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"

android:contentDescription="@string/nav_header_image_desc"
    app:srcCompat="@drawable/account_icon_circle" />

    <com.google.android.material.textview.MaterialTextView
        android:id="@+id/textView"
        android:layout_margin="4dp"
        android:textSize="18dp"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"/>
</LinearLayout>

```

Файл «registration_activity.xml»

```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <com.google.android.material.appbar.AppBarLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content">
        <com.google.android.material.appbar.MaterialToolbar
            android:id="@+id/idToolbar"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:minHeight="?attr/actionBarSize"
            app:navigationIcon="@drawable/arrow_back"
            app:navigationIconTint="?attr/colorControlNormal"
            app:title="@string/str_title_toolBar_for_registration"
            app:titleCentered="true"/>
    </com.google.android.material.appbar.AppBarLayout>
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"

```

```

        android:orientation="vertical"
        android:layout_weight="1"
        android:padding="16dp"
        android:gravity="center">

        <com.google.android.material.textfield.TextInputLayout
            android:id="@+id/editInputLayout"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginBottom="8dp"
            app:startIconDrawable="@drawable/mail">

        <com.google.android.material.textfield.TextInputEditText
            android:id="@+id/idEditEmail"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"

            android:hint="@string/str_input_email"
            android:imeOptions="actionNext"
            android:inputType="textEmailAddress"/>
        </com.google.android.material.textfield.TextInputLayout>

        <com.google.android.material.textfield.TextInputLayout
            android:id="@+id/passInputLayout"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginBottom="8dp"
            app:endIconMode="password_toggle"
            app:startIconDrawable="@drawable/lock">

        <com.google.android.material.textfield.TextInputEditText
            android:id="@+id/idEditPassword"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"

            android:hint="@string/str_input_password"
            android:imeOptions="actionNext"
            android:inputType="textPassword"/>
        </com.google.android.material.textfield.TextInputLayout>

        <com.google.android.material.textfield.TextInputLayout
            android:id="@+id/confirmInputLayout"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginBottom="8dp"
            app:endIconMode="password_toggle"
            app:startIconDrawable="@drawable/lock">

```

```

<com.google.android.material.textfield.TextInputEditText
    android:id="@+id/idEditConfirmPassword"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"

    android:hint="@string/str_confirm_password"
    android:imeOptions="actionDone"
    android:inputType="textPassword" />
</com.google.android.material.textfield.TextInputLayout>

</LinearLayout>
<com.google.android.material.button.MaterialButton
    android:id="@+id/enterButton"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginLeft="16dp"
    android:layout_marginBottom="16dp"
    android:layout_marginRight="16dp"
    android:text="@string/str_authorization"
    android:enabled="true"/>
</LinearLayout>

```

Файл «user_description.xml»

```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <com.google.android.material.appbar.AppBarLayout
        android:id="@+id/appBarLayout"
        android:layout_width="match_parent"
        android:layout_height="wrap_content">

        <com.google.android.material.appbar.MaterialToolbar
            android:id="@+id/toolbar"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:minHeight="?attr/actionBarSize"
            app:navigationIcon="@drawable/arrow_back"
            app:navigationIconTint="?attr/colorControlNormal"
            app:title="Обо мне"
            app:titleCentered="true" />
    </com.google.android.material.appbar.AppBarLayout>

    <androidx.constraintlayout.widget.ConstraintLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"

```

```

        android:padding="16dp">

        <com.google.android.material.textfield.TextInputLayout
            android:id="@+id/titleInputLayout"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:hint="Возраст"

            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="parent">

        <com.google.android.material.textfield.TextInputEditText
            android:id="@+id/editAge"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:imeOptions="actionNext"
            android:inputType="numberDecimal"/>
        </com.google.android.material.textfield.TextInputLayout>

        <com.google.android.material.textfield.TextInputLayout
            android:id="@+id/typeInputLayout"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:hint="Век"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintStart_toStartOf="parent"

            app:layout_constraintTop_toBottomOf="@+id/titleInputLayout">

        <com.google.android.material.textfield.TextInputEditText
            android:id="@+id/editWeight"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:inputType="none" />
        </com.google.android.material.textfield.TextInputLayout>

        <com.google.android.material.textfield.TextInputLayout
            android:id="@+id/timeInputLayout"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:hint="Рост"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintStart_toStartOf="parent"

            app:layout_constraintTop_toBottomOf="@+id/typeInputLayout">

        <com.google.android.material.textfield.TextInputEditText

```

```

        android:id="@+id/editHeight"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:inputType="none" />
</com.google.android.material.textfield.TextInputLayout>

<com.google.android.material.textfield.TextInputLayout
    android:id="@+id/textInputLayout6"
    style="?attr/textInputFilledStyle"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="8dp"
    android:hint="Применяемые средства"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"

app:layout_constraintTop_toBottomOf="@+id/timeInputLayout">

<com.google.android.material.textfield.TextInputEditText
    android:id="@+id/editDescription"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" />
</com.google.android.material.textfield.TextInputLayout>

<com.google.android.material.button.MaterialButton
    android:id="@+id/addButton"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Отправить"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
</LinearLayout>

```

Приложение В

(Обязательно)

Стартап-проект. Приложение-трекер «Привычки красоты» для ОС Android

Презентация

					<i>ПГУ1.090301.21.001 ПЗ</i>	Лист
						111
Изм.	Лист	№ докум.	Подпись	Дата		

Стартап-проект. Приложение-трекер «Привычки красоты» для ОС Android

Автор работы: студент группы 20ВВП1 Папшев Н.С.

Научный руководитель: к.т.н., доцент кафедры ВТ Юрова О.В.

Пенза, 2024

Актуальность проекта

В современном мире люди все чаще начинают следить за своим здоровьем, питанием и состоянием собственного тела. Но для этого человеку приходится прилагать усилия, причем чаще всего эти усилия заключаются в систематических повторениях каких-либо действий. При должном внимании и ежедневной работе у человека вырабатываются полезные привычки.

Трекер-привычек может помочь людям отслеживать свой прогресс, а для удобства пользователей в качестве трекера-привычек можно использовать мобильное приложение, так как мобильное устройство всегда находится под рукой у человека.

Цели работы

Цель работы – необходимо разработать приложение-трекер для телефонов на базе операционной системы Android, позволяющее вырабатывать привычки для ухода за собственным телом.

В приложении должно быть реализовано:

- регистрация пользователя;
- настройку и добавление уведомлений;
- добавление привычек с выбором типа привычки;
- редактирование привычки (изменение названия, описания и времени);
- просмотр статистики.

Обзор аналогов

Loop



Достоинства:

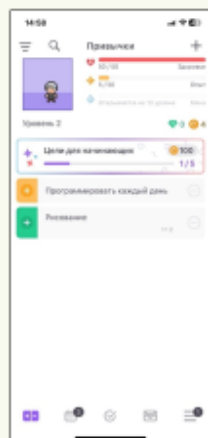
- отслеживание привычек;
- статистика;
- оповещения;
- удобство.

Недостатки:

- нет русской локализации;
- нет установки времени для выполнения задания.

Обзор аналогов

Habitica



Достоинства:

- отслеживание привычек выполнено в виде заданий;
- пользователи могут объединяться и работать над общими целями;
- игровой подход.

Недостатки:

- не подходит тем, кто не любит игровой подход;
- нет стандартных функций трекера;
- перегруженное приложение.

Стек технологий



Firestore

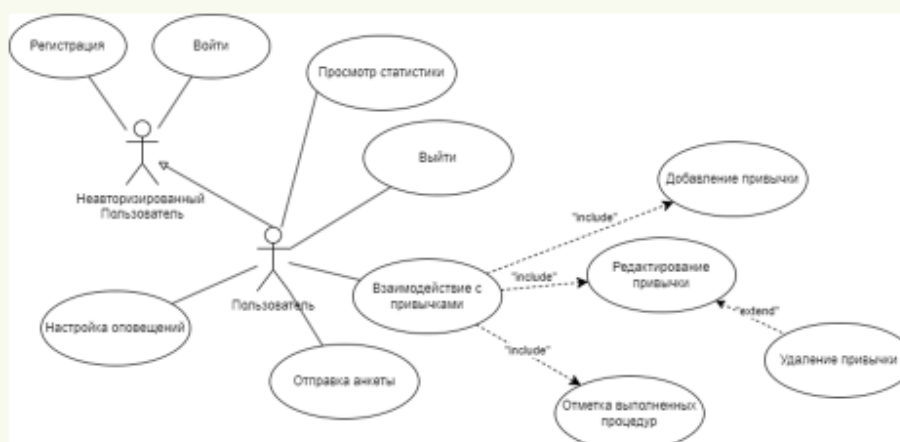
Изм.	Лист	№ докум.	Подпись	Дата

ПГУ1.090301.21.001 ПЗ

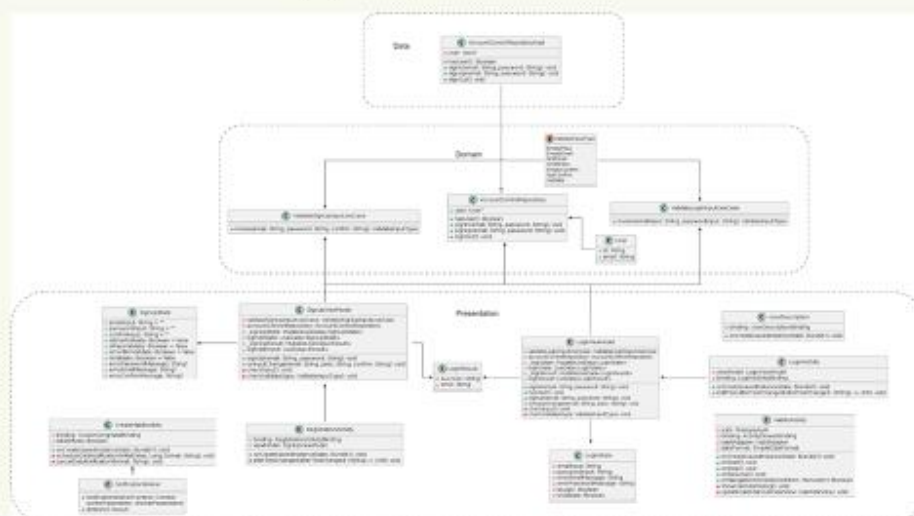
Лист

114

UML-диаграмма прецедентов



UML-диаграмма классов



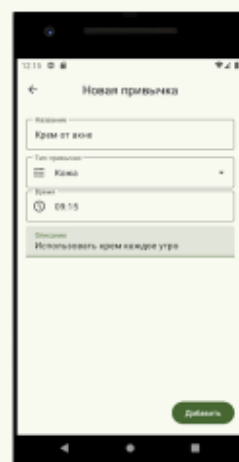
Приложение «Привычки красоты»



Регистрация пользователя



Просмотр списка привычек



Добавление новых привычек

РЫНОК

Целевая аудитория:

- фитнес-центры. Могут использовать приложение для улучшения удержания клиентов;
- фармацевтические компании. Могут интегрировать приложение для напоминания о приеме лекарственных средств;
- косметические компании. Могут использовать приложение для продвижения рутинного ухода за телом.

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ1.090301.21.001 ПЗ

Лист

116

Бизнес модель

Элементы бизнес-модели:

Ключевые ресурсы:

- Технологическая платформа
- Команда разработчиков
- Маркетологи

Ключевые действия:

- Разработка и обновление ПО
- Маркетинг и продажи
- Техническая поддержка

Модели дохода:

- Покупка приложения: компании платят за приложение
- Премиум-подписка для пользователей
- Партнерские программы
- Реклама

Риски

Для того чтобы успешно реализовать стартап-проект требуется провести анализ всех рисков, которые могут негативно повлиять на его исход.

Возможными рисками являются, например, недостаточный спрос на предлагаемое мобильное приложение, проблемы с финансированием проекта, технические трудности при внедрении проекта и нестабильность экономической ситуации.

Важно проанализировать и предусмотреть меры по решению или минимизированию негативных последствий рисков.

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ1.090301.21.001 ПЗ

Лист

117

Заключение

Результатом выполнения данной выпускной квалификационной работы можно считать успешно разработанное мобильное приложение, полностью соответствующее заданию.

В результате было создано приложение для мобильных устройств написанное на языке программирования Kotlin. Также в ходе разработки были получены навыки работы с Firebase. Результаты тестирования показали, что приложение полностью соответствует всем поставленным задачам, имеет удобный пользовательский интерфейс и выполняет свою работу корректно.

Спасибо за внимание!

					ПГУ1.090301.21.001 ПЗ	Лист
						119
Изм.	Лист	№ докум.	Подпись	Дата		

МИНОБРНАУКИ РОССИИ
ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Кафедра "Вычислительная техника"

УТВЕРЖДАЮ

Заведующий кафедрой

Митрохин М.А.

(Подпись)

(Фамилия, инициалы)

«___» _____ 20__ г.

ЗАДАНИЕ НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ В ФОРМЕ СТАРТАПА

Студент Папшев Никита Сергеевич

Направление подготовки 09.03.01 Информатика и вычислительная техника.

Профиль Программное обеспечение средств вычислительной техники и автоматизированных систем

Форма обучения Очная

1. Тема: Стартап-проект. «Приложение-трекер «Привычки красоты» для ОС Android», которое будет выполняться данным членом команды.

Тема одобрена экспертной комиссией и утверждена приказом ректора от «19» апреля 2024 г. № 40570

2. Срок представления работы к защите «___» _____ 20__ г.

3. Общая часть ВКР:

Подготовка компьютерной презентации стартапа.

4. Индивидуальная часть ВКР:

4.1. Исходные данные:

Разработать мобильное приложение «Привычки красоты», позволяющее отслеживать прогресс пользователя по выработке привычек. Составить бизнес-модель и финансовый план. Язык программирования Kotlin, среда разработки Android Studio. Программа должна обладать интуитивно понятным интерфейсом и использовать следующие технологии:

1. Firebase authentication

2. Cloud Firebase

3. Асинхронное программирование

4. Material design 3

4.2. Содержание пояснительной записки ВКР (по разделам):

Анализ предметной области, обоснование актуальности разработки, обзор аналогов, анализ требований к разработке, постановка задачи, выбор инструментов разработки, архитектура приложения, проектирование и разработка, тестирование и отладка, БЖД, бизнес-план: финансовые и временные ограничения на проект (бюджет, сроки реализации, ожидаемые затраты и доходы и т.д.).

5. Перечень графического материала: UML-диаграммы (диаграмма классов, диаграмма развертывания, диаграмма деятельности, диаграмма вариантов использования, диаграмма последовательности)

Руководитель ВКР

Задание принял к исполнению

ф.и.о.
(подпись, дата)

Папшев Н.С.
(Ф.И.О.)

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ1.090301.21.001 ПЗ

Лист

120

Экспертное заключение
о согласовании темы ВКР в форме стартапа

Экспертная комиссия в составе декан ФВТ д.т.н., профессор Л.Р. Фионова, доцент кафедры ВТ, к.э.н., И.В. Акифьев, ведущий системный архитектор ООО «Ай-теко ВЦ», к.т.н. М.П. Синев, рассмотрев бизнес-проект Стартап-проект. Приложение-трекер «Привычки красоты» для ОС Android, Папшева Никиты Сергеевича подтверждает его актуальность, практическую значимость и возможность коммерциализации.

Решение экспертной комиссии: выпускная квалификационная работа на тему Стартап-проект. Приложение-трекер «Привычки красоты» для ОС Android, Папшева Никиты Сергеевича может выполняться в форме стартапа (бизнес-проекта).

Председатель ЭК

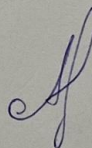
д.т.н., профессор
декан ФВТ



Л.Р. Фионова

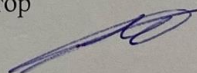
Члены ЭК

к.э.н., доцент кафедры ВТ



И.В. Акифьев

к.т.н., ведущий системный архитектор
ООО «Ай-теко ВЦ»



М.П. Синев

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ1.090301.21.001 ПЗ

Лист

121