МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное
учреждение высшего образования
«Южно-Уральский государственный университет»
(национальный исследовательский университет)
Высшая школа экономики и управления
Кафедра «Информационные технологии в экономике»

ДОПУСТИТЬ К ЗАЩИТЕ
Зав. кафедрой,д.т.н., с.н.с.
_____/ Б.М. Суховилов /
«_____»_____20____ г.

Разработка мобильного приложения «Tavria_app»

на базе операционной системы Android

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
ЮУрГУ – 09.03.03.2021.255.ВКР

Консультант, доцент, к.т.н.
_____/ В.А. Конов /
«_____» _____2021 г.

Руководитель, доцент, к.т.н.
_____/ В.А. Конов /
«_____»_____2021 г.

Автор
студент группы ЭУз – 582.
_____/ В.Д. Рябов /
«_____» _____2021 г.

Нормоконтролер, ст.преподаватель
_____/Е.Н. Горных/
«_____»_____2021 г.

Челябинск 2021

АННОТАЦИЯ

Рябов В.Д. Разработка мобильного приложения на базе ОС Android. – Челябинск: ЮУрГУ, ЭУз-582, 2021. – 77 с., 9 табл., 32 рис., библиографический список – 13наим.

Объектом исследования являются бизнес-процессы, а именно их оптимизация средствами программы AndroidStudio, путем создания нового программного продукта. Цель работы – разработка и создание конечного программного продукта для внутренних нужд компании МПК «Таврия».

В первой главе работы дана характеристика общества с ограниченной ответственностью Мясоперерабатывающий Комбинат «Таврия» (далее МПК«Таврия»). Обозначены проблемы существующих на рынке решений. Сделан вывод о необходимости разработки новой системы удалённой подачи заявок.

Во второй главе проведен сбор требований к приложению, обоснован выбор средства реализации ПП. Разработано мобильное приложение для МПК «Таврия».

В третьей главе работы определены затраты на разработку мобильного приложения МПК «Таврия». Определены доходы от внедрения в работу приложения. Произведен расчет экономической эффективности программного продукта.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Актуальность темы выпускной квалификационной работы: на сегодняшний день на рынке программных продуктов, множество комплексных решений для организации работы мобильной торговли. Однако функционал многих программных продуктов ограничен в ряду аспектов применения или ряду областей ввиду особенностей разработки и специфики рынка. В связи с этим целесообразно создание нового мобильного приложения.

Операционная система Android (далее – ОС Android)— является одной из наиболее распространенных, простых и одновременно комплексных платформ.

В рамках дипломной работы описано создание приложения «Мобильная торговля» для торговых представителей (далее – ТП) на базе ОС Android, ориентированной на 5 конкретных заводов. Преимущества ОС Android обусловлены массовостью и широким распространением современных устройств на базе Android, оснащенных различными функциями и приложениями, делающими работу ТП удобной. Ориентированность на мобильные приложения объясняется тем, что в подавляющем большинстве ТП являются обладателями и пользователями современных смартфонов, планшетов и всегда находятся в пути.

Объектом выпускной квалификационной работы является представление деятельности МПК «Таврия» в мобильной среде.

Предметом квалификационной работы является мобильное приложение для МПК «Таврия».

Целью дипломной работы является создание программного продукта на базе ОС Android для использования ТП в качестве системы выполнения заказа продукции для клиентов МПК «Таврия». Для этого в рамках дипломной работы должны быть решены следующие задачи:

- разработка графического пользовательского интерфейса;
- проанализировать аналогичные мобильные приложения;
- создание базы данных заказов (временная для демонстрации);

- реализация программного модуля для отчетности перед клиентами;
- внедрение доп. возможностей для слабовидящих;
- реализация системы авторизации (для разделения баз данных);
- добавление настроек для разных экранов и локализация.

Источниками информации ВКР являются: техническая литература по разработке и отладке программного обеспечения.

Объем ВКР составляет 61 страницы основной части и 15 страниц приложения А. Таблиц в работе 9, иллюстраций в работе 32.

В первой главе работы дана характеристика общества с ограниченной ответственностью МПК«Таврия». Поставлена задача по внедрению нового ПО. Обозначены проблемы существующих на рынке решений, что обосновывает необходимость разработки новой системы удалённой подачи заявок.

Во второй главе рассмотрена поставленная задача, дана характеристика программного продукта и рассмотрены выбранные инструменты реализации поставленной задачи, спроектировано и разработано мобильное приложение.

В третьей главе работы определены затраты на разработку мобильного приложения для МПК «Таврия». Определены доходы от внедрения в работу приложения. Произведен расчет экономической эффективности программного продукта. Подтверждена целесообразность разработки и внедрения приложения.

1. ХАРАКТЕРИСТИКА ООО МПК «ТАВРИЯ»

1.1 Описание предприятия

В состав МПК «Таврия» входит шесть крупных мясокомбинатов, расположенных в Челябинске, Екатеринбурге, Нижнем Новгороде, Калининграде, Тольятти и Волгограде. Одним из первых мясокомбинатов группы компаний, а также объектом исследования является Челябинский мясокомбинат, производящего продукцию под торговой маркой «Таврия». Компания занимается производством и продажей колбасной продукции, мясных деликатесов и полуфабрикатов.

Компания «Таврия» существует на рынке с 2008 года и стремительно развивается с момента своего основания. На сегодняшний день Компания входит в ТОП успешных компаний УРФО, имеет производство площадью 7000 $м^2$ и производственной мощностью до 3000 тонн в месяц, укомплектовано самым современным европейским оборудованием: «SCHALLERTECHNOLOGY» (Германия),«LASKA» (Австрия), «Handtmann» (Германия), Poly-clip (Германия), Vemag (Германия). Складские помещения оборудованы новейшей системой охлаждения и хранения и составляют 2000 $м^2$.За 8 лет работы объемы продаж компании выросли с 40 тонн до 1300-1500 тонн продукции в месяц.

Продукция компании пользуется широким спросом у населения за счет высокого качества выпускаемой продукции и среднего ценового сегмента. Потребителями продукции компании являются как население, так и комбинаты школьного питания.

Компания «Таврия» выпускает широкий ассортимент продукции, который насчитывает в своем составе более 250 наименований колбас, мясных деликатесов и полуфабрикатов.

Структура предприятия отображена на рисунке 1.

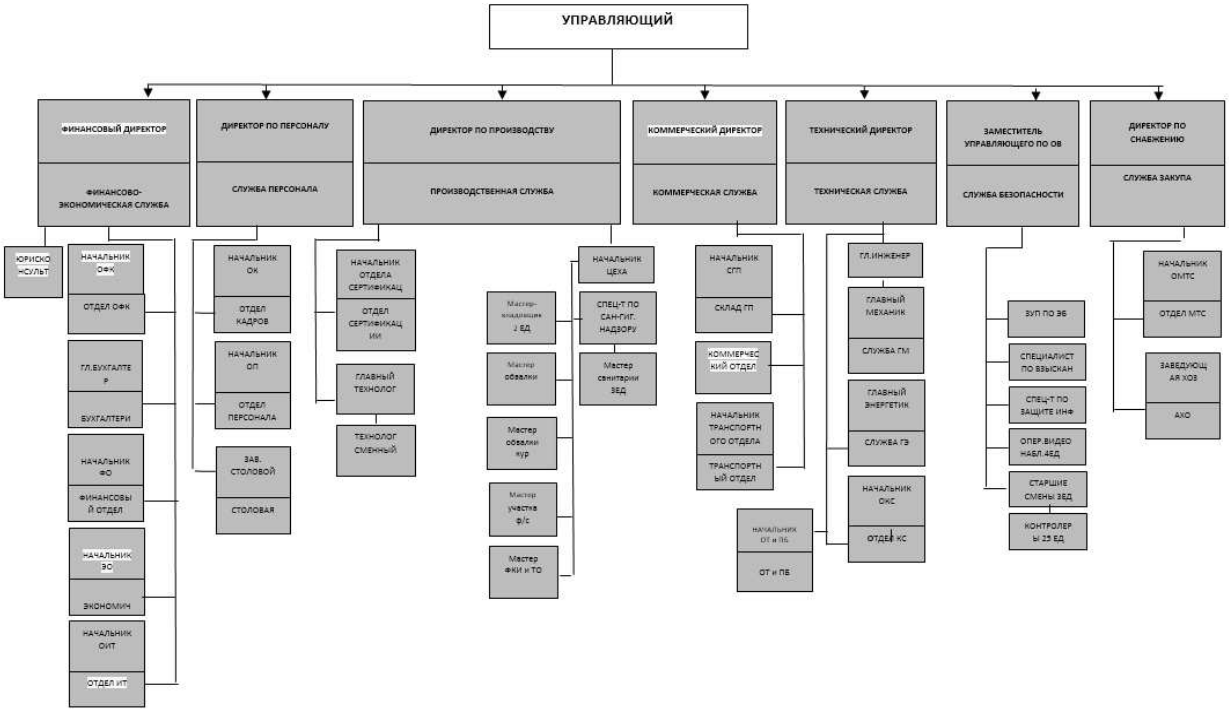


Рисунок 1 – Структурная схема предприятия

Руководством МПК «Таврия» принято решение о создании собственного или приобретении прав на использование мобильного приложения на платформах Android для повышения имиджа, расширения рынка сбыта и для простоты доступа к информации. Отделу информационных технологий (далее IT-отдел) представлено техническое задание на поиск или разработку мобильного приложения на базе операционной системы Android, состоящее из следующих пунктов:

- Приложение должно включать в себя информацию о заказах, продукции и клиентах;
- Приложение должно быть реализовано на русском и английском языке;
- Ориентация экрана должна меняться при изменении угла наклона;

- Запуск приложения должен начинаться с авторизации пользователя;
- Приложение должно работать только на операционной системе Android.

Согласно перечню предъявляемых требований, необходимо провести анализ рынка программных продуктов для поиска готового программного продукта, по итогам которого будет принято решение о приобретении готового программного продукта либо разработке собственного программного продукта.

1.2 Анализ готовых решений

На сегодняшний день на рынке программных продуктов, множество комплексных решений для организации мобильной торговли. Однако функционал многих программных продуктов ограничен из-за специфики рынка.

Цель: анализ существующих программных решений, представленных на рынке для определения целесообразности разработки программного комплекса для осуществления экономической деятельности в рамках предприятия.

Программный продукт должен включать в себя: возможность отправки заявок и актуализации данных, простоту интеграции в уже существующую информационную структуру, низкие затраты на разработку, введение в эксплуатацию и обслуживание, высокие эксплуатационные характеристики, возможность простого расширения функциональных возможностей.

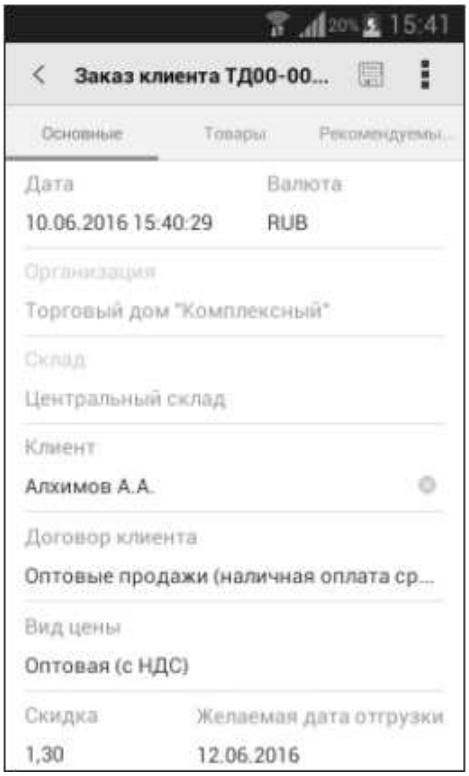
Для проведения детального анализа рассмотрим три популярных программных продукта: 1С: Мобильная торговля, АСМТ «Наполеон», ST-мобильная торговля. Рассматриваемые программные продукты служат для решения схожих производственных задач. В своей работе отразим функциональные возможности, преимущества и недостатки, ценовую политику данных программных продуктов.

1.2.1 1С:Мобильная торговля

Продукт «1С:Мобильная торговля»[1] предназначается для автоматизации мобильной торговли с помощью мобильных устройств (смартфонов, планшетных компьютеров) под управлением операционной системы Android, показан на рисунке 2. Продукт позволяет автоматизировать различные виды деятельности торговых представителей оптовых и дистрибьюторских компаний, использующих учетные системы на платформе «1С: Предприятие 8», например: «1С:Управление торговлей» или «1С:ERP Управление предприятием».

Лицензия на 1 устройство стоит 6000 руб.за 1 год использования. При условии, что на предприятии работает более 500 ТП по всей России, то ценник выходит примерно в 3000000 руб. в год.

Используя продукт "1С: Мобильная торговля", торговый представитель получает инструмент, который помогает ему выполнять свою работу более эффективно – с минимальными затратами времени и с меньшим количеством промежуточных операций.



[1] «Фирма 1С»: [сайт] URL: https://solutions.1c.ru/catalog/mobile_trade

Рисунок 2– 1С: Мобильная торговля. Заказ покупателя.

К преимуществам данного программного продукта можно отнести:

- Дружественный интерфейс;
- Простоту настройки;

Недостатками программного продукта является:

- В приложении существуют проблемы с выгрузкой базы данных;
- Высокая стоимость приобретения ПО и лицензий.

1.2.2 АСМТ «Наполеон»

Мобильная торговля "Наполеон"[2] отлично подходит производителям и дистрибьюторам для автоматизации работы торговых представителей, показан на рисунке 3. Во-первых, уже в базовом решении всегда есть: автоматизация сбора предварительных заказов агентами и продажи с борта автомобиля, презентация товара, взаиморасчеты, мерчандайзинг, анкетирование клиентов. Во-вторых, полный и бесплатный GPS контроль визитов и перемещений торговых представителей. В-третьих, аудит работы торгового отдела (совместный и догоняющий), аналитика и интеграция с другими системами. Стоимость лицензии на 1 устройство 2000руб.

[2] АСМТ "Наполеон" : [сайт] URL: https://grsoft.ru/

Рисунок 3–АСМТ «Наполеон». Главное меню

К преимуществам программного продукта можно отнести:

- Дружественный интерфейс;
- Простота настройки;
- Низкая стоимость;
- Возможность заказа доп. расширений.

Недостатками программного продукта является:

- Низкий уровень качества поддержки приложения.

1.2.3 ST-мобильная торговля

ST-мобильная торговля[3], показан на рисунке 4. Менеджер (супервайзер) планирует территории и маршруты, проверяет документы и оперативную отчётность, ставит задачи и полноценно контролирует удаленных подчиненных.Система отслеживает GPS-координаты сотрудников, время

[3] ST-мобильная торговля: [сайт] URL: https://sys4tec.com/products/st-mobile/

нахождения в каждой точке, выполнение задач, действия и перемещения на маршруте. Сформированный из этих данных ежедневный отчет автоматически отправляется руководителю. Фотоотчеты помогают контролировать полку и оформление мест продаж.«ST-мобильная торговля» хранит историю продаж, индивидуальные прайсы и скидки, отвечает за формирование и печать документов, рассчитывает зависимость цены от типа оплаты, подсказывает условия трейд-маркетинговых акций.

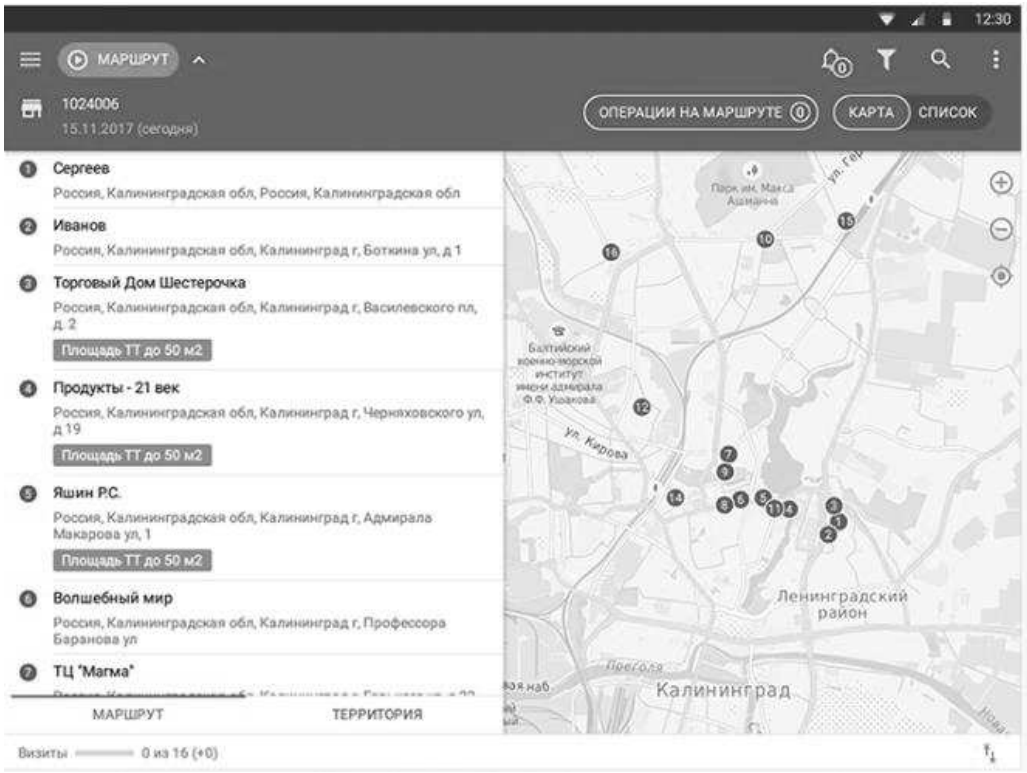


Рисунок 4 –ST-мобильная торговля

К преимуществам программного продукта можно отнести:

- Управление командой выездных сотрудников;
- Автоматизация работы супервайзера в местах продаж;
- Управление территориями и маршрутами агентов;
- Контроль и анализ работы торговой команды

Недостатками программного продукта является:

- Зависимость ТП от установки маршрутов

Таким образом, проведя детальный анализ выше описанных программных продуктов, был выявлен ряд недостатков, в числе которых высокая стоимость, низкий уровень качества поддержки и реализации. Выявленные недостатки являются существенными и не удовлетворяют предъявляемым требованиям, что в свою очередь рождает необходимость разработки собственного мобильного приложения.

Выводы по первой главе

В первой главе выпускной квалификационной работы дана характеристика информационной структуры МПК «Таврия». Кратко описали общую структуру предприятия и структуру производства продукции.

Выявлены преимущества мобильного приложения после анализа предметной области. Принято решение использовать среду Android Studio и язык программирования Kotlin после анализа похожих проектов и популярных сред для разработки.

В рассмотренных программных продуктах выявлен ряд недостатков, в числе которых высокая стоимость, низкий уровень качества поддержки и реализации. Выявленные недостатки являются существенными и не удовлетворяют предъявляемым требованиям, что в свою очередь рождает необходимость разработки собственного мобильного приложения.

2. РАЗРАБОТКА МОБИЛЬНОГО ПРИЛОЖЕНИЯ

2.1 Проектирование архитектуры приложения для МПК «Таврия»

Создано приложение для торговли. «Tavria_app» - это сложное приложение, для создания которого потребовалось много слоёв. На главном экране отображается список заказов, и пользователи смогут добавлять новые заказы или выбирать существующие, чтобы просмотреть и отредактировать их.

Для обоснования необходимости использования мобильного приложения на платформе Android следует выделить и проанализировать процессы, нуждающиеся в облегчении работы с заявками клиентов. На основании собранных требований предъявляемых к реализации задачи по построению информационной системы в рамках конкретного предприятия МПК «Таврия» подготовлен ряд диаграмм по стандартам UML, DFD и IDEF3.Данные материалы служат для облегчения процесса создания программного продукта, а так же корректного составления технического задания. Концептуально-функциональная модель процесса деятельности «Подача заявки» продемонстрирована на рисунке 5.

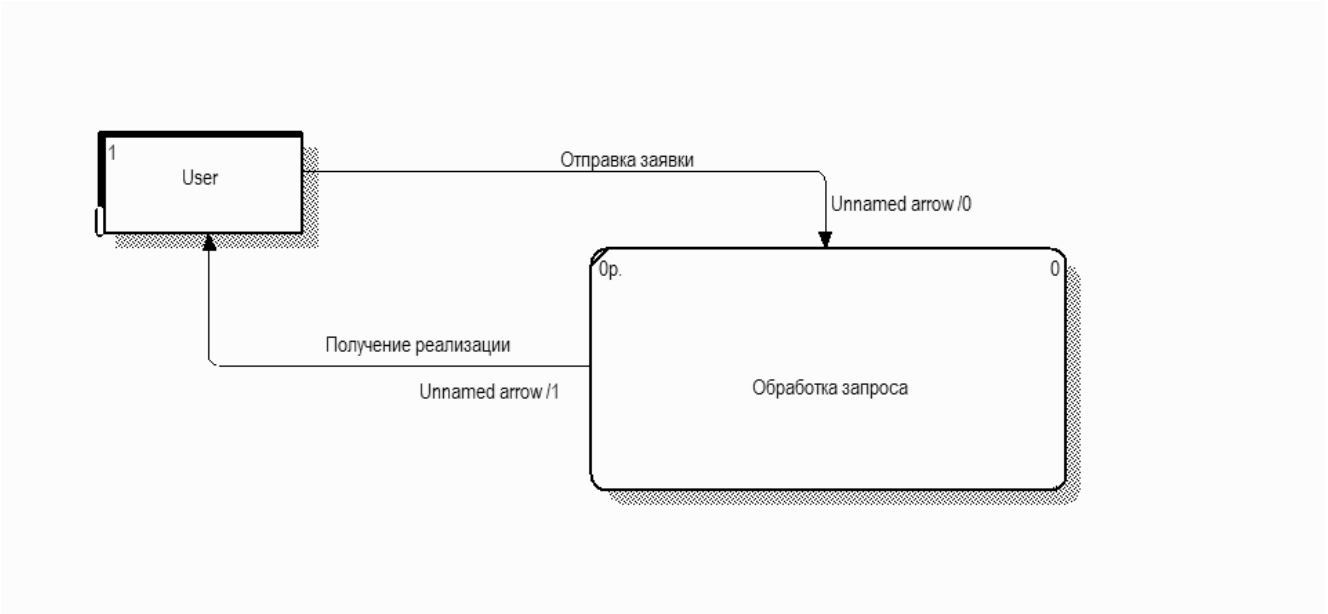


Рисунок 5 – Верхний уровень DFD диаграммы

На диаграмме показана бизнес-модель выполнения заказа в организации МПК «Таврия» в общем виде. Процесс протекает следующим образом: клиент связывается по удобному для него каналу связи (телефон, электронная почта) или лично посещает и делает заказ, где ТП данной организации должен понять и, если требуется, посоветовать клиенту, как лучше осуществить его пожелания. После того, как все согласовано в области заказа клиента, берется со склада нужное количество товара. Следующим этапом является процесс сборки, а за ним – отправки данной продукции. Готовый продукт упаковывается, после чего осуществляется оплата и доставка заказчику.

Следующая диаграмма показывает декомпозицию главного процесса, она продемонстрирована на рисунке 6. Каждый блок взаимосвязан и, также все последующие блоки начинаются после завершения предыдущих блоков.

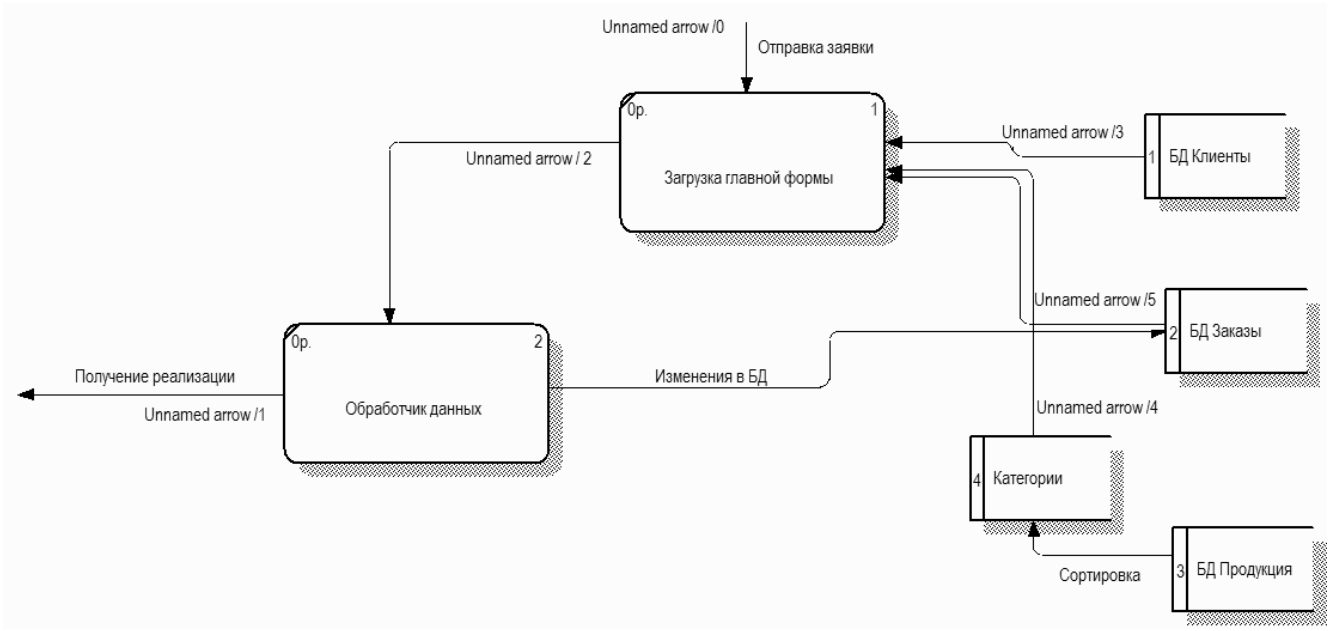


Рисунок 6–Второй уровень DFD диаграммы

На рисунке видно, что происходят три основных процесса:

- подача заявки от клиента;
- сборка заказа клиента и формирование отгрузки;
- формирование реализации.

Процесс «Подача заявки от клиента», декомпозиция которого показана на рисунке 7, связан с подачей клиента заявки на нужный ему товар, с его необходимым оформлением и утверждением.

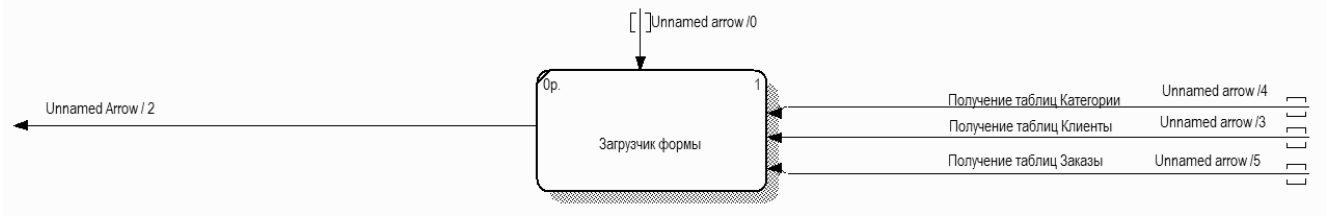


Рисунок 7–Третий уровень DFD диаграммы

Для наиболее полного понимания архитектуры решаемой задачи подготовим диаграмму по стандарту IDEF3, позволяющую подробнее понять причинно-следственные связи между ситуациями и событиями в процессе работы

разрабатываемой информационной системы. Диаграмма по стандарту IDEF3 показана на рисунке 8.

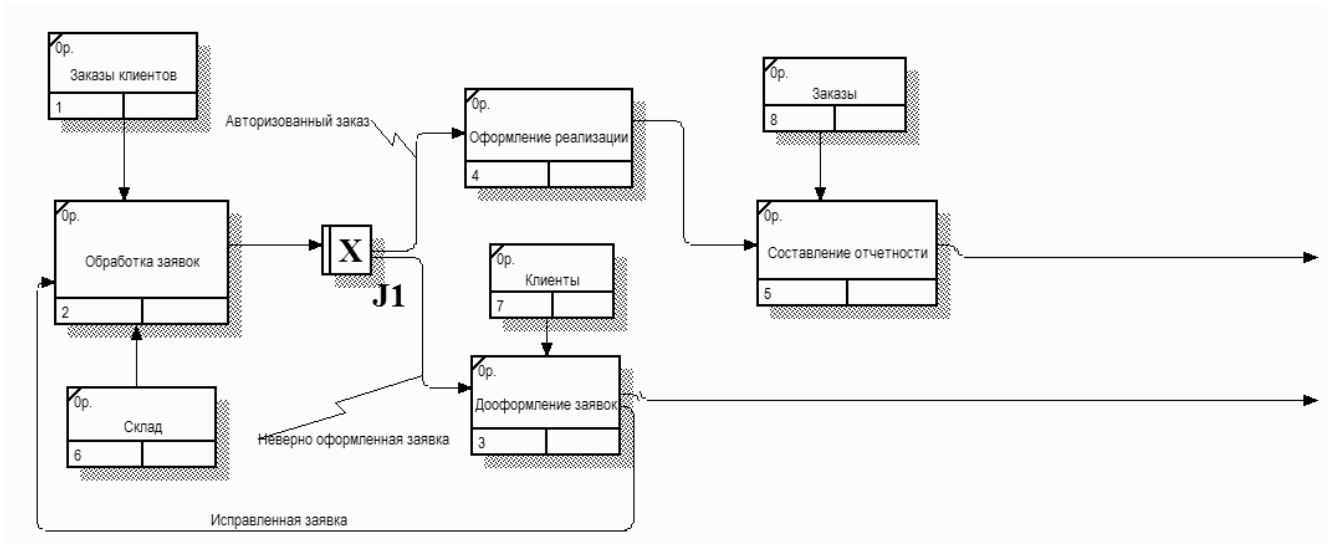


Рисунок 8–Диаграмма IDEF3

Исходя из описания бизнес-процессов организации МПК «Таврия», можно увидеть, что персоналу приходится тратить достаточно много рабочего времени на обработку так называемых «мелких» заказов.

Диаграмма компонентов- элемент языка моделирования UML, статическая структурная диаграмма, которая показывает разбиение программной системы на структурные компоненты и связи между компонентами.

Для определения структуры и обозначения данных, которые необходимы для приложения. В ходе анализа решено, что в мобильное приложение будут внедрены в качестве исходных данных следующие разделы:

- Список покупателей.
- Раздел заказа продукции.
- Раздел авторизации пользователя.
- Добавление фото материалов.
- Отчетность.
- Доп. функции для слабовидящих.

- Локализация и расширение экрана.

В качестве физических компонентов, показанных на рисунке 9, могут выступать файлы, библиотеки, модули, исполняемые файлы.

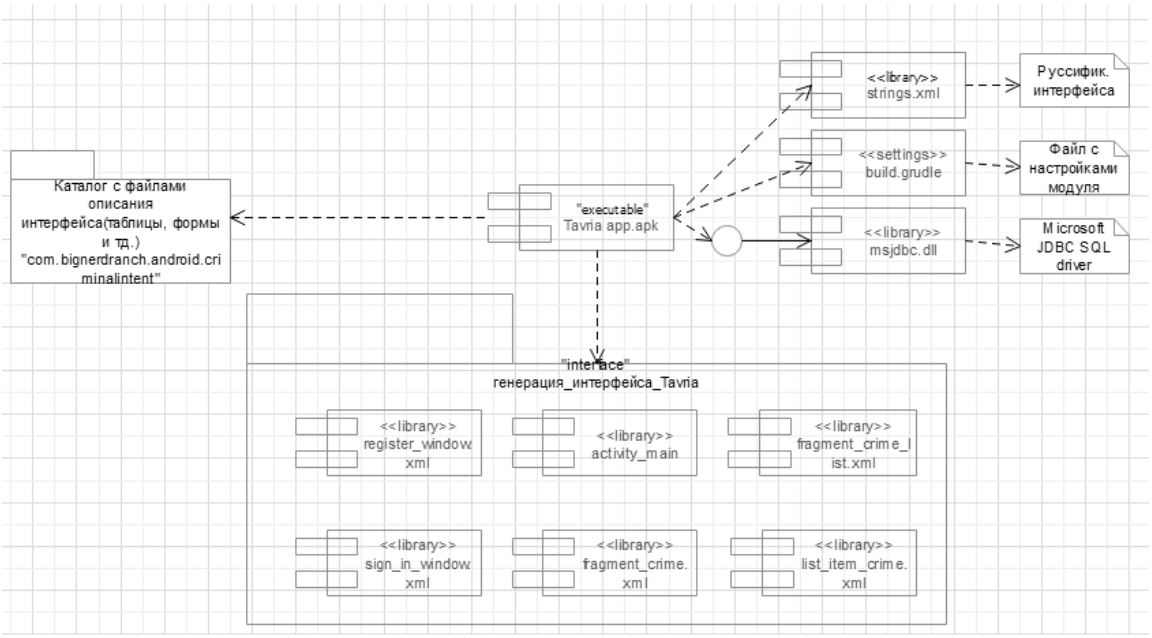


Рисунок 9 – Диаграмма компонентов

Диаграмма последовательности - UML-диаграмма, показанная на рисунке 10, на которой для некоторого набора объектов на единой временной оси показан жизненный цикл объекта и взаимодействие актеров информационной системы в рамках прецедента.

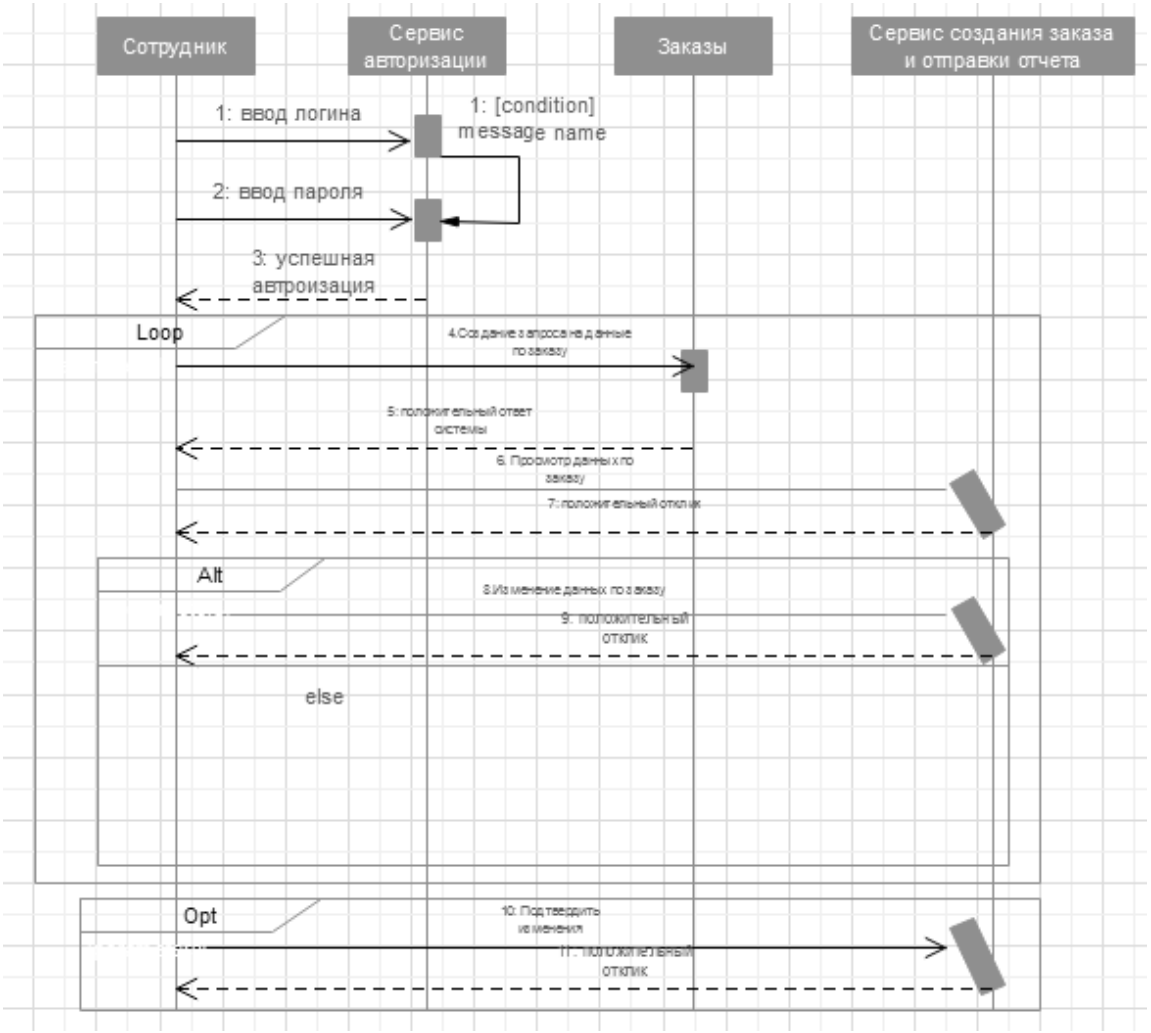


Рисунок 10–Диаграмма последовательностей

Опишем систему подачи заявки от потребителя с точки зрения диаграммы вариантов использования. Благодаря диаграмме вариантов использования появляется возможность обозначить внешние системы, взаимодействующие с системой, главные процессы и их взаимосвязанность. Диаграммы вариантов использования предоставляют возможность обозначить функциональную структуру системы, не вникая в детали ее реализации. А также происходит предварительное выявление и классификация системных объектов. В результате сформированной модели создается план разработки системы.

Указано, какие возможности обязана иметь разрабатываемая система: пользователь применяет систему для осуществления регистрации и подачи заявки

на приобретение товаров, системный администратор выполняет контроль поведения системы, наделяет описанием товары, редактирует данные. Автоматизированная информационная система осуществляет хранение данных и предоставляет к ним доступ посредством сети Интернет с возможностью поиска необходимой информации.

Названные выше пункты позволяют обозначить следующие прецеденты, которые реализованы в системе показаны в таблице 1.

Таблица 1 - Краткое описание прецедентов

Прецеденты	Краткое описание
1	2
Регистрация	Регистрация пользователя в мобильном приложении
Добавление товаров	Добавление товаров с описанием и их количеством
Подача заявки на приобретение товаров	Формирование заявки и ее отправление исполнителю
Получение результата после отправки заявки	Предоставление пользователю информации о результате выполненного заказа
Редактирование данных	Редактирование информации о продукции

Разработанная диаграмма вариантов использования для основных прецедентов проектируемой системы представлена на рисунке 11, где актеры, изображенные на ней – это группа лиц или другие системы, взаимодействующие с данной системой, а прецеденты (варианты использования) – это сервисы, обеспечиваемые системой. Сплошными линиями на диаграмме использования являются отношения ассоциаций, олицетворяющие способность применения

актером прецедента. На диаграмме использования можно заметить границы системы.

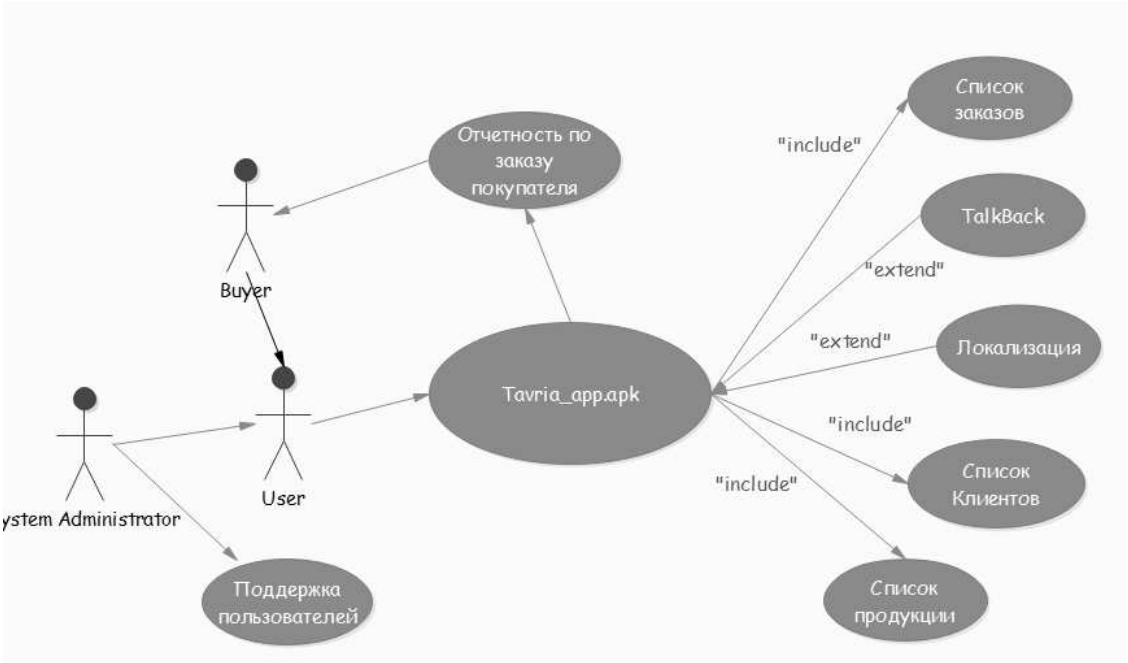


Рисунок 11 – Диаграмма прецендентов

Создана диаграмма классов, показанная на рисунке 12, олицетворяющая классы, их операции и атрибуты, связи между ними. На диаграмме представлены основные классы: User, Category, Product, Order, и интерфейс Mobile App.

Класс **User** имеет возможность просматривать, выбирать товары и производить заказы через Mobile App, данный класс хранит в себе данные такие как: имя, адрес, город, регион, номер телефона, эл. почта, пароль для авторизации и список выбранных им товаров.

Класс **Category** содержит виды продукции организации, а именно их наименование и изображение категории.

Класс **Product** содержит данные о товаре, такие как: наименование, категорию, цену товара, наличие товара на складе, его количество и его описание.

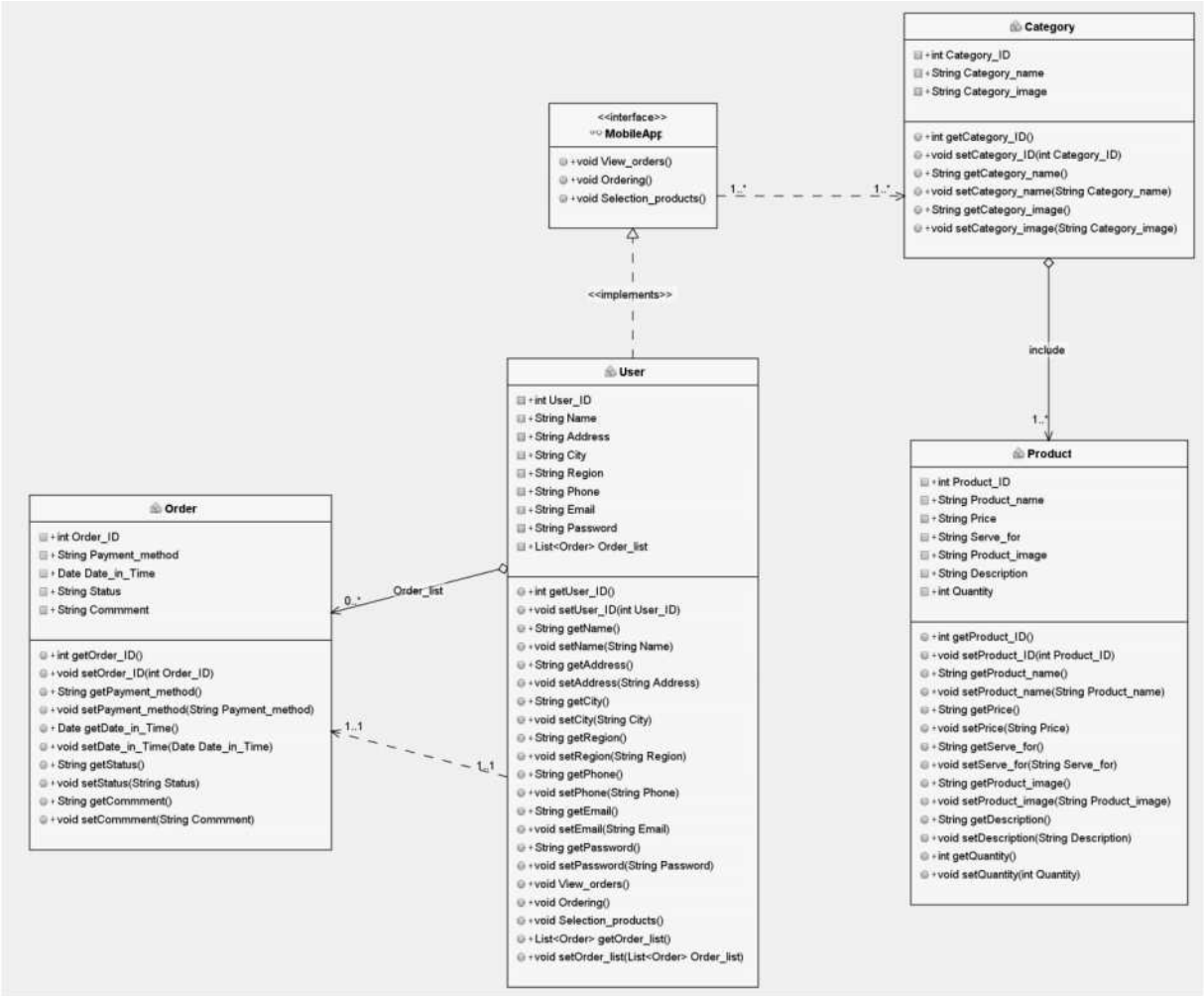


Рисунок 12 – Диаграмма классов

Mobile App предоставляет возможность пользователю просматривать товары, выбирать необходимую ему продукцию в нужном ему количестве, а также оформлять заказ.

Order хранит в себе данные о заказе сделанным пользователем например: способ оплаты, дату доставки, статус заказа и комментарий.

Наполнение базы данных данными производиться через 1С: Предприятие. Обслуживание базы данных сводиться к своевременному выполнению резервного копирования для минимизации последствий возможных сбоев.. На рисунке 13 изображена диаграмма базы данных, построенная на основе диаграммы классов.

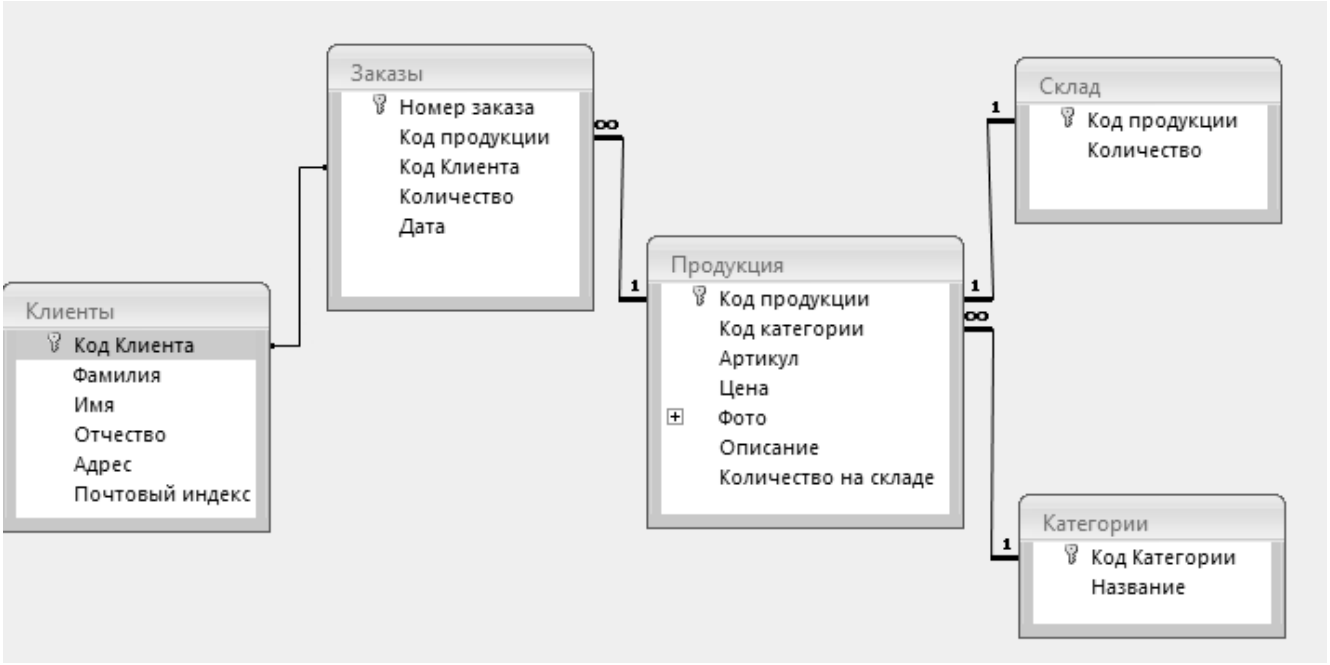


Рисунок 13 – Схема базы данных

Информационная модель отображает информацию о системных объектах в виде логической структуры. На диаграмме проиллюстрированы следующие сущности: Заказы, Клиенты, Склад, Категория и Продукция со связями различной кратности. Один товар может принадлежать только к одной категории товаров, что показано связью типа «один-к-одному» между сущностями Категория и Товар, но одна категория может содержать несколько товаров, что отображено связью вида «один-ко-многим».

2.2 Выбранная среда верстки приложения

Средой разработки мобильного приложения выбрана Android Studio, показанная на рисунке 14. Он состоит из следующих компонентов: Android SDK, компоненты графического дизайна, приложение для загрузки компонентов всех версий Android, эмулятор мобильного устройства для запуска приложения.

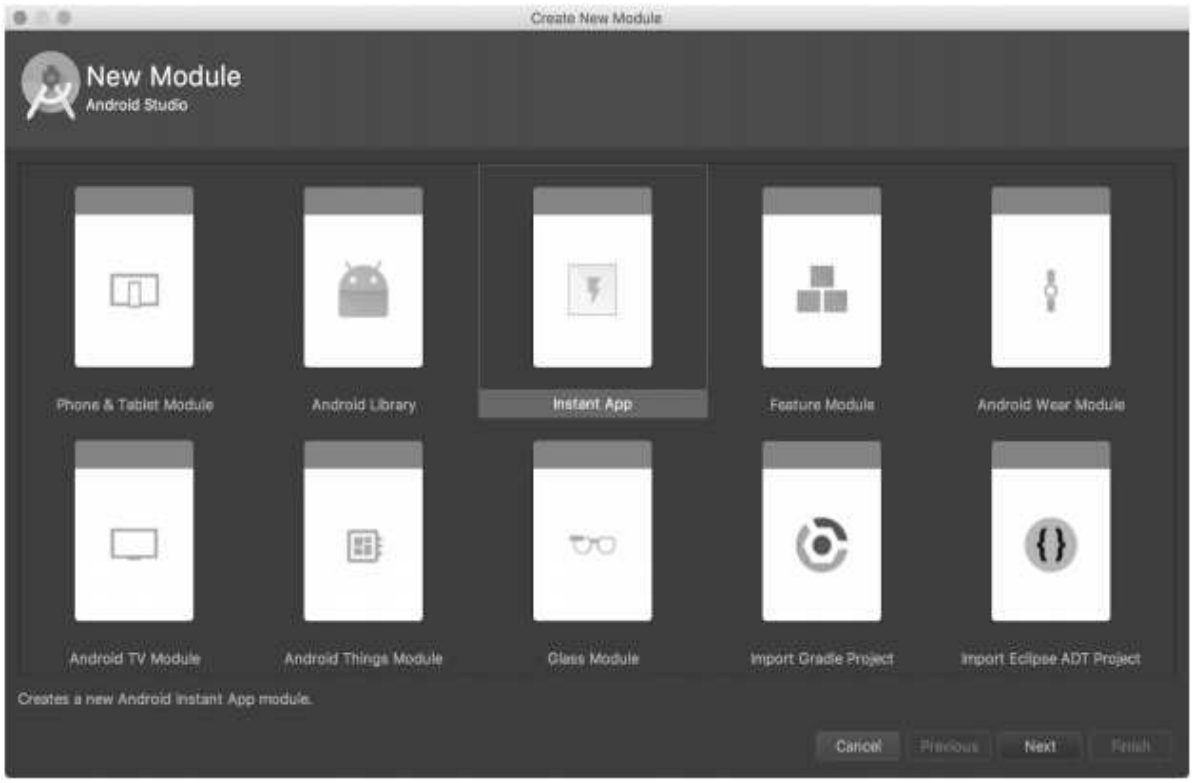


Рисунок 14 – Создание нового проекта в Android Studio

Важным моментом при выборе Android Studio стало то, что она имеет возможность разрабатывать программы для всех версий операционной системы Android. Многоцветный код упрощает навигацию в больших объемах кода. Программа включает в себя Google Cloud Messaging, целью которой является настройка отправлений уведомлений для приложений, при котором задействуются облачные сервисы под управлением ОС Android. Таким образом, можно создавать многозадачные программы.

Программное обеспечение призвано предоставить пользователям возможность для совершения процедур купли-продажи между ТП и клиентами МПК «Таврия».

2.3 Физическое моделирование автоматизированной системы

Современные мобильные приложения реализуются в архитектуре «клиент-сервер», показана на рисунке 15.

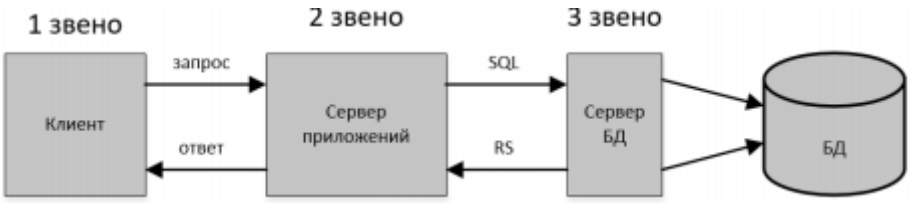


Рисунок 15 – Трехзвенная клиент-серверная архитектура

Первое звено – мобильное приложение на платформе Android – «тонкий клиент». Второе звено – сервер приложения – программный продукт переходного уровня. Синхронизирует работу компонентов системы и создает их взаимосвязь. В основном, данную задачу выполняют программные обеспечения и сетевые службы. Третье звено – сервер баз данных - не взаимодействует с клиентом напрямую, вследствие чего повышается безопасность системы.

Плюсы:

- Возможность масштабируемости;
- Простота конфигурации.

Минусы:

- Сложность производства программного продукта;
- Относительно высокие требования к продуктивности сервера базы данных и серверов приложений.

Итоговый вариант реализации представлен на диаграмме реализации. Показан на рисунке 16.

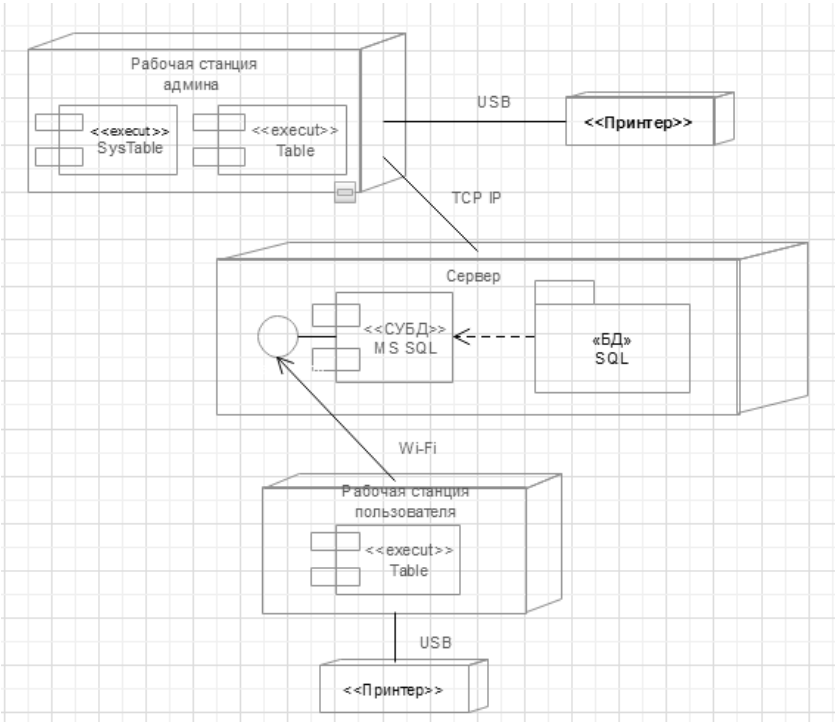


Рисунок 16 – Диаграмма реализации

2.4 Реализация функциональных требований

Создана стартовая страница приложения. По умолчанию всегда идет сразу функционал программы, но этот экран будет разделять базы при помощи авторизации. Поэтому стартовым экраном (слоем) стала авторизация. Всё начинается с манифеста. AndroidManifest.xml - это необходимый файл в любом проекте. Он определяет глобальные значения для пакета, в нем описывается, что находится внутри приложения: деятельности, сервисы и т.д. Так же определяется, как все эти элементы взаимодействуют с Андроид. Полный листинг манифеста находится в приложении А.

Реализация стартового экрана. показана на рисунке 17.

Рисунок 17 – Стартовый экран activity_main

На слое авторизации 2 кнопки. Регистрация (Register) и Вход (Enter).Обе кнопки это ссылки на следующие слои. Полный код написан в приложении А.

При нажатии кнопки Регистрация, происходит переход на следующий слой программы, а именно на слой регистрации нового пользователя. Он показан на рисунке 18.

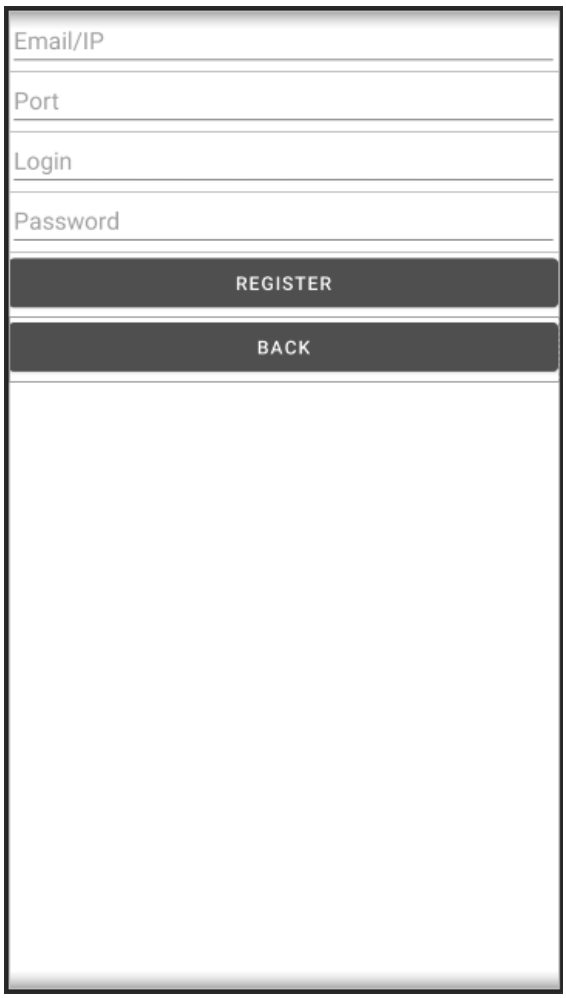


Рисунок 18 – Слой регистрации register_window

На данном слое имеется 4 текстовых поля и 2 кнопки. Кнопка Back (Назад) возвращается нас на стартовый экран.

Кнопка Register запускает цепь проверки введенной информации в 4 текстовых поля. На их бэкграунде написано, где и какую информацию ждет программа для завершения регистрации в системе. Код проверяет, чтобы все поля были заполнены. Одно пропущенное поле и идет возврат к началу с выведением ошибки, чего не хватает. Далее информация отправляется на проверку в базу данных. Программа не создает нового пользователя. На самом деле таким образом происходит запись на сервере для первой и вероятно последней идентификации пользователя на данном устройстве. Смартфон закрепляется только за 1 пользователем и перерегистрировать его на другого человека можно только с

помощью системного администратора и манипуляций в базе данных на сервере. Со смартфона берется IMEI - уникальный идентификационный номер аппарата и приписывается определенному человеку, кому будет принадлежать база данных. Он и не позволяет другому пользователю зарегистрироваться на устройстве. Само изъятие идентификатора производит сервер и в программе никак не описывается.

Сама процедура проверки описана в листинге 1.

Листинг 1 – Проверка регистрации

```
dialog.setPositiveButton("Register", DialogInterface.OnClickListener { dialogInterface,
which ->
if(TextUtils.isEmpty(email.getText().toString())) {
Snackbar.make(root!!, "EnteryourEmailorIPadress", Snackbar.LENGTH_SHORT).show()
return@OnClickListener
}
if(TextUtils.isEmpty(name.getText().toString())) {
Snackbar.make(root!!, "EnterPORT", Snackbar.LENGTH_SHORT).show()
return@OnClickListener
}
if(TextUtils.isEmpty(phone.getText().toString())) {
Snackbar.make(root!!, "EnteryourLOGIN", Snackbar.LENGTH_SHORT).show()
return@OnClickListener
}
if(pass.getText().toString().length() <5) {
Snackbar.make(root!!, "Enterpasswordwithmin 5 simbols", Snackbar.LENGTH_SHORT).show()
return@OnClickListener
}
// Регистрацияпользователя
auth.createUserWithEmailAndPassword(email.getText().toString(), pass.getText().toString())
            .addOnSuccessListener(object: OnSuccessListener<AuthResult?>() {
fun onSuccess(authResult: AuthResult?) {
val user = User()
                    user.setEmail(email.getText().toString())
                    user.setName(name.getText().toString())
                    user.setPass(pass.getText().toString())
                    user.setPhone(phone.getText().toString())
                    users.child(user.name())
                            .setValue(user)
                            .addOnSuccessListener(object : OnSuccessListener<Void?>() {
fun onSuccess(aVoid: Void?) {
Snackbar.make(root!!, "Registration complete", Snackbar.LENGTH_SHORT).show()
                                }
                            })
                }
            })
})
dialog.show()
}
```

Вернемся к стартовому экрану. Описание кнопки Enter (Вход). Данная кнопка нужна уже для повторных входов в программу после прохождения фазы авторизации. Кнопка Enter показана на рисунке 19.

Рисунок 19 – Слой sign_in_window

Принцип работы слоя примерно такой же, как и в Register. За исключением того, что полей проверяется меньше и меняется смысл проверки. Итог будет тот же, мы попадаем на экран главной активности.

Процедура проверки описана в листинге 2.

Листинг 2 – Проверка авторизации

```
dialog.setNegativeButton("Back", DialogInterface.OnClickListener { dialogInterface, which
->dialogInterface.dismiss() })
dialog.setPositiveButton("Enter", DialogInterface.OnClickListener { dialogInterface, which
```

```
->
if (TextUtils.isEmpty(name.getText().toString())) {
        Snackbar.make(root!!, "Enter LogIn", Snackbar.LENGTH_SHORT).show()
return@OnClickListener
}
if (pass.getText().toString().length() <5) {
        Snackbar.make(root!!, "Enter password with min 5 simbols", Snack-
bar.LENGTH_SHORT).show()
return@OnClickListener
}
    auth.createUserWithEmailAndPassword(name.getText().to String, pass.getText().to
String)
            .addOnSuccessListener(new OnSuccessListener<AuthResult>{
@Override
                public void onSuccess(AuthResult authResult) {
// Успешноедобавление, можетеуказатькакое
startActivity(new Intent(MainActivity this, CrimeListViewModel class));
                    finish();
// полебудетключем, кпримеруполе name
}).addOnFailureListener(new OnFailureListener() {
SnackBar.make(root, "Ошибкаавторизации" +e.getMassege() SnackBar.LENGHT_SHORT).show
}
@Override
                public void onFailure(@NonNull Exception e) {

                }
            });
})
dialog.show()
}
```

Пройдя стартовую страницу, попадаем на главную активность. На список клиентов и заявок. Здесь можно увидеть все адреса и наименования организаций, которые по итогам авторизации привязаны к данному пользователю. Это крайне важно. Нельзя чтобы базы перекликались. Такое возможно только у ЗНОП отделов продаж. Всё завязано на иерархии подчиненности в 1С. По сколько у ЗНОПа множество подчиненных, ему будут выгружаться все клиенты своих подопечных. Это сделано специально, для подстраховки, если человек уволился, заболел и т.д. Либо смартфон с логином и паролем передается заменяющему отсутствующего торговому представителю.

Главная активность будет выглядеть как представлено на рисунке 20.

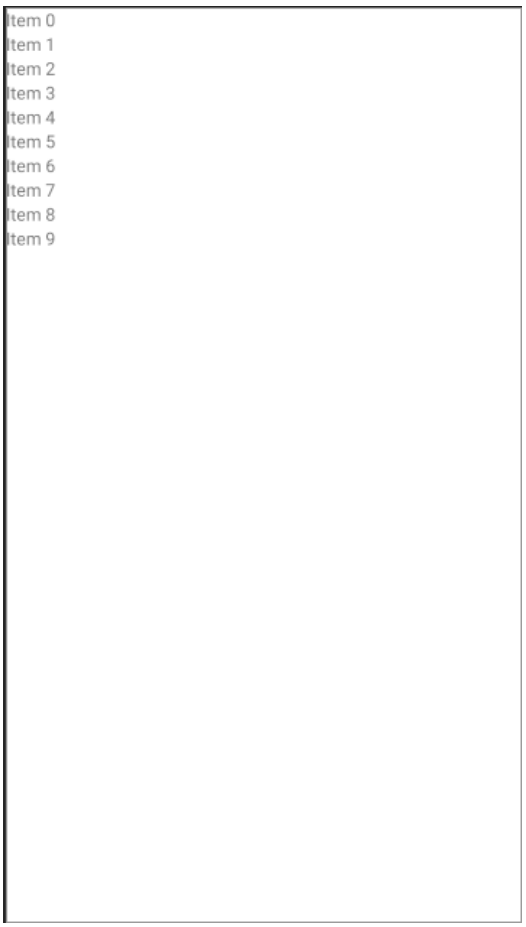


Рисунок 20 – Слой fragment_crime_list

Так выглядит именно, что пустой список заказов. Т.е. без подключения и подкачки базы данных. По сколько у нас нету доступа к базам данных SQL. Была реализована временная мера, а именно построена таблица на SQLLite. Писать на нём очень непросто и тем более его вычищать. Смартфон становится хранителем базы данных, что откликается во времени работы программы. Чем больше база, тем дольше идет обращение к записям. Следовательно, производительность падает. Благодаря Android Studio, подкачана фиктивная база данных и проверена работоспособность списка Код создания базы написан в листинге 4.

Листинг 3 – Код класса для обращения к БД

```
@Entity
data class Crime(@PrimaryKey val id: UUID = UUID.randomUUID(),
var title: String ="",
var date: Date = Date(),
var isSolved: Boolean = false,
```

```
var suspect: String = ""){
val photoFileName
get() = "IMG_$id.jpg"
}
packagecom.bignerdranch.android.criminalintent.database
importjava.util.UUID
@Dao
interfaceCrimeDao {
    @Query("SELECT * FROMcrime")
    fungetCrimes(): an-
droidx.lifecycle.LiveData<List<com.bignerdranch.android.criminalintent.Crime>>
    @Query("SELECT * FROMcrimeWHEREid=(:id)")
    fungetCrime(id: UUID): an-
droidx.lifecycle.LiveData<com.bignerdranch.android.criminalintent.Crime?>
    @Update
fun updateCrime(crime: com.bignerdranch.android.criminalintent.Crime)
    @Insert
fun addCrime(crime: com.bignerdranch.android.criminalintent.Crime)
}
@Database(entities = [ com.bignerdranch.android.criminalintent.Crime::class ], version=2,
exportSchema = false)
@TypeConverters(CrimeTypeConverters::class)
abstract class CrimeDatabase : RoomDatabase() {
abstract fun crimeDao(): CrimeDao
}
val migration_1_2 = object : Migration(1, 2) {
override fun migrate(database: SupportSQLiteDatabase) {
        database.execSQL(
"ALTER TABLE Crime ADD COLUMN suspect TEXT NOT NULL DEFAULT ''"
)
    }
}
```

Обновление ячеек происходит путем создания новой "версии" базы данных. По сути программа понимает это, как создание альтернативной версии исходной базы данных и поэтому производит замещение всех ячеек. Пример заполнения базы данных фиктивными данными показан на рисунке 21.

Рисунок 21 – Слой fragment_crime_list с тестовыми данными

Галочки – это отдельный слой для отображения написания каждой заявки, точнее проверка того, что заявка по статусу "Готова". Это означает, что больше её редактировать нельзя, как и отправлять. Это замок, для предотвращения задвоения поступающих заказов. Слой показан на рисунке 22.

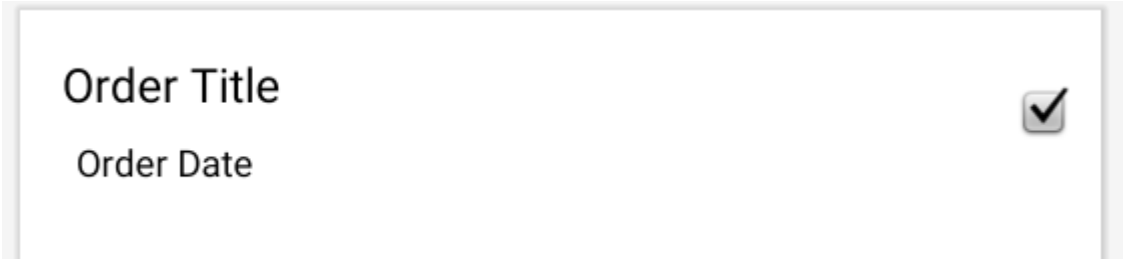


Рисунок 22 – Слой list_item_crime

Функция добавления нового заказа. Это так называемый Action bar, показанный на рисунке 23. Он создан, по умолчанию его нет, для добавления функции создания новых заказов, т.е. создания перехода между слоями fragment_crime_list и fragment_crime без необходимости иметь какие-либо записи изначально.

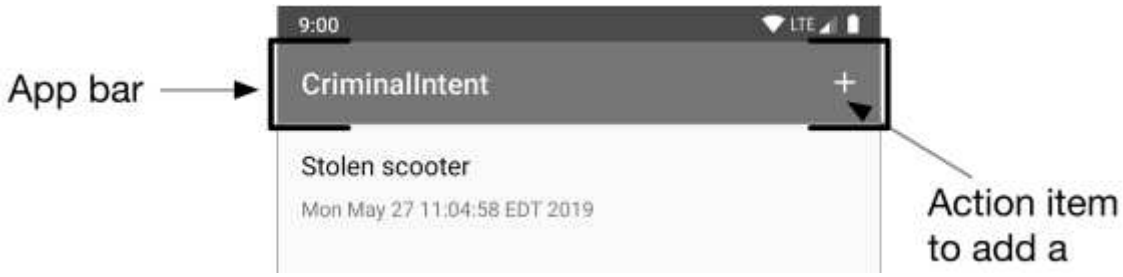


Рисунок 23 –Actionbar

Реализация слоя создания самого заказа, показана на рисунке 24.

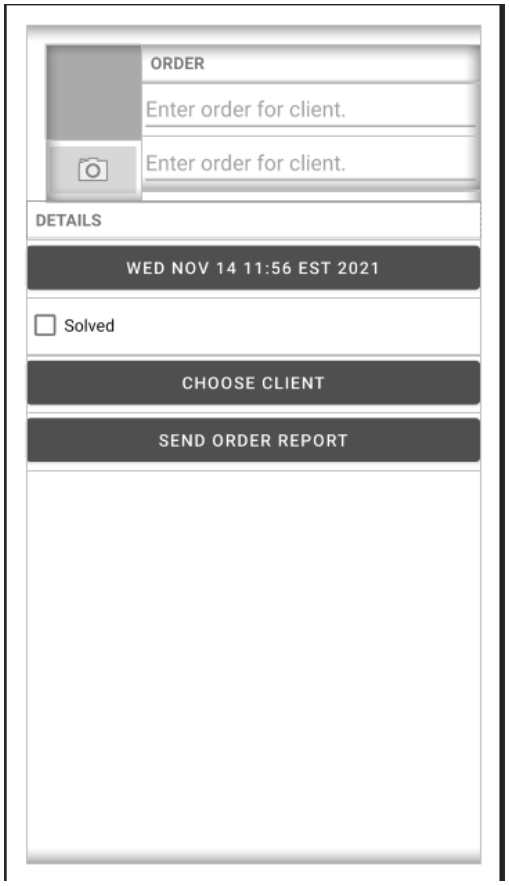


Рисунок 24 – Слой fragment_crime

Данный слой самый комплексный в написании, т.к. тут идет весь функционал программы заказа товаров. Сбор фото материала. Это своего рода слежка за работой торговых представителей. Решено отказаться от традиционного снятия GPS и GSM координат, т.к. от них легко избавиться и подделать. Суть заключается в том, что торговый представитель всегда должен быть в дороге и создавать заказы непосредственно перед клиентом. И чтобы доказать, что человек отработал свой день, к каждому созданному заказу прикладывается фотография места, где именно создавалась заявка. Без фото материала, заявку не отправить.

Текстовые поля для введения названий номенклатуры и ввода необходимого количества показаны на рисунке 25.

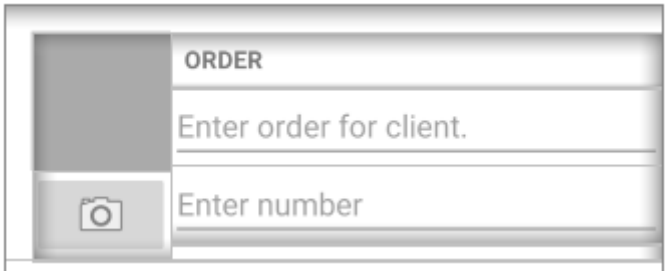


Рисунок 25 – Слой fragment_crime

Сейчас всё это реализовано в виде текстового поля, а не отдельной ссылки на дополнительный слой с наименованием всей номенклатурной группы. Реализован такой вариант, как временная мера из-за отсутствия доступа к таблицам баз данных 1С. Выгрузка в будущем будет осуществляется именно из неё. Таблицы клиентов, номенклатуры и заказов. Сейчас данная реализация подходит разве что для малого бизнеса. Цены и вся информация по складу, тоже будет подкачиваться из таблиц номенклатуры.

Маркировщик даты показан на рисунке 26.

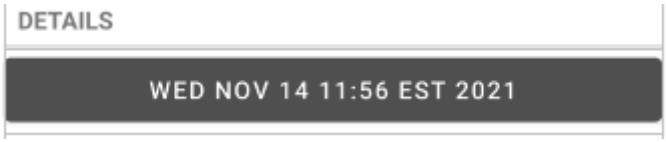


Рисунок 26 – Слойfragment_crimeДата

Это поле виджета. При нажатии на него открывается календарь и выбирается дата. Это поле ничто иное, как выбор даты поставки, т.е. какого числа должна произвестись отгрузка на клиента. Календарь показан на рисунке 27.

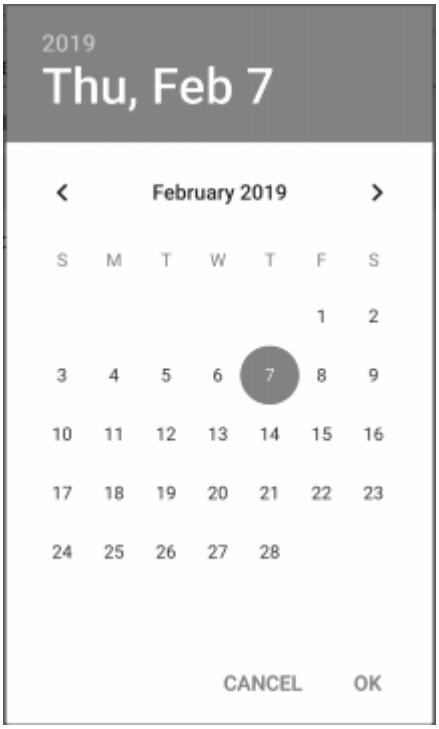


Рисунок 27 – Виджет «Календарь»

Самое главное и самое важное для каждой заявки. Это флаг готовности заказа, показанный на рисунке 28.

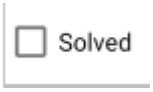


Рисунок 28– Флаг готовности на слое fragment_crime

Этот флаг, при его отметке, сигнализирует, что при нажатии клавиши Send Order Report заказ отправится на сервер и уже не будет подлежать редактированию. Это окончательное решение. Перед установкой флага необходимо проверить тщательно сам заказ, выбрать покупателя и установленную дату, т.к. возможности вернуть всё вспять или отказаться от заказа уже пользователю не выйдет. Это сделано специально, чтобы не нарушать работу склада и операторов. Склад собирает сотни заказов в час, у мастеров склада просто не будет времени и возможностей для поиска и расформирования отказных заказов. К тому же им не прийдет оповещений об отказах. Отказ от заявки, это комплексное мероприятие, где задействованы будут целых 4 отдела, а именно:

- склад;
- отдел операторов (собирают документы для отправки заказов);
- коммерческий отдел;
- программисты 1С.

Последние функции это Choose Client (Выбрать клиента) и Send Order Report. Их функционал схож с тем, что оба задействуют список контактов. Лучше всего, оповещать клиента сразу, как только создается и отправляется на сервер его заявка. Клиент в курсе дел, и пользователь не беспокоится об отчетности. Выбор типа отправки полностью на пользователе. Отправка возможна как почтой, так и через SMS ну и любые мессенджеры (Viber, Whatapp, Telegram). Отправка отчета показана на рисунке 29.

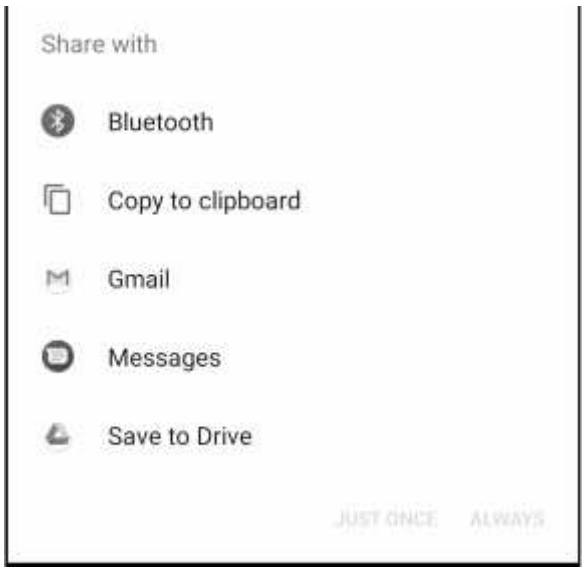


Рисунок 29 – Отправка отчета о заказе

На рисунке 30 показан тестовый выбор пользователя на отправку заказа. Выбирается из контактов пользователя непосредственно с телефона.

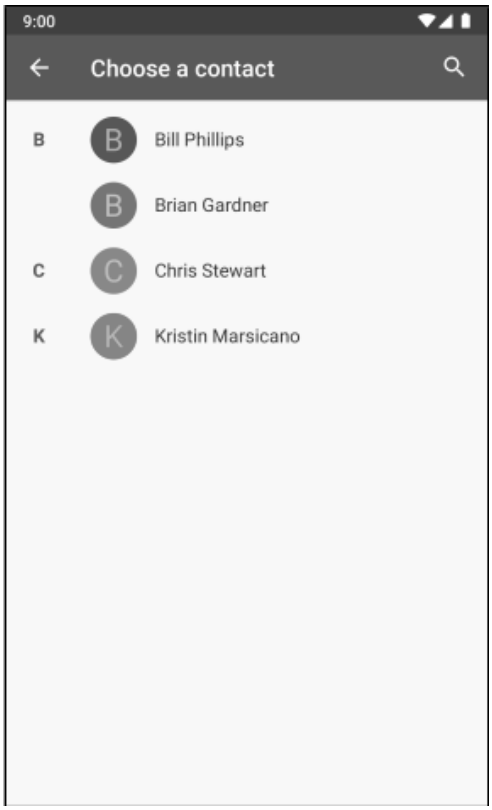


Рисунок 30 – Список контактов

Локализация и Talk Back. С локализацией всё просто. По манифесту, Android Studio всячески рекомендует сохранять все имена переменных и текстовых названий кнопок, слоёв и т.д. в специфичном файле под названием strings.xml. Создавать локализацию на любой язык мира из этого файла проще простого. Все имена в кодах программы идут, как ссылки на строки именно этого файла.

Для понимания как всё выглядит, код strings показан в листинге 4.

Листинг 4 – Код файла локализации string

```
<resources>
<string name="app_name">Tavria App</string>
<string name="crime_title_hint">Enter order for client.</string>
<string name="crime_title_hint1">Enter number</string>
<string name="crime_title_label">Order</string>
<string name="crime_details_label">Details</string>
<string name="crime_solved_label">Solved</string>
<string name="crime_photo_button_description">Take photo of order scene</string>
<string name="crime_photo_no_image_description">
Order scene photo (not set)
</string>
<string name="crime_photo_image_description">Order scene photo (set)</string>
<string name="new_crime">New Order</string>
<string name="crime_suspect_text">Choose Client</string>
<string name="crime_report_text">Send Order Report</string>
<string name="app_in_name">Tavria App</string>
<string name="app_name_after">Register App</string>
<string name="text_bottom">Enter your ID</string>
<string name="crime_report">%1$s!
The crime was discovered on %2$s. %3$s, and %4$s
</string>
<string name="crime_report_solved">The case is solved</string>
<string name="crime_report_unsolved">The case is not solved</string>
<string name="crime_report_no_suspect">there is no suspect.</string>
<string name="crime_report_suspect">client is %s.</string>
<string name="crime_report_subject">Order Report</string>
<string name="send_report">Send order report via</string>
</resources>
```

Листинг кода strings для русской версии показан в листинге 5.

Листинг 5 – Код файла русской локализации string

```
<resources>
<string name="app_name">Tavria App</string>
<string name="crime_title_hint">Введитезаказдляклиента</string>
<string name="crime_title_label">Заказ</string>
<string name="crime_details_label">Детали</string>
<string name="crime_solved_label">Решено</string>
```

```
<string name="new_crime">Новыйзаказ</string>
<string name="crime_suspect_text">Выберитеклиента</string>
<string name="crime_report_text">Отправитьотчетклиенту</string>
<string name="app_in_name">Tavria App</string>
<string name="app_name_after">Регистрация</string>
<string name="text_bottom">Введитесвоиидентификационныеданные</string>
<string name="crime_report">%1$s!
The crime was discovered on %2$s. %3$s, and %4$s
</string>
<string name="crime_report_solved">The case is solved</string>
<string name="crime_report_unsolved">The case is not solved</string>
<string name="crime_report_no_suspect">there is no suspect.</string>
<string name="crime_report_suspect">client is %s.</string>
<string name="crime_report_subject">Отчетозаказе</string>
<string name="send_report">Отправитьотчетозаказе via</string>
</resources>
```

Так же можно с помощью этого файла сжимать и расширять разрешение экрана для разных моделей смартфонов и планшетов. Расширение берется из названия к самому файлу, называется он strings(sw600dp). Android уже в зависимости от параметров самой системы сам решает, применять его или нет. Так происходит и с локализацией. В зависимости от языка системы, будет автоматически выбираться локализация программы. Ручной выбор не предусмотрен.

Talk Back – это дополнительная функция для слабовидящих пользователей. Скачивается отдельно из Google Play, показана на рисунке 31.

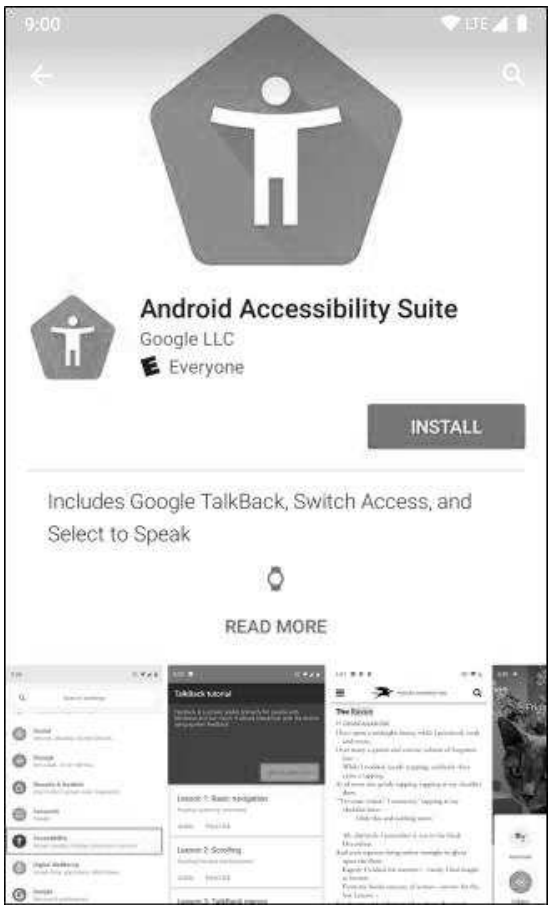


Рисунок 31 – Пакет программ в который входит Talk Back

Функциям программы необходимо прописать тексты озвучки вручную. Пример функции озвучки написан в листинге 6:

Листинг 6 – Код описания действия

```
android:contentDescription="@string/crime_photo_no_image_description" />
```

Т.е. по сути добавляется описание к действиям и функциям. От чего программа уже начинает озвучивать действия пользователя. Полный листинг всех описаний входит в состав всех файлов и приложен в приложении А.

Вывод по второй главе

Во второй главе выпускной квалификационной мы разработали мобильное приложение, где были проанализированы логические и физические моделирования БД. Рассмотренные пункты устанавливают структуру БД, нужные для её создания. Создана концептуальная модель потоков данных для процесса отправки заказов от пользователя, опираясь на которую были определены функциональные требования, предъявляемые к автоматизированной информационной системе для подачи заявки клиентом, связанные с созданием мобильного приложения для отображения данных товара, и созданием (отправки) заявки на приобретение продукции МПК «Таврия».

3. РАСЧЕТ ЭКОНОМИЧЕСКОЙ ЭФФЕКТИВНОСТИ ОТ ВНЕДРЕНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Разработка программного обеспечения сопряжена с капитальным вложением, как на приобретение техники и программного обеспечения, так и на разработку проектов, выполнение подготовительных работ и т.д. Поэтому необходимо экономическое обоснование целесообразности ведения разработок.

Хорошо составленный бизнес-план является частью успеха хорошего распространения и продаж разрабатываемого программного обеспечения. Он содержит данные необходимые для эффективного внедрения на рынок и успешного варьирования в зависимости от состояния рынка, а также содержит ряд показателей, дающих представления об экономической и коммерческой эффективности данной работы.

3.1 Определение затрат на разработку продукта

В данном разделе определяются необходимые инвестиции для разработки и реализации программного продукта. Для этого произведём расчет единовременных затрат, показанный в таблице 2.

Таблица 2 – Расчет единовременных затрат

№	Наименование технического средства и ПО	Тип или модель	Стоимость, в руб.
1	Операционная система	MS Windows 10	6000
2	Системный блок	ACER Aspire XC-830, Intel Celeron J4025D, DDR4 4ГБ, 128ГБ (SSD), Intel UHD Graphics 600, CR	17999
3	Монитор	Asus VA24EHE	8499
4	Принтер	HP LaserJet 1018	8000
5	Клавиатура и мышь	Logitech Wireless Combo MK270	1999
6	Стол компьютерный		5000
7	Стул		4000
Итого:			51497

Подсчитаем единовременные затраты:

Зк = 6000 + 17999 + 8499 + 8000 + 1999 + 5000 + 4000 = 51497 руб.

Затраты на создание программного продукта складываются из расходов по оплате труда разработчика и расходов по оплате машинного времени при отладке кода.

Произведём расчет текущих затрат, показанных в таблице 3.

Таблица 3 – Календарный план-график основных этапов выполнения ВКР

Наименование этапа	Дата начала	Длительность, недель
1. Подготовительный (подбор, изучение литературы, анализ состояния вопроса, составление технического задания)	12.01.2021	1
2. Теоретическая разработка (разработка теоретических обоснований)	19.01.2021	1
3. Проектирование и выполнение технического задания на ЭВМ	26.01.2021	4
4. Консультации с руководителем проекта	23.02.2021	0
5. Машинные расчёты и отчёт в электронном виде	04.06.2021-18.06.2021	2

Затраты на теоретическую часть и работу с литературой рассчитываются по формуле 3.1:

$$Зт = Ср * Тт \quad (3.1)$$

где Ср = 225 руб./час - тарифная ставка инженера II категории;

Тт - время, затраченное на работу с литературой и теоретический анализ;

Зт = 225х40 = 9000 руб.

Затраты на теоретические разработки рассчитываются по формуле 3.2:

$$Зрс = Ср * Тр \quad (3.2)$$

где Тр - время, затраченное на теоретические разработки.

Зрс = 225 х40 = 9000 руб.

Затраты на проектирование и выполнение технического задания на ЭВМ рассчитываются по формуле 3.3:

$$Зп = Ср* Тп \quad (3.3)$$

где Тп - время, затраченное на проектирование и выполнение технического задания на ЭВМ.

Зп = 225х160 = 36000 руб.

Затраты на оплату машинного времени рассчитываются по формуле 3.4:

$$Зм = См * Тм \quad (3.4)$$

где См =30 руб./час - стоимость одного часа машинного времени;

Тм =160 + 80 час - время использования машины.

Зм = 30 х240 = 7200 руб.

Затраты на консультацию с руководителем рассчитываются по формуле 3.5:

$$Зкп = Скр * Ткп \quad (3.5)$$

где Скр =35руб./час - тарифная ставка ведущего инженера;

Ткп - время консультаций с руководителем.

Зкп = 35 х 0 = 0 руб.

Полные затраты при использовании ЭВМ для расчетов (3.6) и составления отчета представлены в таблице 4.

$$Зсум = Зт + Зтр + Зп + Зм + Зкп \quad (3.6)$$

Зсум = 9000 + 9000 + 36000 + 7200 + 0 = 61200 руб.

Таблица 4 – Составляющие затрат при проектировании

Наименование затрат	**Обозначение**	**Сумма**	
		в рублях	**в %**
1. Работа с литературой и теоретическая часть	Зт	9000	14,7
2. Теоретические разработки	Зтр	9000	14,7
3. Проектирование и выполнение технического задания на ЭВМ	Зп	36000	37,9
4. Оплата машинного времени	Зм	7200	11,8
5. Консультация с руководителем	Зкп	0	0
Итого:	Зсум	61200	100

На основании значений, полученных по данным формулам, составляется календарный план-график работы над проектом.

Затраты на накладные расходы, связанные с проектированием и отладкой ПП. Накладные расходы, связанные с проектированием и отладкой ПП, в том числе стоимость используемых материалов (бумаги, картриджей к принтерам и т.п.), описаны в таблице5.

Для выполнения работ необходимо материалов на сумму 3270,00руб, показанных в таблице 5.

Таблица 5 – Потребности

Материалы	**Потребность**	**Стоимость одной ед., руб.**	**Общая стоимость, руб.**	**Примечания**
Бумага	300 листов	0.5	150	Печать текстов
Расходные материалы для лазерного принтера	1 шт.	1000	1000	Для печати необходимых данных
Перезаписываемый диск флеш-накопитель	1 шт.	800	800	Резервирование данных и материалов
Использование сети Internet	150 часов	8,8	1320	Поиск информации и литератур.
Итого:	3270			

Оплата за пользование электричеством составляет

Зэ=0,5кВатт*320 часа* 2,05 руб= 328,00 руб.

Текущие затраты (себестоимость) (С) включают затраты на постановку задачи, разработку алгоритмов и программ, а также затраты, связанные с содержанием и эксплуатацией ВТ.Рассчитываем их по следующей формуле 3.7:

$$С = Зпр + Нз + Змаш + Зн + Зэ \quad (3.7)$$

где Зпр - затраты на заработную плату проектировщиков и программистов;

Зпр = 25019,2 руб.

Таблица 6 – Текущие затраты

Наименование статей затрат	**Сумма**
1. Затраты на заработную плату (Зпр)	40000,00
2. Отчисления в фондыс заработной платы труда (ФФОМС, ПФР, ФСС)(Зф)	12000,00
3. Затраты, связанные с использованием машинного времени (Змаш)	7200,00
5. Накладные расходы (Зн)	3270,00
6. Расходы на электричество при пользовании вычислительной техникой	328,00
Итого затрат (З)	62798,00

Сведем расчет затрат в общую таблицу 7:

Таблица 7 – Расчет затрат на разработку программного средства

Затраты и ожидаемые доходы от внедрения ИС	**Период**			
Затраты	1 квартал	2 квартал	3 квартал	4 квартал
1. Капитальные затраты	51497,00	51497,00	51497,00	51497,00
3. Текущие затраты:	62798,00	62798,00	62798,00	62798,00
Заработная плата	40000,00	40000,00	40000,00	40000,00
Отчисления в фонды (ФФОМС, ПФР, ФСС)	12000,00	12000,00	12000,00	12000,00
Затраты, связанные с использованием машинного времени	7200,00	7200,00	7200,00	7200,00
Накладные расходы	3270,00	3270,00	3270,00	3270,00
-Расходы на электричество при пользовании вычислительной техникой	328,00	328,00	328,00	328,00
Итого затрат:	114295,00	114295,00	114295,00	114295,00

Отчисления от фонда оплаты труда (Зф) (30,2 % от Зпр);

Зф = 30,2%*Зпр = 12080руб.

Змаш - затраты, связанные с использованием машинного времени на разработку и отладку программ. Змаш = 7200 руб.

Зн - накладные расходы, связанные с проектированием и отладкой ПП, в том числе стоимость используемых материалов. После расчета единовременных и текущих затрат составим общую таблицу потребности в инвестициях.

3.2 Определение доходов от внедрения в работу программного обеспечения

Сведем расчет всех доходов в общую таблицу 8:

Таблица 8 – Доходы

Доходы/Месяцы	**1 квартал**	**2 квартал**	**3 квартал**	**4 квартал**
1 Снижение затрат на ручную обработку больших объемов информации, руб.	12150	12150	12150	12150
2. Привлечение дополнительных клиентов, руб.	0	0	0	0
Итого доходов тыс. руб.:	12150	12150	12150	12150

На данный момент при отсутствии программного обеспечения на предприятии работа по переводу иностранных документов имеет большие временные затраты. Вся информация переводится вручную, что может занять достаточно долгое время. Зная заработную плату за день (в среднем она составляет 300 руб./день) можно подсчитать, что работы с переводом документации обходится предприятию в сумму равную 300*3дн./нед.*4,5нед= 4050 рублей в месяц, (4050*3=12150) за квартал.

За счет введения в эксплуатацию приложения количество клиентов не должно увеличиться, при этом, чем дольше будет работать приложение, тем меньше будет затрат на содержание мобильных устройств.

3.3 Показатели эффективности

Расчет экономической эффективности проводится по следующим показателям:

а) Чистый дисконтированный доход (ЧДД) или интегральный эффект;

б) Индекс доходности (ИД);

в) Внутренняя ставка доходности (ВСД);

г) Срок окупаемости;

Чистый дисконтированный доход - превышение интегральных результатов над интегральными затратами. Определяется как сумма текущих эффектов за весь расчетный период, приведенная к начальному шагу.

Метод дисконтированных денежных потоков предполагает анализ потоков инвестиционного капитала, причем как затратных, так и прибыльных, с учетом их временно-стоимостной оценки.

Выбор данного метода был обусловлен следующими причинами:

- необходимость инфляционной корректировки показателей;
- обеспечение наглядности и простоты расчета срока окупаемости;
- стабильность и предсказуемость денежных потоков;
- равномерность денежных потоков.

Этот метод нашел широкое применение в зарубежной практике в ходе оценки инвестиций в материальные и нематериальные активы, особенно в случаях нестабильной макроэкономической ситуации (инфляция, возможность кризисных ситуаций и т. д.).

Суть этого метода заключается в отнесении предполагаемых денежных потоков (и положительных, и отрицательных) к временным интервалам, начиная с момента начала инвестирования. Как правило, при малых показателях рентабельности продукта сроком анализа этого метода выбирают нормативный «срок жизни» продукта, или срок окупаемости продукта-субститута. В данном случае для установления срока окупаемости достаточно ограничиться сроком в 3 года.

К недостаткам данного метода можно отнести то, что он не позволяет учесть случайные риски (как внешние, так и внутренние), а также известную условность расчета срока окупаемости, связанную с выбором ставки дисконтирования.

Чистый дисконтированный доход определяется как сумма текущих эффектов за весь расчетный период по формуле 3.8:

$$ЧДД = \sum_{t=1}^{5} (Д_t - P_t) \cdot \frac{1}{(1+\alpha)^t} \quad (3.8)$$

где Дt - результаты, достигаемые на t-ом шаге расчета;

Pt - затраты, осуществляемые на том же шаге;

Дt - Pt - эффект достигаемый на t-ом шаге.

$1/(1+ a)^t$ - коэффициент приведения по времени результатов и затрат;

а - норма дисконта, равная приемлемой для инвестора норме дохода на капитал -0,14.

Предполагаемые доходы от продаж (Д) определяются по формуле 3.9:

$$Д = Цпр*N, \quad (3.9)$$

где Цпр - цена продажи ПП, руб;

N - объем продаж по периодам в соответствии с исследованиями рынка, шт.

Расходы (Р) включают, кроме текущих затрат, расходы на тиражирование и рекламупо формулам 3.10 и 3.11:

$$Р = Зтир*N+Зр, \quad (3.10)$$

$$Р = Зтир*N + Зр*n\ мес; \quad (3.11)$$

где n - количество месяцев в рассматриваемом периоде

Таблица 9 – Технико-экономические показатели работы

Периоды (месяц)	**Показатели**		**1/(1+a)^t**	**Дисконтированные**	**Годовая экономическая эффективность**	**ЧДД с нарастающим итогом**	
	Доходы	Расходы					
	1	2	3	4=1*3	5=2*3	6=4-5	7
1 квартал 2021	42150	114295,00	0.877192982	36973.68	100258.77	-63285.09	-51009.91
2 квартал 2021	52150	114295,00	0.769467528	40127.73	87946.30	-47818,57	-66476.49
3 квартал 2021	62150	114295,00	0.674971516	41949.48	77145.86	-35196.38	-79098.62
4 квартал 2021	72150	114295,00	0.592080277	42718.59	67671.81	-24953.22	-89341.78
1 квартал 2022	82150	114295,00	0.455586548	37426.43	52071.26	-14644.83	-99650.17
2 квартал 2022	92150	114295,00	0.519368664	47859.82	59361.24	-11501.42	-102793.6
3 квартал 2022	102150	62798,00	0,529862435	54125,44	33278,30	20851,14	41946,86
Итого:	412900	748568	3.369298852	247055.73	444455.24		

Индекс доходности - представляет собой отношение суммы приведенных эффектов к величине капитальных вложений. Расчитывается по формуле 3.12:

$$ИД = \frac{\sum_{t=1}^{T} Д_t * \frac{1}{(1+\alpha)^t}}{\sum_{t=1}^{T} P_t * \frac{1}{(1+\alpha)^t}} \quad (3.12)$$

Индекс доходности строится из тех же элементов, что и ЧДД. Если ЧДД положителен, то ИД > 1 и наоборот. Индекс доходности: 1,39

Расчет ЧДД инвестиционного проекта показывает, является ли он эффективным при некоторой заданной норме дисконта.

Срок окупаемости - минимальный временной интервал (от начала осуществления проекта), за пределами которого интегральный эффект становится положительным и в дальнейшем остается неотрицательным. Это период (измеряемый в месяцах, кварталах или годах), начиная с которого первоначальные вложения и другие затраты, связанные с проектом, покрывается суммарными результатами его осуществления.

Определим срок окупаемости графически. Для этого необходимо построить график срока окупаемости проекта, показанный на рисунке 32.

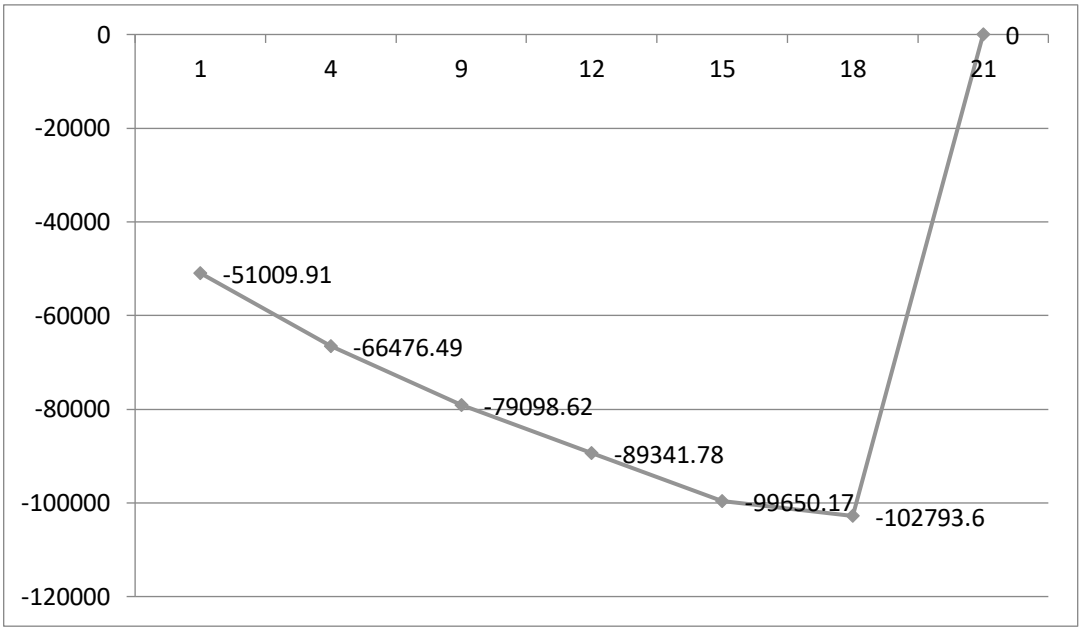


Рисунок 32 – График окупаемости

В данном разделе ВКР проведён анализ основных разделов бизнес-плана, осуществлена калькуляция темы с оценкой экономической эффективности реализации программного продукта.

Затраты на разработку составляют 114295,00 руб. ежеквартально. Экономический эффект от использования данного программного продукта за расчётный период (1 год и 9 месяцев) составит 41946,86 руб. при этом разработчик покроет свои расходы на создание мобильного приложения

ориентировочно за 21 месяц и затем начнёт получать прибыль. Можно сделать вывод о целесообразности и актуальности разработки данного программного продукта.

Вывод по третьей главе

В третьей главе выпускной квалификационной работы определены затраты на разработку мобильного приложения для предприятия МПК «Таврия». Для этого нами был подготовлен исчерпывающий перечень затрат на разработку программного продукта.

Определены доходы от внедрения в работу мобильного приложения, что составило 12150руб. в квартал и 48600 руб. в год. Экономия от приобретения лицензий составит 12000000 руб. в год.

Произведен расчет экономической эффективности нашего программного продукта, для чего мы вычислили срок окупаемости программного продукта, который составит 1 год и 9 месяцев.

Таким образом, разработка и внедрение мобильного приложения для предприятия МПК «Таврия» полностью экономически обосновано и целесообразно, что подтверждается расчетами в нашей выпускной квалификационной работе.

ЗАКЛЮЧЕНИЕ

В рамках выпускной квалификационной работы разработано мобильное приложение, позволяющее пользователю создавать заказы для работы МПК «Таврия». Были проанализированы технологии создания мобильного приложения, учитывая аналитику и ссылаясь на техническое задание, выбрана операционная система мобильного устройства для работы приложения и среда для ее разработки. Приложение «Tavria_app» разработано по заказу МПК «Таврия». В ходе выполнения выпускной квалификационной работы были решены следующие задачи:

- произведен анализ аналогичных проектов и предметной области;
- проанализированы исходные данные заказчика;
- реализовано мобильное приложение на базе операционной системы Android;
- разработан дизайн;
- приложение наполнено тестовыми данными;

Результатом работы является создание мобильного приложения для МПК «Таврия». Разработанный проект соответствует всем требования технического задания, не имеет неисправностей. Приложение будет готово к использованию после прохождения доработки у специалистов 1С (добавление баз данных и реализация выгрузки с серверов 1С) и одобрена к работе специалистами из отдела

безопасности (установка ключей шифрования и подключение к устройствам внешнего вещания). Таким образом, задачи выпускной квалификационной работы с моей стороны полностью решены и цель исследования достигнута.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1 Гриффитс Р. Д. HeadFirst. Программирование для Android [Текст] / Р. Д. Гриффитс – Санкт-Петербург: Питер, 2016. – 704 с.

2 Аарон Хиллегасс. Objective-C. Программирование для Android. 2012г.

3 Дэрси Л. Разработка приложений для Android-устройств. Базовые принципы [Текст] /Л. Дэрси, Ш. Кондер – Том 1. – Москва: Эксмо, 2014. – 598 с.

4 Использование мобильных устройств [Электронный ресурс]. – Режим доступа: http://www.wi-life.ru/stati/wi-fi/marketingovye-stati2/mobiledevices-use-aruba-research

5 Якоб Нильсен, Ралука Будиу. Как создавать идеально удобные приложения для мобильных устройств. 2013г.

6 Основные этапы разработки мобильных приложений [Электронный ресурс] – Режим доступа: https://spark.ru/startup/componentix/blog/4499/ osnovnie-etapi-razrabotkimobilnih-prilozhenij

7 Амелин К. С., Граничин О. Н., Кияев В. И., Корявко А. В.. Введение в разработку приложений для мобильных платформ. [Текст] Издательство ВВМ, 2011.

8 .1С:Мобильная торговля [Электронный ресурс]. – Режим доступа: https://solutions.1c.ru/catalog/mobile_trade

9 Харди Б., Филлипс Б. Программирование под Android. М.: Питер, 2016. — 640 с.

10 Голощапов А. GoogleAndroid: программирование для мобильных устройств. СПб.: БХВ-Петербург, 2010. — 448 с.

11. АСМТ Наполеон [Электронный ресурс]. – Режим доступа: https://grsoft.ru/

12 Бреслав А. Язык программирования Kotlin // Открытые системы. — 2011. — № 09

13 Josh Skeen, David Greenhalgh Kotlin Programming: The Big Nerd Ranch Guide (Big Nerd Ranch Guides)Издательство: Big Nerd Ranch, 2021г. – 700с.

14 ST-мобильная торговля [Электронный ресурс]. – Режим доступа: https://sys4tec.com/products/st-mobile/

ПРИЛОЖЕНИЕ А

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
package="com.bignerdranch.android.criminalintent">
<uses-feature android:name="android.hardware.camera"
android:required="false"/>
<application
android:name=".CriminalIntentApplication"
android:allowBackup="true"
android:icon="@mipmap/ic_launcher"
android:label="@string/app_name"
android:roundIcon="@mipmap/ic_launcher_round"
android:supportsRtl="true"
android:theme="@style/Theme.CriminalIntent">
<activity android:name=".MainActivity">
<intent-filter>
<action android:name="android.intent.action.MAIN" />
<category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
</activity>
<provider
android:name="androidx.core.content.FileProvider"
android:authorities="com.bignerdranch.android.criminalintent.fileprovider"
android:exported="false"
android:grantUriPermissions="true">
<meta-data
android:name="android.support.FILE_PROVIDER_PATHS"
android:resource="@xml/files"/>
</provider>
</application>
</manifest>

package com.bignerdranch.android.criminalintent.database
import java.util.UUID
@Dao
interface CrimeDao {
    @Query("SELECT * FROM crime")
```

```
fun getCrimes(): an-
droidx.lifecycle.LiveData<List<com.bignerdranch.android.criminalintent.Crime>>
    @Query("SELECT * FROM crime WHERE id=(:id)")
fun getCrime(id: UUID): an-
droidx.lifecycle.LiveData<com.bignerdranch.android.criminalintent.Crime?>
    @Update
fun updateCrime(crime: com.bignerdranch.android.criminalintent.Crime)
    @Insert
fun addCrime(crime: com.bignerdranch.android.criminalintent.Crime)
}

@Database(entities = [ com.bignerdranch.android.criminalintent.Crime::class ], version=2,
exportSchema = false)
@TypeConverters(CrimeTypeConverters::class)
```

```
}
val migration_1_2 = object : Migration(1, 2) {
override fun migrate(database: SupportSQLiteDatabase) {
        database.execSQL(
"ALTER TABLE Crime ADD COLUMN suspect TEXT NOT NULL DEFAULT ''"
)
    }
}

packagecom.bignerdranch.android.criminalintent.database
import java.util.Date
import java.util.UUID
class CrimeTypeConverters {
    @TypeConverter
fun fromDate(date: java.util.Date?): Long? {
returndate?.time
}
    @TypeConverter
fun toDate(millisSinceEpoch: Long?): java.util.Date? {
returnmillisSinceEpoch?.let{
java.util.Date(it)
}
}
    @TypeConverter
fun toUUID(uuid: String?): UUID? {
returnUUID.fromString(uuid)
    }
    @TypeConverter
fun fromUUID(uuid: UUID?): String? {
returnuuid?.toString()
    }
}

package com.bignerdranch.android.criminalintent
import android.R
import android.content.DialogInterface
import android.hardware.biometrics.BiometricPrompt
import android.os.Bundle
import android.text.TextUtils
import android.view.LayoutInflater
import android.widget.RelativeLayout
import androidx.appcompat.app.AppCompatActivity
import com.google.android.material.snackbar.Snackbar
import android.support.design.widget.Snackbar;
import android.support.v7.app.AlertDialog;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.Button;
import java.util.*

@Entity
data class Crime(@PrimaryKey val id: UUID = UUID.randomUUID(),
var title: String ="",

var date: Date = Date(),
var isSolved: Boolean = false,
var suspect: String = ""){
val photoFileName
get() = "IMG_$id.jpg"
}
```

```
package com.bignerdranch.android.criminalintent
import java.util.UUID
class CrimeDetailViewModel() :androidx.lifecycle.ViewModel() {
private val crimeRepository = CrimeRepository.get()
private val crimeIdLiveData = androidx.lifecycle.MutableLiveData<UUID>()
var crimeLiveData: androidx.lifecycle.LiveData<Crime?> =
        androidx.lifecycle.Transformations.switchMap(crimeIdLiveData) { crimeId ->
crimeRepository.getCrime(crimeId)
}
fun loadCrime(crimeId: UUID) {
crimeIdLiveData.value = crimeId
    }
fun saveCrime(crime: Crime) {
crimeRepository.updateCrime(crime)
    }
fun getPhotoFile(crime: Crime): java.io.File {
return crimeRepository.getPhotoFile(crime)
    }
}


packagecom.bignerdranch.android.criminalintent
import android.os.Bundle
import android.text.Editable
import android.text.TextWatcher
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.EditText
import androidx.fragment.app.Fragment
import java.util.UUID
import java.util.Observer
import java.util.Date
import kotlin.sequences.sequence as sequence1
private const val TAG = "CrimeFragment"
private const val ARG_CRIME_ID = "crime_id"
private const val DIALOG_DATE = "DialogDate"
private const val REQUEST_DATE = 0
private const val REQUEST_CONTACT = 1
private const val REQUEST_PHOTO = 2
private const val DATE_FORMAT = "EEE, MMM, dd"

class CrimeFragment: Fragment(), DatePickerFragment.Callbacks {
private lateinit var crime: Crime
private lateinit var photoFile: java.io.File
private lateinit var photoUri: android.net.Uri
private lateinit var titleField: EditText
private lateinit var dateButton: android.widget.Button
private lateinit var solvedCheckBox: android.widget.CheckBox
private lateinit var reportButton: android.widget.Button
private lateinit var suspectButton: android.widget.Button
private lateinit var photoButton: android.widget.ImageButton
private lateinit var photoView: android.widget.ImageView

private val crimeDetailViewModel: CrimeDetailViewModel by lazy {
ViewModelProviders.of(this).get(CrimeDetailViewModel::class.java)
}
override fun onCreate(savedInstanceState: Bundle?) {
super.onCreate(savedInstanceState)
```

```
crime = Crime()
val crimeId: UUID = arguments?.getSerializable(ARG_CRIME_ID) as UUID
    }
override fun onCreateView(
        inflater: LayoutInflater,
container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
val view = inflater.inflate(R.layout.fragment_crime, container, false)
titleField = view.findViewById(R.id.crime_title) as EditText
dateButton = view.findViewById(R.id.crime_date) as android.widget.Button
solvedCheckBox = view.findViewById(R.id.crime_solved) as android.widget.CheckBox
reportButton = view.findViewById(R.id.crime_report) as android.widget.Button
suspectButton = view.findViewById(R.id.crime_suspect) as android.widget.Button
photoButton = view.findViewById(R.id.crime_camera) as android.widget.ImageButton
photoView = view.findViewById(R.id.crime_photo) as android.widget.ImageView

return view
    }
override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
super.onViewCreated(view, savedInstanceState)
crimeDetailViewModel.crimeLiveData.observe(
viewLifecycleOwner,
Observer { crime ->
crime?.let {
this.crime = crime
photoFile = crimeDetailViewModel.getPhotoFile(crime)
photoUri = FileProvider.getUriForFile(
                        requireActivity(),
"com.bignerdranch.android.criminalintent.fileprovider",
photoFile
)
                    updateUI()
}
            })
    }
override fun onStart() {
super.onStart()
val titleWatcher = object : TextWatcher {
override fun beforeTextChanged(s: CharSequence?, start: Int, count: Int, after: Int) {
            }
override fun onTextChanged(s: CharSequence?, start: Int, before: Int, count: Int) {
crime.title =
generateSequence { }.toString()  // проверитьнабудущее sequence.toString()
}

override fun afterTextChanged(sequence: Editable?) {
            }
        }
titleField.addTextChangedListener(titleWatcher)
solvedCheckBox.apply {
setOnCheckedChangeListener { _, isChecked ->
crime.isSolved = isChecked
}
        }
dateButton.setOnClickListener {
DatePickerFragment.newInstance(crime.date).apply {
setTargetFragment(this@CrimeFragment, REQUEST_DATE)
                show(this@CrimeFragment.requireFragmentManager(), DIALOG_DATE)
}
```

```
        }
reportButton.setOnClickListener {
Intent(Intent.ACTION_SEND).apply {
type = "text/plain"
putExtra(Intent.EXTRA_TEXT, getCrimeReport())
                putExtra(
                    Intent.EXTRA_SUBJECT,
                    getString(R.string.crime_report_subject)
                )
}.also { intent ->
val chooserIntent =
                    Intent.createChooser(intent, getString(R.string.send_report))
                startActivity(chooserIntent)
}
        }
suspectButton.apply {
valpickContactIntent =
                Intent(Intent.ACTION_PICK, ContactsContract.Contacts.CONTENT_URI)
            setOnClickListener {
startActivityForResult(pickContactIntent, REQUEST_CONTACT)
}
val packageManager: PackageManager = requireActivity().packageManager
val resolvedActivity: ResolveInfo? =
packageManager.resolveActivity(
                    pickContactIntent,
                    PackageManager.MATCH_DEFAULT_ONLY
                )
if (resolvedActivity == null) {
                isEnabled = false
}
}
photoButton.apply {
val packageManager: PackageManager = requireActivity().packageManager
val captureImage = Intent(MediaStore.ACTION_IMAGE_CAPTURE)
val resolvedActivity: ResolveInfo? =
                packageManager.resolveActivity(
                    captureImage,
                    PackageManager.MATCH_DEFAULT_ONLY
                )
if (resolvedActivity == null) {
                isEnabled = false
}
            setOnClickListener {
captureImage.putExtra(MediaStore.EXTRA_OUTPUT, photoUri)
val cameraActivities: List<ResolveInfo> =
                    packageManager.queryIntentActivities(
                        captureImage,
                        PackageManager.MATCH_DEFAULT_ONLY
                    )
for (cameraActivity in cameraActivities) {
                    requireActivity().grantUriPermission(
                        cameraActivity.activityInfo.packageName,
                        photoUri,
                        Intent.FLAG_GRANT_WRITE_URI_PERMISSION
                    )
}
                startActivityForResult(captureImage, REQUEST_PHOTO)
}
        }
return view  // проверитьретурн
```

```
}
override fun onStop() {
super.onStop()
        crimeDetailViewModel.saveCrime(crime)
    }
override fun onDetach() {
super.onDetach()
        requireActivity().revokeUriPermission(photoUri,
            Intent.FLAG_GRANT_WRITE_URI_PERMISSION)
    }
override fun onDateSelected(date: Date) {
        crime.date = date
        updateUI()
    }

private fun updateUI() {
        titleField.setText(crime.title)
        dateButton.text = crime.date.toString()
        solvedCheckBox.apply {
isChecked = crime.isSolved
            jumpDrawablesToCurrentState()
}
if (crime.suspect.isNotEmpty()) {
            suspectButton.text = crime.suspect
        }
        updatePhotoView()
    }
private fun updatePhotoView() {
if (photoFile.exists()) {
val bitmap = getScaledBitmap(photoFile.path, requireActivity())
            photoView.setImageBitmap(bitmap)
            photoView.contentDescription =
                    getString(R.string.crime_photo_image_description)
        } else {
            photoView.setImageDrawable(null)
            photoView.contentDescription =
                    getString(R.string.crime_photo_no_image_description)
        }
    }
override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {
when{
            resultCode != Activity.RESULT_OK ->return
requestCode == REQUEST_CONTACT && data != null -> {
val contactUri: Uri? = data.data
// Указать, для каких полей запрос должен возвращать значения.
valqueryFields = arrayOf(ContactsContract.Contacts.DISPLAY_NAME)
// Запрос contactUri похожнапредложение "where"
val cursor = requireActivity().contentResolver
                    .query(contactUri, queryFields, null, null, null)
                cursor?.use {
// Отобразитьхотябы 1 результат
if (it.count == 0) {
return
}
// Первыйстолбецпервойстрокиданных - имя "suspect"
it.moveToFirst()
val suspect = it.getString(0)
crime.suspect = suspect
crimeDetailViewModel.saveCrime(crime)
suspectButton.text = suspect
```

```
}
}
            requestCode == REQUEST_PHOTO -> {
                requireActivity().revokeUriPermission(photoUri,
                    Intent.FLAG_GRANT_WRITE_URI_PERMISSION)
                updatePhotoView()
            }
        }
    }
private fun getCrimeReport(): String {
val solvedString = if (crime.isSolved) {
            getString(R.string.crime_report_solved)
        } else {
            getString(R.string.crime_report_unsolved)
        }
val dateString = DateFormat.format(DATE_FORMAT, crime.date).toString()
var suspect = if (crime.suspect.isBlank()) {
            getString(R.string.crime_report_no_suspect)
        } else {
            getString(R.string.crime_report_suspect, crime.suspect)
        }
return getString(
            R.string.crime_report,
            crime.title, dateString, solvedString, suspect
        )
    }
companion object {
fun newInstance(crimeId: UUID): CrimeFragment {
val args = Bundle().apply {
putSerializable(ARG_CRIME_ID, crimeId)
}
return CrimeFragment().apply {
arguments = args
}
}
    }
}

packagecom.bignerdranch.android.criminalintent
import java.util.UUID
import java.util.Observer
import androidx.recyclerview.widget.RecyclerView
private const val TAG = "CrimeListFragment"
class CrimeListFragment : androidx.fragment.app.Fragment() {
/**
     * Требуемыйинтерфейс
     */
interface Callbacks {
fun onCrimeSelected(crimeId: UUID)
    }
private var callbacks: Callbacks? = null
    private lateinit var crimeRecyclerView: RecyclerView
private var adapter: CrimeAdapter? = CrimeAdapter(emptyList())

private val crimeListViewModel: CrimeListViewModel by lazy {
androidx.lifecycle.ViewModelProvider.of(this).get(CrimeListViewModel::class.java)
}
override fun onAttach(context: android.content.Context) {
super.onAttach(context)
callbacks = context as Callbacks?
```

```
    }
override fun onCreate(savedInstanceState: android.os.Bundle?) {
super.onCreate(savedInstanceState)
        setHasOptionsMenu(true)
    }
override fun onCreateView(
        inflater: android.view.LayoutInflater,
        container: android.view.ViewGroup?,
        savedInstanceState: android.os.Bundle?
    ): android.view.View? {
valview = inflater.inflate(R.layout.fragment_crime_list, container, false)
crimeRecyclerView =
            view.findViewById(R.id.crime_recycler_view) as RecyclerView
crimeRecyclerView.layoutManager = an-
droidx.recyclerview.widget.LinearLayoutManager(context)
crimeRecyclerView.adapter = adapter
return view
    }
override fun onViewCreated(view: android.view.View, savedInstanceState: an-
droid.os.Bundle?) {
super.onViewCreated(view, savedInstanceState)
crimeListViewModel.crimeListLiveData.observe(
viewLifecycleOwner,
            Observer { crimes ->
crimes?.let {
android.util.Log.i(TAG, "Got orders ${crimes.size}")
                    updateUI(crimes)
}
            })
    }
override fun onDetach() {
super.onDetach()
        callbacks = null
}
override fun onCreateOptionsMenu(menu: android.view.Menu, inflater: an-
droid.view.MenuInflater) {
super.onCreateOptionsMenu(menu, inflater)
        inflater.inflate(R.menu.fragment_crime_list, menu)
    }
override fun onOptionsItemSelected(item: android.view.MenuItem): Boolean {
return when (item.itemId) {
            R.id.new_crime -> {
val crime = Crime()
                crimeListViewModel.addCrime(crime)
                callbacks?.onCrimeSelected(crime.id)
true
}
else ->return super.onOptionsItemSelected(item)
        }
    }
private fun updateUI(crimes: List<Crime>) {
        adapter = CrimeAdapter(crimes)
crimeRecyclerView.adapter = adapter
}

private inner class CrimeHolder(view: android.view.View)
        : RecyclerView.ViewHolder(view), android.view.View.OnClickListener {
private lateinit var crime: Crime
private val titleTextView: android.widget.TextView =
itemView.findViewById(R.id.crime_title)
```

```
private val dateTextView: android.widget.TextView = itemView.findViewById(R.id.crime_date)
private val solvedImageView: android.widget.ImageView =
itemView.findViewById(R.id.crime_solved)
init {
itemView.setOnClickListener(this)
        }
fun bind(crime: Crime) {
this.crime = crime
titleTextView.text = this.crime.title
            dateTextView.text = this.crime.date.toString()
solvedImageView.visibility = if (crime.isSolved) {
                android.view.View.VISIBLE
} else {
                android.view.View.GONE
}
        }
override fun onClick(v: android.view.View) {
callbacks?.onCrimeSelected(crime.id)
        }
    }
private inner class CrimeAdapter(var crimes: List<Crime>)
        : RecyclerView.Adapter<CrimeHolder>() {
override fun onCreateViewHolder(parent: android.view.ViewGroup, viewType: Int)
                : CrimeHolder {
val view = layoutInflater.inflate(R.layout.list_item_crime, parent, false)
return CrimeHolder(view)
        }
override fun getItemCount() = crimes.size
override fun onBindViewHolder(holder: CrimeHolder, position: Int) {
val crime = crimes[position]
            holder.bind(crime)
        }
    }
}

packagecom.bignerdranch.android.criminalintent

class CrimeListViewModel : androidx.lifecycle.ViewModel() {
private val crimeRepository = CrimeRepository.get()
val crimeListLiveData = crimeRepository.getCrimes()
fun addCrime(crime: Crime) {
crimeRepository.addCrime(crime)
    }
}

package com.bignerdranch.android.criminalintent
import java.util.UUID
private const val DATABASE_NAME = "crime-database"
class CrimeRepository private constructor(context: android.content.Context) {
private val database : CrimeDatabase = Room.databaseBuilder(
        context.applicationContext,
        CrimeDatabase::class.java,
DATABASE_NAME
).addMigrations(migration_1_2)
        .build()
private val crimeDao = database.crimeDao()
private val executor = java.util.concurrent.Executors.newSingleThreadExecutor()
private val filesDir = context.applicationContext.filesDir
fun getCrimes(): androidx.lifecycle.LiveData<List<Crime>> = crimeDao.getCrimes()
fun getCrime(id: UUID): androidx.lifecycle.LiveData<Crime?> = crimeDao.getCrime(id)
```

```
fun updateCrime(crime: Crime) {
executor.execute {
crimeDao.updateCrime(crime)
}
}
fun addCrime(crime: Crime) {
executor.execute {
crimeDao.addCrime(crime)
}
}
fun getPhotoFile(crime: Crime): File = File(filesDir, crime.photoFileName)
companion object {
private var INSTANCE: CrimeRepository? = null
        fun initialize(context: Context) {
if (INSTANCE == null) {
INSTANCE = CrimeRepository(context)
            }
        }
fun get(): CrimeRepository {
return INSTANCE ?:
throw IllegalStateException("CrimeRepository must be initialized")
        }
    }
}
```

```
packagecom.bignerdranch.android.criminalintent

class CriminalIntentApplication : android.app.Application() {
override fun onCreate() {
super.onCreate()
        CrimeRepository.initialize(this)
    }
}
```

```
package com.bignerdranch.android.criminalintent
import java.util.Date
private const val ARG_DATE = "date"
class DatePickerFragment : androidx.fragment.app.DialogFragment() {
interface Callbacks {
fun onDateSelected(date: Date)
    }
override fun onCreateDialog(savedInstanceState: android.os.Bundle?): android.app.Dialog {
val dateListener = android.app.DatePickerDialog.OnDateSetListener {
_: android.widget.DatePicker, year: Int, month: Int, day: Int ->
val resultDate : Date = java.util.GregorianCalendar(year, month, day).time
            targetFragment?.let{ fragment ->
(fragment as Callbacks).onDateSelected(resultDate)
}
        }
val date = arguments?.getSerializable(ARG_DATE) as Date
val calendar = java.util.Calendar.getInstance()
        calendar.time = date
val initialYear = calendar.get(java.util.Calendar.YEAR)
val initialMonth = calendar.get(java.util.Calendar.MONTH)
val initialDay = calendar.get(java.util.Calendar.DAY_OF_MONTH)
return android.app.DatePickerDialog(
            requireContext(),
            dateListener,
```

```
            initialYear,
            initialMonth,
            initialDay
        )
    }
companion object {
fun newInstance(date: Date): DatePickerFragment {
val args = android.os.Bundle().apply {
putSerializable(ARG_DATE, date)
}
return DatePickerFragment().apply {
arguments = args
}
}
    }
}

package com.bignerdranch.android.criminalintent

import android.content.DialogInterface
import java.util.UUID
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.text.TextUtils
import android.view.LayoutInflater
import android.view.View
import android.widget.Button
import android.widget.RelativeLayout
import com.google.android.material.snackbar.Snackbar
private const val TAG = "MainActivity"
class MainActivity : AppCompatActivity(), CrimeListFragment.Callbacks {
var btnSignIn: Button? = null
    var btnRegister: Button? = null
    var auth: FirebaseAuth? = null
    var db: FirebaseDatabase? = null
    var users: DatabaseReference? = null
    var root: RelativeLayout? = null
    override fun onCreate(savedInstanceState: Bundle?) {
super.onCreate(savedInstanceState)
        setContentView(android.R.layout.activity_main)
btnSignIn = findViewById(android.R.id.btnSignIn)
btnRegister = findViewById(android.R.id.btnRegister)
root = findViewById(android.R.id.root_element)
auth = FirebaseAuth.getInstance()
db = FirebaseDatabase.getInstance()
users = db.getReference("Users")
        btnRegister.setOnClickListener(object : OnClickListener() {
fun onClick(v: View?) {
showRegisterWindow()
            }
        })
}
private fun showSignInWindow() {
val dialog: AlertDialog.Builder = Builder(this)
        dialog.setTitle("Enter")
        dialog.setMessage("Add Data for Enter")
val inflater = LayoutInflater.from(this)
val sign_in_window: View = inflater.inflate(android.R.layout.sign_in_window, null)
        dialog.setView(register_window)
val pass: MaterialEditText = sign_in_window.findViewById(android.R.id.passField)
```

```
val name: MaterialEditText = sign_in_window.findViewById(android.R.id.nameField)
        dialog.setNegativeButton("Back", DialogInterface.OnClickListener { dialogInter-
face, which ->dialogInterface.dismiss() })
        dialog.setPositiveButton("Enter", DialogInterface.OnClickListener { dialogInter-
face, which ->
if (TextUtils.isEmpty(name.getText().toString())) {
                Snackbar.make(root!!, "Enter LogIn", Snackbar.LENGTH_SHORT).show()
return@OnClickListener
}
if (pass.getText().toString().length() <5) {
                Snackbar.make(root!!, "Enter password with min 5 simbols", Snack-
bar.LENGTH_SHORT).show()
return@OnClickListener
}
auth.createUserWithEmailAndPassword(name.getText().to String, pass.getText().to String)
                    .addOnSuccessListener(new OnSuccessListener<AuthResult>{
@Override
public void onSuccess(AuthResult authResult) {
// Успешноедобавление, можетеуказатькакое
startActivity(new Intent(MainActivity this, CrimeListViewModel class));
                            finish();
// полебудетключем, кпримеруполе name
}).addOnFailureListener(new OnFailureListener() {
SnackBar.make(root, "Ошибкаавторизации" +e.getMassege() SnackBar.LENGHT_SHORT).show
}
@Override
public void onFailure(@NonNull Exception e) {
                        }
                    });
})
        dialog.show()
    }
private fun showRegisterWindow() {
val dialog: AlertDialog.Builder = Builder(this)
        dialog.setTitle("Register")
        dialog.setMessage("Enter all Data for Register")
val inflater = LayoutInflater.from(this)
val register_window: View = inflater.inflate(android.R.layout.register_window, null)
        dialog.setView(register_window)
val email: MaterialEditText = register_window.findViewById(android.R.id.emailField)
val pass: MaterialEditText = register_window.findViewById(android.R.id.passField)
val name: MaterialEditText = register_window.findViewById(android.R.id.nameField)
val phone: MaterialEditText = register_window.findViewById(android.R.id.phoneField)
        dialog.setNegativeButton("Back", DialogInterface.OnClickListener { dialogInter-
face, which ->dialogInterface.dismiss() })
        dialog.setPositiveButton("Register", DialogInterface.OnClickListener { di-
alogInterface, which ->
if (TextUtils.isEmpty(email.getText().toString())) {
                Snackbar.make(root!!, "Enter your Email or IP adress", Snack-
bar.LENGTH_SHORT).show()
return@OnClickListener
}
if (TextUtils.isEmpty(name.getText().toString())) {
                Snackbar.make(root!!, "Enter PORT", Snackbar.LENGTH_SHORT).show()
return@OnClickListener
}
if (TextUtils.isEmpty(phone.getText().toString())) {
                Snackbar.make(root!!, "Enter your LOGIN", Snackbar.LENGTH_SHORT).show()
return@OnClickListener
}
```

```
if (pass.getText().toString().length() <5) {
                Snackbar.make(root!!, "Enter password with min 5 simbols", Snack-
bar.LENGTH_SHORT).show()
return@OnClickListener
}
// Регистрацияпользователя
auth.createUserWithEmailAndPassword(email.getText().toString(), pass.getText().toString())
                    .addOnSuccessListener(object : OnSuccessListener<AuthResult?>() {
fun onSuccess(authResult: AuthResult?) {
val user = User()
                            user.setEmail(email.getText().toString())
                            user.setName(name.getText().toString())
                            user.setPass(pass.getText().toString())
                            user.setPhone(phone.getText().toString())
                            users.child(user.name())
                                    .setValue(user)
                                    .addOnSuccessListener(object : OnSuccessListen-
er<Void?>() {
fun onSuccess(aVoid: Void?) {
Snackbar.make(root!!, "Registration complete", Snackbar.LENGTH_SHORT).show()
                                        }
                                    })
                        }
                    })
})
dialog.show()
    }
override fun onCreate(savedInstanceState: Bundle?) {
super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
val currentFragment =
            supportFragmentManager.findFragmentById(R.id.fragment_container)
if (currentFragment == null) {
val fragment = CrimeListFragment.newInstance()
            supportFragmentManager
                .beginTransaction()
                .add(R.id.fragment_container, fragment)
                .commit()
        }
    }
override fun onCrimeSelected(crimeId: UUID) {
val fragment = CrimeFragment.newInstance(crimeId)
        supportFragmentManager
            .beginTransaction()
            .replace(R.id.fragment_container, fragment)
            .addToBackStack(null)
            .commit()
    }
}

package com.bignerdranch.android.criminalintent

fun getScaledBitmap(path: String, activity: android.app.Activity): android.graphics.Bitmap
{
val size = android.graphics.Point()
    activity.windowManager.defaultDisplay.getSize(size)
return getScaledBitmap(path, size.x, size.y)
}

fun getScaledBitmap(path: String, destWidth: Int, destHeight: Int): an-
```

```
droid.graphics.Bitmap {
// Чтениеразмеровизображениянадиске
var options = android.graphics.BitmapFactory.Options()
    options.inJustDecodeBounds = true
android.graphics.BitmapFactory.decodeFile(path, options)
val srcWidth = options.outWidth.toFloat()
val srcHeight = options.outHeight.toFloat()
// Вычисляемнасколькоонужноуменьшить
var inSampleSize = 1
if (srcHeight > destHeight || srcWidth > destWidth) {
val heightScale = srcHeight / destHeight
val widthScale = srcWidth / destWidth
val sampleScale = if (heightScale > widthScale) {
            heightScale
        } else {
            widthScale
        }
        inSampleSize = Math.round(sampleScale)
    }
    options = android.graphics.BitmapFactory.Options()
    options.inSampleSize = inSampleSize
// Чтениеисозданиеокончательногорастровогоизображения
return android.graphics.BitmapFactory.decodeFile(path, options)
}

packagecom.bignerdranch.android.criminalintent
class User {
var name: String? = null
    var email: String? = null
    var pass: String? = null
    var phone: String? = null
constructor() {}
constructor(name: String?, email: String?, pass: String?, phone: String?) {
this.name = name
this.email = email
this.pass = pass
this.phone = phone
    }
}

<?xml version="1.0" encoding="utf-8"?>

<menu xmlns:app="http://schemas.android.com/apk/res-auto"
xmlns:android="http://schemas.android.com/apk/res/android">
xmlns:app="http://schemas.android.com/apk/res-auto">
<item
android:id="@+id/new_crime"
android:icon="@drawable/ic_menu_add"
android:title="@string/new_crime"
app:showAsAction="ifRoom|withText" />
</menu>

<?xml version="1.0" encoding="utf-8"?>
<resources>
<string name="app_name">CriminalIntent</string>
<string name="crime_title_hint">Введитезаказдляклиента</string>
<string name="crime_title_label">Заказ</string>
<string name="crime_details_label">Детали</string>
<string name="crime_solved_label">Решено</string>
<string name="new_crime">Новыйзаказ</string>
```

```
<string name="crime_suspect_text">Выберитеклиента</string>
<string name="crime_report_text">Отправитьотчетклиенту</string>
<string name="app_in_name">Tavria App</string>
<string name="app_name_after">Регистрация</string>
<string name="text_bottom">Введитесвоиидентификационныеданные</string>
<string name="crime_report">%1$s!
The crime was discovered on %2$s. %3$s, and %4$s
</string>
<string name="crime_report_solved">The case is solved</string>
<string name="crime_report_unsolved">The case is not solved</string>
<string name="crime_report_no_suspect">there is no suspect.</string>
<string name="crime_report_suspect">client is %s.</string>
<string name="crime_report_subject">Отчетозаказе</string>
<string name="send_report">Отправитьотчетозаказе via</string>
</resources>

<?xml version="1.0" encoding="utf-8"?>
<paths>
<files-path name="crime_photos" path="."/>
</paths>

dependencies {
    implementation "org.jetbrains.kotlin:kotlin-stdlib:$kotlin_version"
implementation 'androidx.core:core-ktx:1.2.0'
implementation 'androidx.lifecycle:lifecycle-extensions:2.0.0'
implementation 'androidx.recyclerview:recyclerview:1.0.0'
implementation 'androidx.appcompat:appcompat:1.2.0'
implementation "androidx.fragment:fragment-ktx:1.1.0"
implementation 'com.google.android.material:material:1.3.0'
implementation 'androidx.constraintlayout:constraintlayout:2.0.4'
implementation 'androidx.room:room-runtime:2.1.0-alpha04'
implementation 'com.android.support:cardview-v7'
kapt 'androidx.room:room-compiler:2.1.0-alpha04'
testImplementation 'junit:junit:4.+'
androidTestImplementation 'androidx.test.ext:junit:1.1.2'
androidTestImplementation 'androidx.test.espresso:espresso-core:3.3.0'
}
```