

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет

Вычислительной техники

Кафедра

Вычислительная техника

Направление подготовки 09.03.01 «Информатика и вычислительная техника»

Профиль «Вычислительные машины, комплексы, системы
и сети»

БАКАЛАВРСКАЯ РАБОТА

на тему

ПРОГРАММА СКЛАДСКОГО УЧЁТА

ПОД ОС ANDROID

Студент 11.06.2025 Хорошильцев Е.А.
(подпись, дата)

Руководитель 16.06.2025 Токарев А.Н.
(подпись, дата)

Нормоконтролёр 17.6.25 А.С. Бычков
(подпись, дата)

Работа допущена к защите (протокол заседания кафедры от
28.05.2025 г. № 16)

Заведующий кафедрой М.А. Митрохин
(подпись)

Работа защищена с отметкой удовлетворительно (протокол заседания ГЭК от 20.06.25 № 15)

Секретарь ГЭК Гурин Е.И.
(подпись)

Пенза, 2025

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

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

«Утверждаю»
Заведующий кафедрой
 М.А. Митрохин
«06» 05 2025 г.

ЗАДАНИЕ
НА БАКАЛАВРСКУЮ РАБОТУ

1. Студент Хорошильцев Егор Александрович гр.21BBV1 факультета вычислительной техники.

2. Тема работы: «Программа складского учёта под ОС Android»

Тема утверждена приказом ПГУ № 601/о от "06" мая 2025 г.

3. Руководитель бакалаврской работы д.т.н., профессор кафедры ВТ Токарев А.Н.

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

Разработать Android-приложение с использованием архитектуры MVC, обладающее интуитивно понятным интерфейсом и комплексным функционалом для учёта склада.

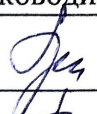
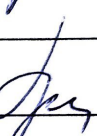
Приложение должно реализовывать следующие функции: предоставление к добавлению новых товаров, редактирование и удаление карточек товаров, предоставление доступа к просмотру карточек товаров, наличие поисковой строки, наличие и использование графика продаж, возможность просмотра категорий. Программа должна обладать интуитивно понятным интерфейсом и использовать следующие технологии:

1. Язык программирования Kotlin;
2. СУБД SQLite.

5. Перечень подлежащих разработке вопросов

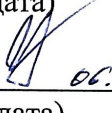
Требуется произвести анализ предметных областей: изучить аналоги существующих приложений, изучить их основные преимущества, изучить разработку приложений на Kotlin и его различный функционал.

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1.	Изучение предметной области	15	06.05.2025-08.05.2025	
2.	Анализ требований	5	09.05.2025-10.05.2025	
3.	Выбор технологий, применяемых при проектировании и разработке	10	11.05.2025-12.05.2025	
4.	Описание алгоритмов	20	13.05.2025-16.05.2025	
5.	Разработка программы на Kotlin, создание графического интерфейса с помощью XML-разметки	60	16.05.2025-28.05.2025	
6.	Оформление пояснительной записки	15	29.05.2025-30.05.2025	

Дата выдачи задания "06" мая 2025 г.

Руководитель бакалаврской работы  06.05.2025 Токарев А.Н.
(подпись, дата)

Задание к исполнению принял студент  06.05.2025 Хорошильцев Е.А.
(подпись, дата)

Работу к защите допустить
Декан факультета  18.06.2025 Фионова Л.Р.
(подпись, дата)

Реферат

Бакалаврская работа содержит 91 лист, 34 рисунка, 2 таблицы, 7 источников и 3 приложения.

ПРОГРАММА СКЛАДСКОГО УЧЁТА ПОД ОС ANDROID, KOTLIN, СУБД, SQLITE

Цель работы – разработать приложение, которое позволит человеку вести учёт и анализ на складском помещении.

Реализуемая программа позволяет пользователям вести учёт склада, а именно: добавлять товары, редактировать и удалять карточки товаров, просматривать карточки товаров, выполнять поиск с помощью поисковой строки, использовать график продаж, а также просматривать категории.

В результате ВКР были выработаны требования к разрабатываемому приложению, спроектирована архитектура, разработано приложение для учёта склада под ОС Android, а также определено направление дальнейшей работы над проектом.

При выполнении ВКР были использованы следующие технологии: ОС Android, СУБД SQLite, язык программирования Kotlin, IDE Android Studio.

					ПГУ 1.090301.21.001 ПЗ			
Изм.	Лист	№ докум.	Подпись	Дата	Программа складского учёта под ОС Android Пояснительная записка	Лит.	Лист	Листов
Разраб.		Хорошильцев Е.А.		11.06.25				
Провер.		Токарев А.Н.		16.06.25				
И. Контр.		Бычков А.С.		17.06.25			2	91
Утверд.		Митрохин М.А.		19.06.25				
						ФВТ, каф. ВТ гр. 21ВВВ1		

Содержание

Перечень сокращений и обозначений	5
Введение	6
1 Программа складского учёта под ОС Android.....	7
1.1 Предпроектный анализ.....	7
1.1.1 Актуальность.....	7
1.1.2 Пользователи и сценарии	7
1.1.3 Анализ аналогов	9
1.1.4 Постановка задачи	16
1.1.5 Выбор языка программирования	18
1.1.6 Выбор IDE	19
1.1.7 Установка и настройка IDE	22
1.1.8 Выбор СУБД	28
1.1.9 Графический интерфейс приложения	29
1.1.10 Функциональные требования.....	30
1.2 Проектирование и разработка	31
1.2.1 Разработка приложения	31
1.2.2 Архитектура пользовательского интерфейса	34
1.2.3 Архитектура программы.....	38
1.2.4 Описание одного из основных методов программы.....	41
1.2.5 Динамика работы системы приложения	42
1.2.6 Описание AndroidManifest.....	43
1.2.7 Описание работы класса DatabaseHelper	45
1.3 Тестирование программы	52
2 Охрана труда	57
2.1 Характеристика потенциального вредоносного влияния при использовании смартфонов	57

2.2 Сводка рекомендаций для минимизации вреда со стороны мобильных устройств	57
2.3 Классификация помещения, в котором происходит работа	58
2.3.1 Эргономические стандарты организации рабочего пространства	59
Заключение.....	61
Список использованных источников.....	62
Приложение А – UML-диаграммы	63
Приложение Б – Листинг программы.....	71
Приложение В – Презентация	86

Перечень сокращений и обозначений

ПО – программное обеспечение

Button – кнопка

CRUD-интерфейс – шаблон проектирования

EditText – компонент для пользовательского ввода, который нужен, если надо отредактировать текст

LinearLayout – макет, выравнивающий все элементы горизонтально, либо вертикально

LineChart – библиотека Jetpack Compose для создания настраиваемых линейных графиков с плавными анимациями. Поддерживает обновления данных в реальном времени, настраиваемый стиль и адаптивный дизайн

ListView – список, который можно прокрутить

Model-view – шаблон проектирования

MVC (Model-View-Controller) – архитектурный шаблон

ScrollView – контейнер, который предназначен для создания прокрутки для такого интерфейса, все элементы которого одновременно не могут поместиться на экране устройства

SearchView – компонент с изображением лупы, при нажатии на себя раскрывается в поле, куда пользователь может вводить названия того, что ищет

Spinner – контейнер, который представляет из себя выпадающий список. В закрытом виде показывает первый элемент списка

TextView – компонент, который отображает заранее написанный текст

Toast - всплывающее уведомление является сообщением, которое появляется на поверхности окна приложения, заполняя необходимое ему количество пространства, требуемого для сообщения. При этом текущая деятельность приложения остаётся работоспособной для пользователя

RelativeLayout – макет, при котором пользователь волен сам распределять, где он хочет расставить элементы

UML (Unified Modeling Language) – унифицированный язык моделирования

					ПГУ 1.090301.21.001 ПЗ	Лист
						5
Изм.	Лист	№ докум.	Подпись	Дата		

Введение

В настоящий момент торговля является актуальным сегментом человеческой жизни, существует множество не только обычных для понимания физических магазинов, которые расположены на улицах и в торговых центрах, но и огромное количество интернет-магазинов и маркет-плейсов. Многими продавцами на таких площадках являются частные, небольшие ИП с маленькими складами. И дабы облегчить себе жизнь они используют простые мобильные приложения для складского учёта.

Современное приложение складского учёта должно включать в себя такие функции:

- управление товарами. Данный пункт включает такие требования, как, например, название товара, его ID, категория и т. д.;
- учёт прихода и расхода – можно указать дату поставки, поставщика, дату списания и т. д.;
- остатки на складе;
- иметь свою базу данных.

К преимуществам использования такого ПО можно отнести шаблонную форму заполнения данных о товарах, графический интерфейс и фиксированное количество пунктов для заполнения.

К недостаткам данного ПО относятся: отсутствие в БД широкой номенклатуры товаров и необходимость обучения сотрудников пользованию данным ПО.

Основной задачей ВКР стало создание мобильного приложения для платформы Android, обеспечивающего простоту использования и оснащенного полным набором функций. Реализуемая программа позволяет пользователям вести учёт склада, опираясь на свои нужды. В ходе выполнения ВКР были использованы умения разработки для мобильных устройств.

Итогом ВКР станет приложение для людей, у которых есть потребность в работе с небольшими складскими помещениями, готовое к использованию.

					ПГУ 1.090301.21.001 ПЗ	Лист
						6
Изм.	Лист	№ докум.	Подпись	Дата		

1 Программа складского учёта под ОС Android

1.1 Предпроектный анализ

1.1.1 Актуальность

Тема «Программа складского учёта под ОС Android» является крайне актуальной на сегодняшний день, так как множеству начинающих предпринимателей необходим подобного рода ПО.

Программы складского учёта – неотъемлемое средство для ведения записей о товарах, материалах и прочих заказах, которые делает предполагаемый заказчик. Ведение вручную подобного может оказаться проблематичным: начиная от того, что заполнение каких-нибудь заметок или условных Excel-таблиц может занимать немало времени, и заканчивая увеличивающейся возможностью критичной ошибки в процессе работы. Программы складского учёта упрощают работу человека со всех сторон в его роде деятельности.

Кроме того, в основном все программы складского учёта обладают своими базами данных, которые не так просто случайно удалить или испортить: они хранятся в корневых папках приложения, так что обычный пользователь не сможет так просто ненарочно их испортить.

В итоге можно сказать, что приложения складского учёта стали неотъемлемой инструментом для работы с заказами предпринимателей: они как были необходимы им, так и продолжают ими быть. Целью данной дипломной работы является разработка и реализация подобного рода приложения, которое будет удовлетворять базовые потребности пользователя. Данная программа должна содержать базу данных для хранения информации о товарах, возможность удаления и создания карточек товаров и так далее.

1.1.2 Пользователи и сценарии

Планируется, что в контингент потребителей, которые будут пользоваться данным приложением, войдут:

– работники малого бизнеса, к которым можно отнести персонал небольших магазинов, складов, сервисных центров и так далее. Зачастую таким

					ПГУ 1.090301.21.001 ПЗ	Лист
						7
Изм.	Лист	№ докум.	Подпись	Дата		

пользователям нужно недорогое, либо вовсе бесплатное решение для менеджмента их товаров;

- индивидуальные предприниматели, в число которых входят, например, фрилансеры, ремесленники и прочие. Таким пользователям в основном не нужны сложные системы для учёта ресурсов;

- сотрудники складов и логистики, у которых нет стабильного доступа к интернету, вынуждены работать в оффлайне.

Пользовательский сценарий для каждой из вышеуказанных мной групп примерно одинаковый, но всё же разберём каждый.

Работники малого бизнеса:

- открытие приложения;
- добавление товаров в базу данных;
- указание даты прихода;
- редактирование данных в случае продаж;
- просмотр остатков после рабочего дня.

Индивидуальные предприниматели:

- открытие приложения;
- внесение, к примеру, новых материалов;
- изменение количества ресурсов, которые были использованы в процессе работы;

- добавление готовых продуктов в базу данных;
- отметить расход готового изделия при продаже.

Сотрудники удалённого склада:

- открытие приложения;
- приём нового груза;
- внесение ресурсов в базу данных;
- если кто-то приезжает забрать материалы, то сотрудник отмечает, сколько было забрано;
- раз в какое-то время проводится инвентаризация.

1.1.3 Анализ аналогов

В рамках дипломной работы можно провести сравнительный анализ различных программ складского учёта на основе их функционала, удобства использования и стоимости подписки.

В таблице 1 представлен анализ аналогов.

Таблица 1 Анализ аналогов

Приложение	Функционал	Удобство использования	Стоимость подписок
«Team: Торговля + Склад + Производство»	Складской учёт, возможность создания разных отчётов и ведомостей, просмотр разных операций, связанных с финансами	Хоть присутствует обучение, но это не делает данное приложение проще в использовании: интерфейс излишне нагруженный и ничуть не интуитивный	Подписка для открытия каких-либо функций не требуется
«Учёт Товаров»	Складской учёт, сканер, возможность добавить отчёт	Приятный и понятный минималистичный интерфейс	Множество разных подписок. Премиум от 199 руб. в месяц
Warehouse Management	Складской учёт, работа в облаке	Минималистичный, но неприятный интерфейс	Доступно только по подписке от 120 долларов в месяц

«Team: Торговля + Склад + Производство» – это комплексное приложение для учёта товаров, услуг, продаж, закупок, производства и бухгалтерии бизнеса на 1С. Имеет множество разных функций, которые не имеются в других приложениях. Интерфейс данного приложения представлен на рисунке 1.

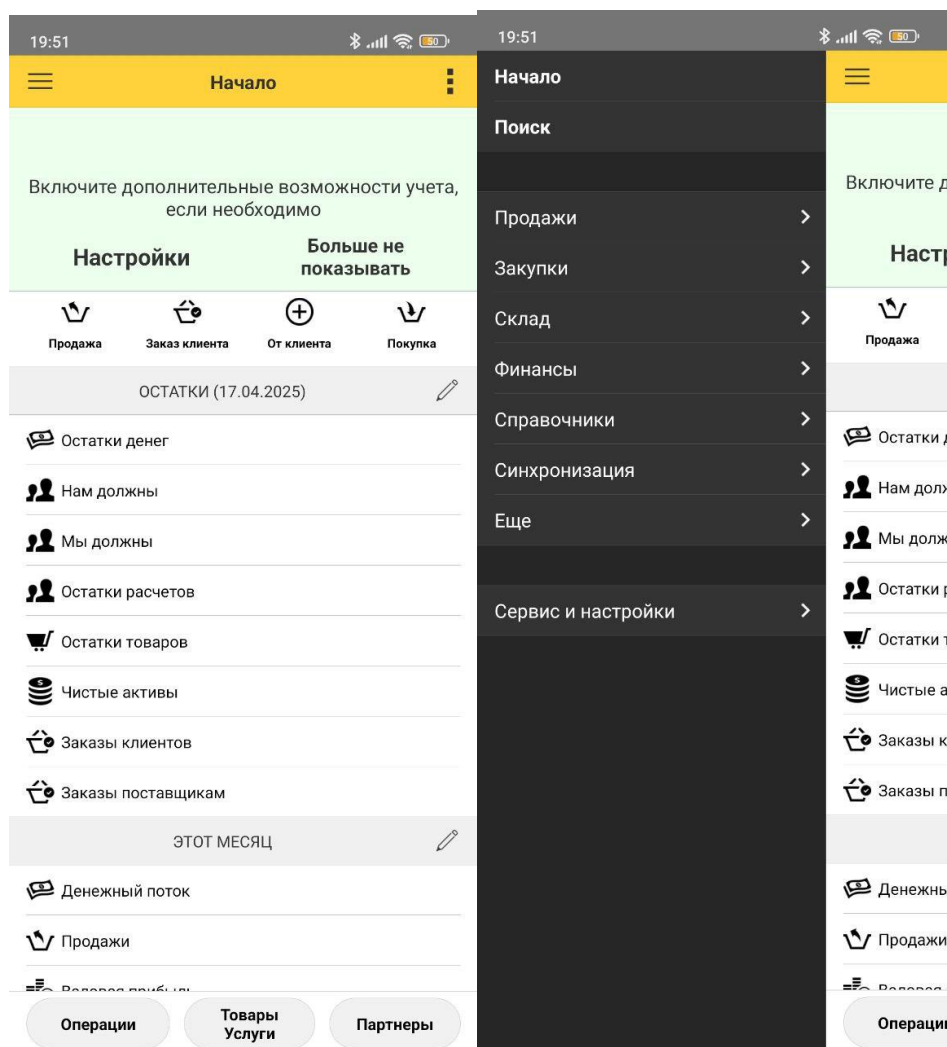


Рисунок 1 – Главное меню первого аналога

«Team: Торговля + Склад + Производство» включает в себя множество возможностей, которые касаются не только склада, но и связанных с другой возможной финансовой деятельностью пользователя. В основном работа в этом приложении представлена в виде таблиц, которые напоминают Excel. В нём доступны просмотр товаров на складе, отчёты по заказам клиентов и поставщиков, разные ведомости, работа с финансами, к примеру, просмотр денежного потока, продаж, валовой прибыли за текущий месяц, возможность просмотра разных

остатков, как денежных, так и товарных. Функционал для работы со складами показан на рисунке 2.

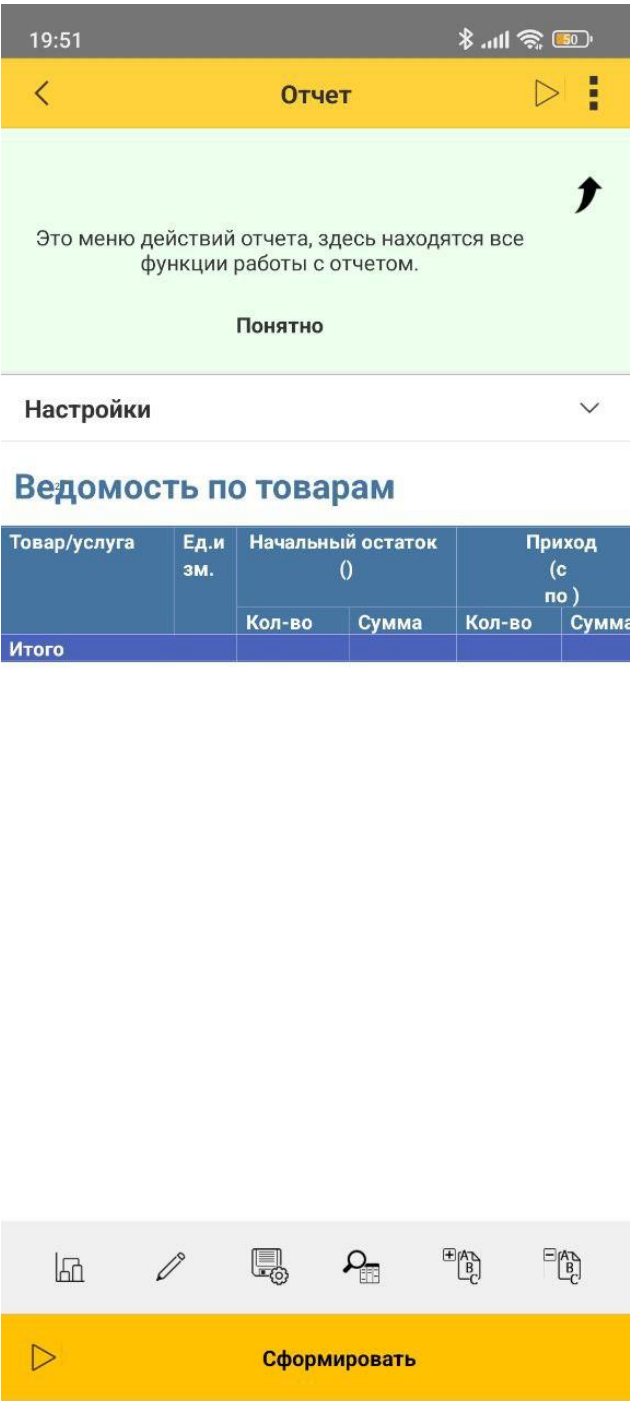


Рисунок 2 – Меню для работы именно со складами первого аналога

Интерфейс данной программы тяжёлый и нагруженный, неподготовленному пользователю будет крайне тяжело разобраться в нём, даже если учесть обучение и подсказки в приложении.

Приложение абсолютно бесплатно.

Преимущества:

- имеет огромное количество возможностей, которые не совсем связаны с простым учётом складов, но могут в теории пригодиться;
- включает в себя обучение и подсказки при работе;
- не требует регистрации;
- бесплатное;
- гибкость, подходит для любого типа бизнеса.

Недостатки:

- тяжесть в освоении;
- табличная форма заполнения.

«Учёт Товаров» – это приложение для учёта товаров для дома, склада или магазина. Содержит в себе возможности для просмотра покупок, продаж и инвентаризации.

В данном приложении также представлен широкий спектр функций для анализа и учёта склада. Эта программа включает в себя возможность ведения учёта товаров, возможности просмотра прихода и расхода, сканер, просмотр затрат, а также создание документов по приходам, расходам и инвентаризации.

Интерфейс данного приложения простой, минималистичный, а главное – интуитивный. Разобраться в нём сможет практически любой пользователь. Все главные возможности размещены на главном экране и имеют иконки.

Приложение включает в себя платные дополнения, которые в основном служат для расширения объёмов данных, которые можно вносить. Подписки могут быть куплены как на месяц, так и на год, а также есть возможность взять «Премиум-подписку» на всю жизнь можно оформить за 4990 рублей. Цена месячной подписки варьируется от 129 до 199 рублей.

Меню данного приложения показано на рисунке 3.

					ПГУ 1.090301.21.001 ПЗ	Лист
						12
Изм.	Лист	№ докум.	Подпись	Дата		

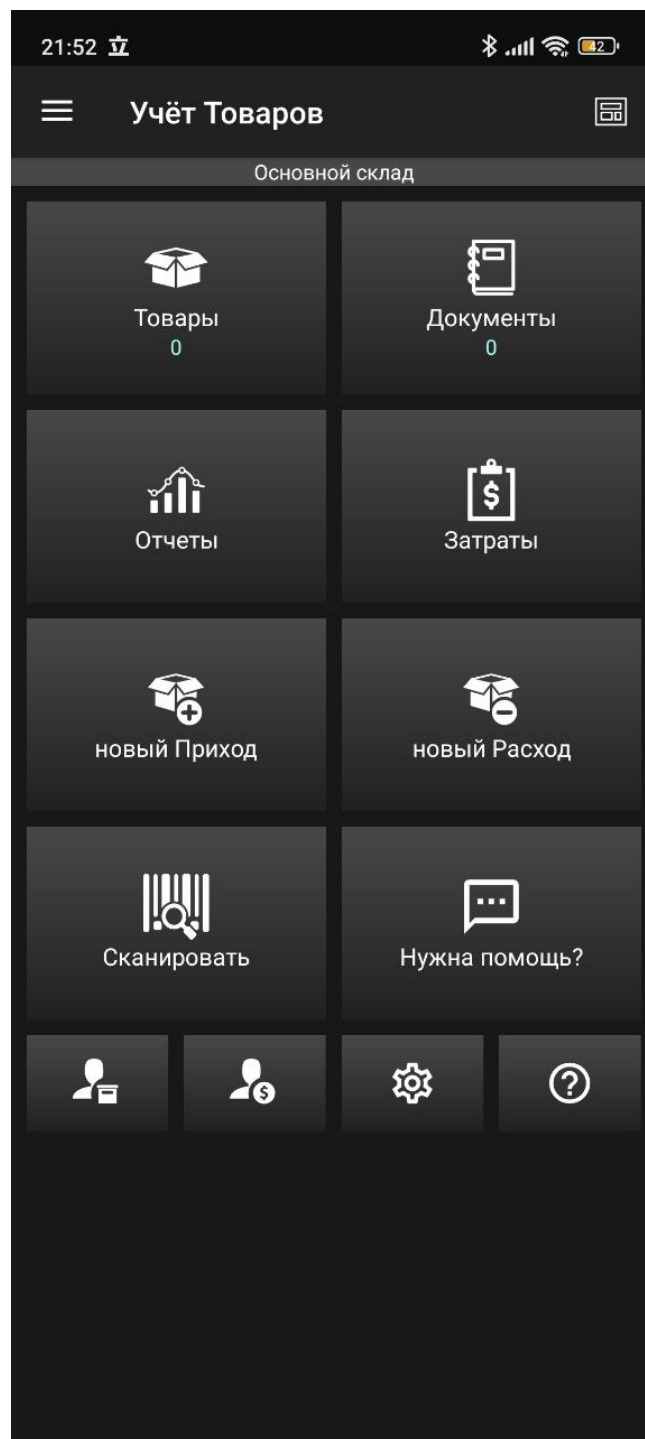


Рисунок 3 – Главное меню «Учёта Товаров»

Преимущества:

- оптимальный спектр возможностей;
- хороший интерфейс;
- не требуется регистрация;
- большинство ключевых функций бесплатны.

Недостатки:

- для кого-то наличие подписок может оказаться минусом;
- судя по отзывам, присутствуют баги.

В итоге можно сказать, что «Учёт Товаров» – крайне удобное и практичное приложение, которое может послужить отличным подспорьем при ведении своего бизнеса.

Warehouse Management – приложение для учёта и анализа склада от компании Microsoft, которое входит в состав Dynamic 365 Supply Chain Management. Предназначено для автоматизации складских процессов.

Warehouse Management поддерживает различные типы складов, может взаимодействовать с другими приложениями из пака Microsoft, возможность ИИ-анализа, автоматические оповещения и большое количество других функций, которыми обычно обладают другие приложения для учёта складов.

С первого взгляда может показаться, что данный минималистичный дизайн интерфейса крайне удобен, но это не совсем так. Хоть главное меню и выглядит вполне органично, но разобраться в каждой из функций данного приложения не совсем просто.

Warehouse Management – полностью платна, без подписки на Dynamic 365 Supply Chain Management ни одна функция приложения не может быть использована. Стоимость подписки составляет около 120 долларов в месяц.

Преимущества:

- качественная экосистема вместе с другими приложениями компании;
- гибкость, подходит для любого бизнеса;
- поддержка голосовых технологий Hands-Free Warehouse.

Недостатки:

- высокая стоимость подписки;
- возможно, ненужные сервисы Microsoft;
- ориентировано больше на корпоративный сегмент.

Интерфейс «Warehouse Management» показан на рисунке 4.

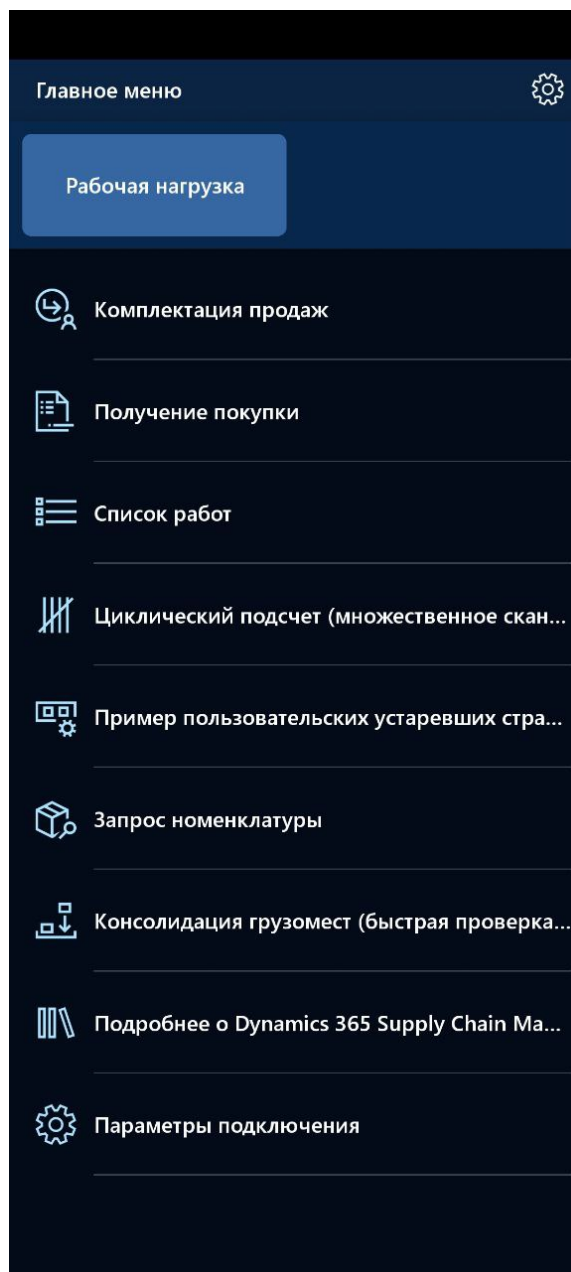


Рисунок 4 – Главное меню Warehouse Management

В итоге можно сказать, что каждая из перечисленных мною программ имеет свои какие-то ключевые особенности, которые отличают каждую друг от друга. Данные программы хоть и могут использоваться для одних и тех же задач, но всё же спектр деятельности у них разный.

Например, приложение от компании Microsoft подходит больше участникам корпораций не только из-за огромнейшей стоимости подписки, без которой приложение окажется мёртвым грузом на устройстве, но и к функционалу, и связанным с ним другими программами, которые входят в эту подписку.

					ПГУ 1.090301.21.001 ПЗ	Лист
						15
Изм.	Лист	№ докум.	Подпись	Дата		

«Team: Торговля + Склад + Производство» подходит больше тем, кому важно отслеживать полный цикл работы с товаром, начиная от закупки сырья и продажи готового товара: данное приложение содержит огромный функционал по ведению разнообразных отчётов и ведомостей. Оно вполне может быть использовано небольшими ИП, но зачастую им не нужен такой грузный функционал.

В любом случае при выборе приложения нужно обязательно оценивать функционал приложения, удобство использования, стоимость подписок.

1.1.4 Постановка задачи

Требуется разработать программу для учёта склада, которая будет соответствовать следующим функциональным и техническим требованиям.

Требуемые основные функции:

- добавление товаров в БД;
- просмотр карточек товаров;
- удаление карточек товаров;
- редактирование карточек товаров;
- поиск по названию;
- просмотр категорий товаров.

Графический интерфейс, который содержит в себе карточки товаров, кнопки для работы с карточками товаров.

Технические требования:

- основной платформой, под которую разрабатывается приложение, является ОС Android;
- минимальная версия Android для работы приложения – 12.0;
- необходимо 30 мегабайт свободного места для установки приложения;
- сохранение данных происходит автоматически при добавлении новых товаров в БД;

Возможности пользователя показаны на рисунке 5.

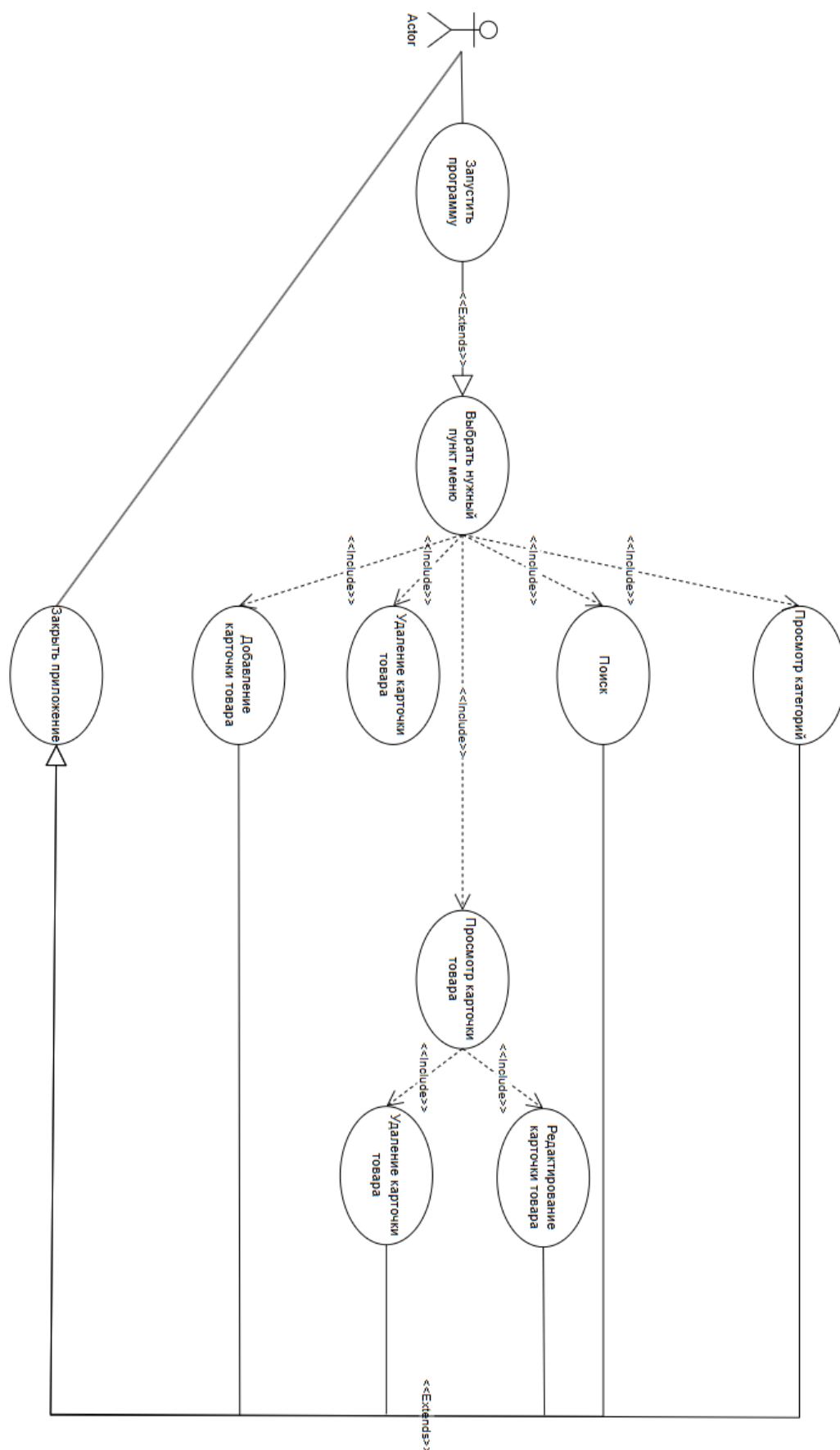


Рисунок 5 – UML-Диаграмма вариантов использования

1.1.5 Выбор языка программирования

Для разработки мобильного приложения существуют несколько языков программирования, но ключевыми являются Java, Kotlin, а также C# с фреймворком .NET MAUI.

Рассмотрим данные языки, выделим их плюсы и минусы.

Java – один из самых популярных языков программирования в мире, занимает четвёртое место в списке популярности именитого сайта tiobe.com. Несмотря на свой возраст, Java обладает огромными возможностями, большим количеством всевозможных библиотек, используется как для разработки десктопных приложений, так и мобильных. Место в списке популярности показано на рисунке 6.

Apr 2025	Apr 2024	Change	Programming Language		Ratings	Change
1	1			Python	23.08%	+6.67%
2	3	▲		C++	10.33%	+0.56%
3	2	▼		C	9.94%	-0.27%
4	4			Java	9.63%	+0.69%
5	5			C#	4.39%	-2.37%
6	6			JavaScript	3.71%	+0.82%
7	7			Go	3.02%	+1.17%
8	8			Visual Basic	2.94%	+1.24%
9	11	▲		Delphi/Object Pascal	2.53%	+1.06%
10	9	▼		SQL	2.19%	+0.57%

Рисунок 6 – Список популярности языков программирования на 2025 год

Достоинства:

- простота в использовании и изучении;
- огромное количество библиотек и фреймворков.

Недостатки:

- многословный синтаксис;
- требует ручного управления ресурсами.

Kotlin – язык программирования, основанный на Java и вместе с тем совместимый с ней. Хотя и не входит в топ популярных языков программирования, но вместе с тем больше всех используется для создания мобильных приложений.

Плюсы:

- может использоваться вместе с Java в одном проекте;
- лаконичный синтаксис, код не такой объёмный и нагруженный, как в Java.

Минусы:

- куда более сложен в освоении;
- может использоваться только под ОС Android.

C# + .NET MAUI/Xamarin – объектно-ориентированный язык программирования от компании «Майкрософт». Также является одним из популярнейших языков программирования в мире, занимая место после Java. Xamarin – фреймворк для разработки мобильных приложений под IOS и Android, появившийся в 2011 году и использовавшийся до 2022 года. .NET MAUI – апгрейд Xamarin, начавший свою работу с .NET 6+.

Плюсы:

- кроссплатформенность: используется один код для iOS, Android, Windows и macOS;
- отлично взаимодействует с другими программами от «Майкрософт» как, например, Azure, SQL Server.

Минусы:

- большой вес приложений;
- некоторая программная сырость, в сравнении с Kotlin.

Каждый из этих языков программирования имеет свои преимущества и недостатки, но в любом случае выбор стоит лишь за определёнными нуждами разработчика. Для проекта подходит Kotlin.

1.1.6 Выбор IDE

Интегрированная среда разработки (IDE) – это комплекс всевозможных программных средств, используемых программистами для создания и отлаживания программного обеспечения.

В основном IDE включает в себя следующие обязательные возможности:

- текстовый редактор;

- компилятор и/или интерпретатор;
- средства автоматизации сборки.

Существует большое количество разных IDE для разработки на Kotlin под ОС Android, рассмотрим лучшие из них: Android Studio, IntelliJ IDEA, Visual Studio Code.

Android Studio – интегрированная среда разработки для работы с платформой Android, анонсированная 16 мая 2013 года на одной из конференций Google. Данная IDE основана на IntelliJ IDEA от компании JetBrains. Kotlin стал официальным языком программирования Android Studio в 17 мая 2017 года.

К плюсам данной IDE можно отнести:

- официальная поддержка от Google;
- встроенный графический редактор макетов для создания интерфейсов;
- создана специально для разработки приложений под ОС Android.

К минусам:

- медленная работа и запуск;
- если нет Android-устройства под рукой, то придётся использовать виртуальные устройства для теста приложений, с каждым новым запуском Android Studio создаёт новый образ виртуального устройства, который занимает место на диске C, возможности изменить место хранения нет;
- потребляет довольно много оперативной памяти.

IntelliJ IDEA – интегрированная среда разработки от компании JetBrains. Даёт возможность работы с многими языками программирования, включая Kotlin, Java, JavaScript и другие. Впервые появилась в 2001 году и получила большую популярность из-за возможности перепроектирования кода, процесса, позволяющего изменить внутреннюю структуру программы так, чтобы облегчить её понимание и не спровоцировать изменения поведения.

Плюсы:

- крайне универсальная IDE с возможностью поддержки большой группы популярных языков программирования;
- можно настроить на разработку Android приложений;

- возможность перепроектирования кода куда лучше, чем в Android Studio.

Минусы:

- отсутствие Android Profiler, возможности анализировать состояние устройства во время работы разрабатываемого приложения;
- отсутствие встроенных шаблонов для разработки Android приложений.

Visual Studio Code – это, скорее, продвинутый текстовый редактор, нежели полноценная интегрированная среда разработки. Была разработана компанией Microsoft, классифицируется как редактор кода для кроссплатформенной разработки, впервые выпущена 14 ноября 2015 года. Распространяется бесплатно, исходный код открыт. Изначально не включает никаких инструментов для разработки, но есть множество расширений, которые делают из данного ПО практически полноценную IDE.

Достоинства:

- лёгкая и быстрая;
- гибкость, обусловленная поддержкой разных расширений, которые позволяют работать с разными языками программирования, включая Kotlin;
- огромное комьюнити, которое и создаёт разноразличные плагины и расширения.

Недостатки:

- поддержка Android слабая;
- требуется ручная настройка, достаточно проблематично разобраться, если используется впервые;
- отсутствует графический редактор, как и в случае с IntelliJ IDEA;
- проблематичная отладка.

В итоге можно сказать, что каждый разработчик выбирает IDE, исходя из своих потребностей. Для проекта подходит Android Studio, так как она больше всего подходит для написания приложений под ОС Android, а вместе с тем содержит встроенный графический редактор для создания интерфейсов.

1.1.7 Установка и настройка IDE

Системные требования Android Studio приведены в таблице 2.

Таблица 2 – Системные требования Android Studio для ОС Windows

Требования	Минимальные	Рекомендуемые
ОС	Windows 10 64-bit	Windows 10-11 64-bit
ОЗУ	8 гигабайт для самой среды разработки 16 гигабайт для среды разработки и встроенного эмулятора	32 гигабайта
Процессор	Процессоры Intel 8th Gen Core i5/ AMD Zen Ryzen	Процессоры Intel и AMD последнего поколения
Свободное место на диске	8 гигабайт для самой среды разработки 16 гигабайт для среды разработки и встроенного эмулятора	32 гигабайта и больше
Видеокарта	Для среды разработки видеокарта необязательна Для среды разработки и эмулятора – видеокарты Nvidia GeForce 10 серии или видеокарты серии AMD RX 50xx от 4 гигабайт видеопамяти	Видеокарты с 8 гигабайтами видеопамяти серии от Nvidia GeForce 20xx или AMD RX 66xx

Для установки ПО нужно зайти на официальный сайт Android Studio, а затем начать скачивание установщика. Место скачивания Android Studio показано на рисунке 7.

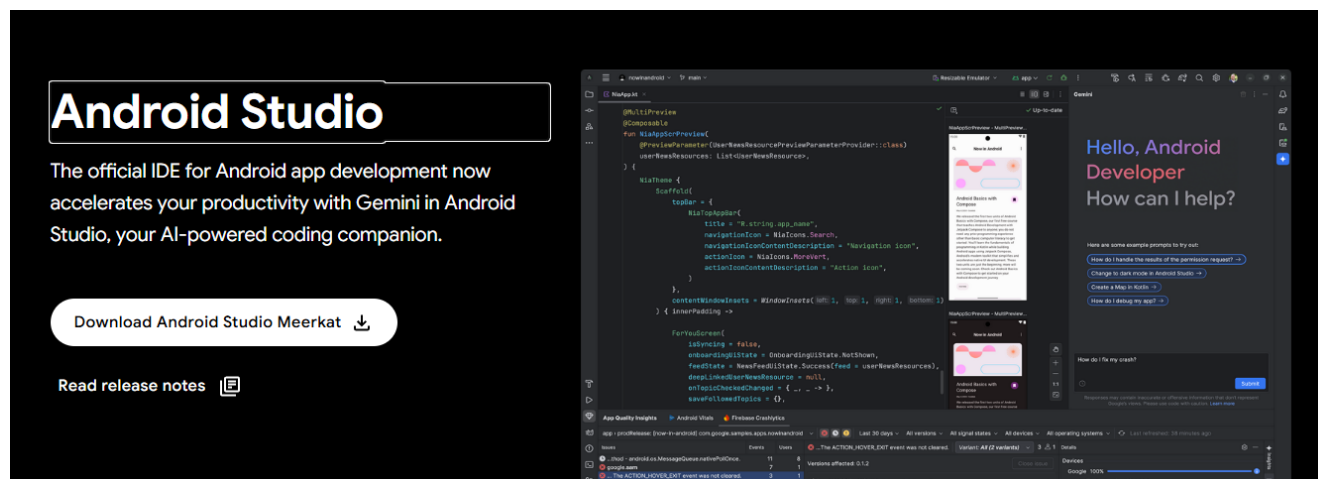


Рисунок 7 – Окно со ссылкой на скачивание установщика

Далее нужно выбирать практически всё по умолчанию, не считая момента, когда установщик просит выбрать место, куда будет установлена Android Studio. Стадии установки программы приведены на рисунках 8 и 9.



Рисунок 8 – Начальный экран установщика

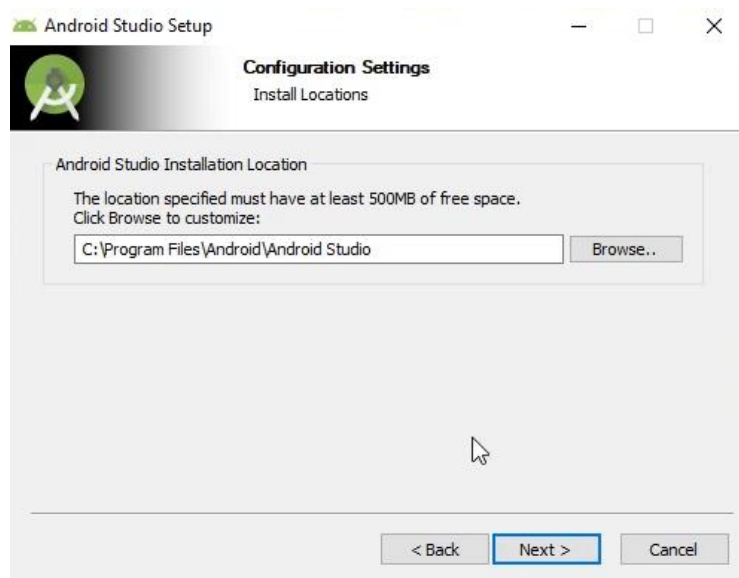


Рисунок 9 – Выбор места, куда будет установлена программа

Нужно выбирать тему, которая будет стоять при использовании Android Studio. Выбор темы показан на рисунке 10.

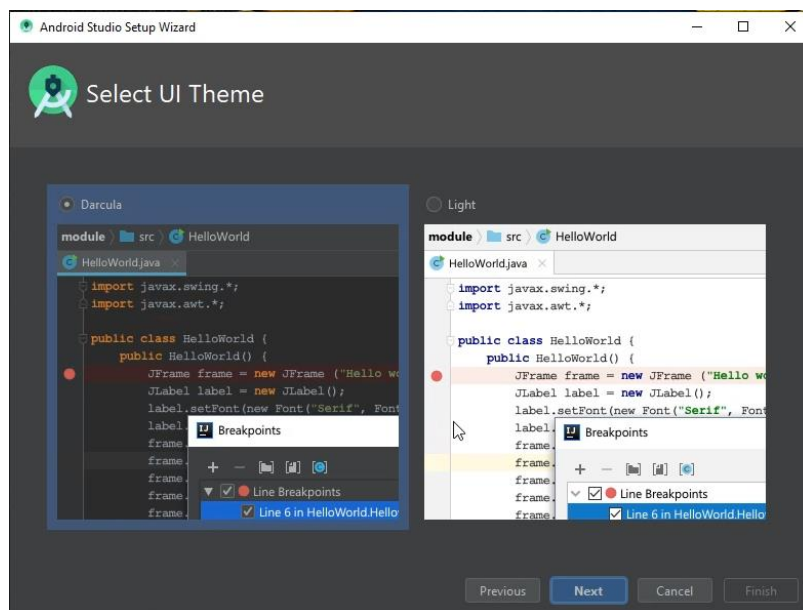


Рисунок 10 – Выбор будущей темы: тёмной или светлой

Далее нужно нажать Finish, и установщик скачает дополнительные пакеты с серверов Google.

Затем, при открытии Android Studio, нужно зайти в настройки и поставить галочки около нужных SDK. SDK – это набор инструментов для разработки программ под конкретную ОС. Настройки показаны на рисунке 11.

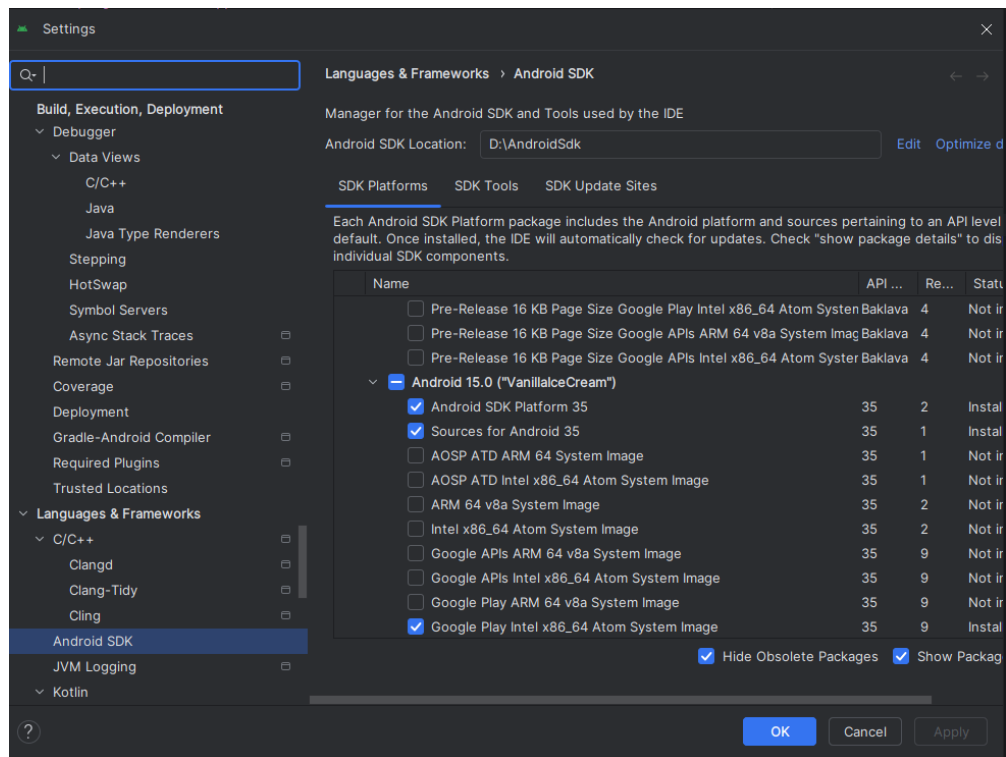


Рисунок 11 – Выбор SDK

Затем можно создать свой первый проект, так что нужно выбрать его тип. Выбор типа проекта показан на рисунке 12.

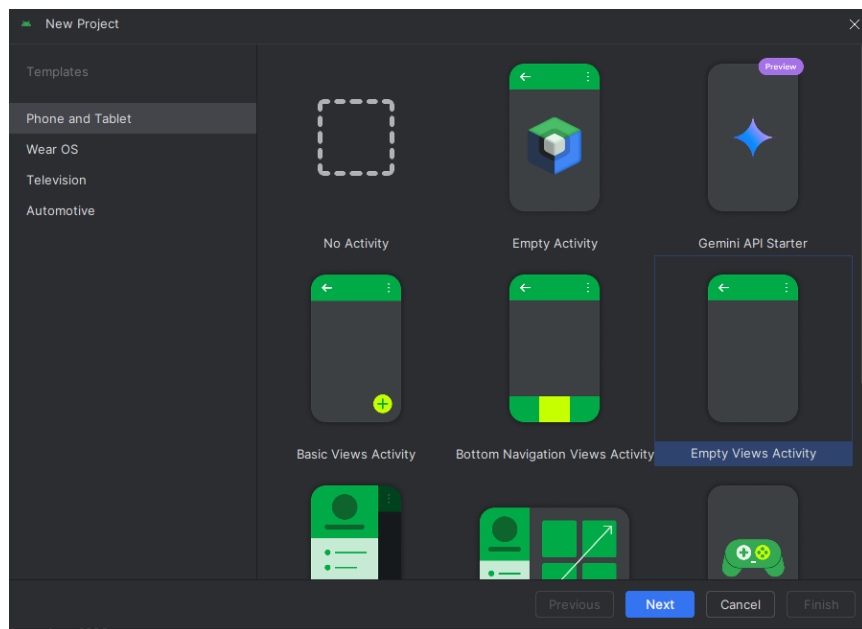


Рисунок 12 – Выбор типа проекта

После нужно задать имя проекта и выбрать нужный язык: Kotlin, либо Java, и минимальную SDK. Создание нового проекта показана на рисунке 13.

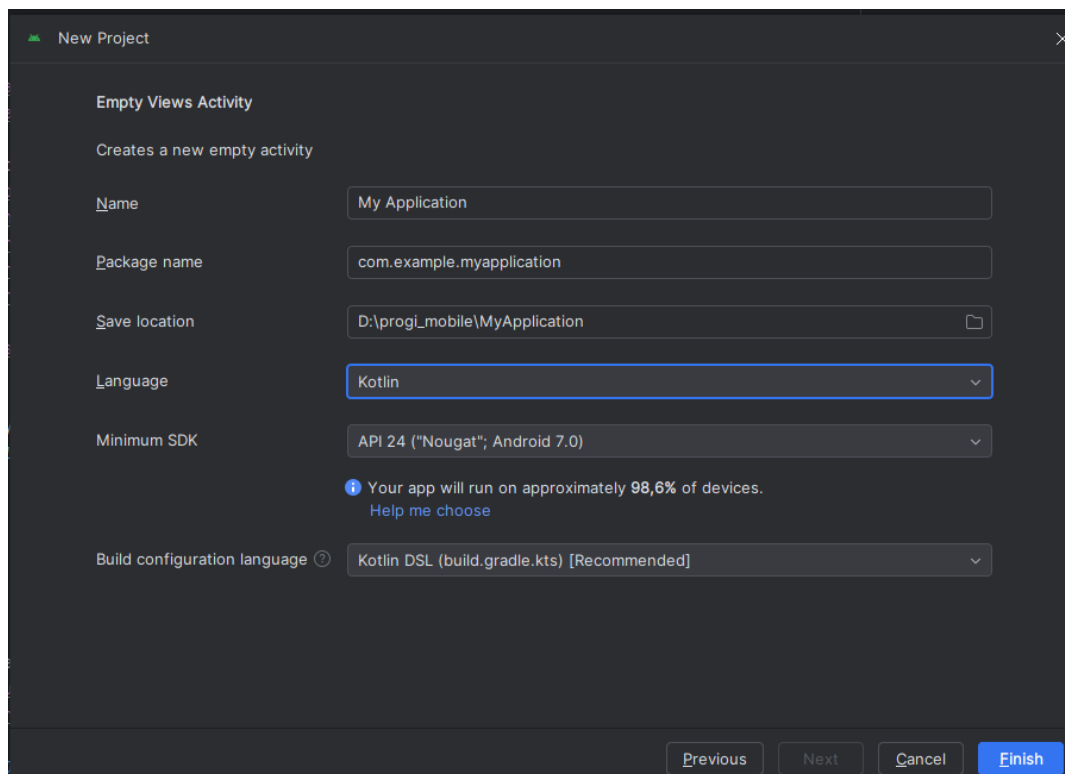


Рисунок 13 – Создание первого проекта

После нажатия кнопки Finish начнётся относительно долгая загрузка Gradle. Gradle – это система автоматической сборки приложений, она собирает все необходимые файлы, библиотеки и прочее воедино для создания исполняемого файла приложения, появляется только в первый раз при создании приложения. Загрузка Gradle показана на рисунке 14.

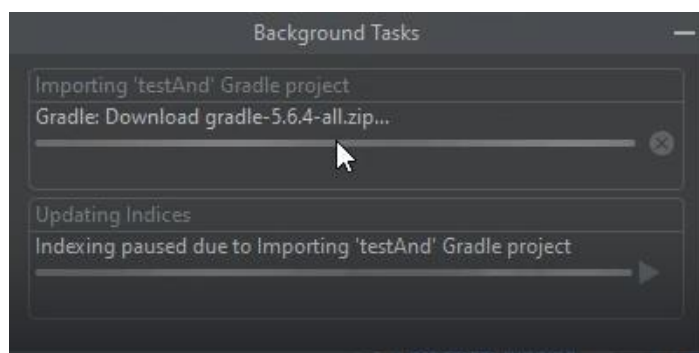


Рисунок 14 – Загрузка Gradle

Также во вкладке Device Manager, можно добавить устройство: виртуальное, либо физическое, которое можно подключить по проводу к ПК. Выбор устройства показан на рисунке 15.

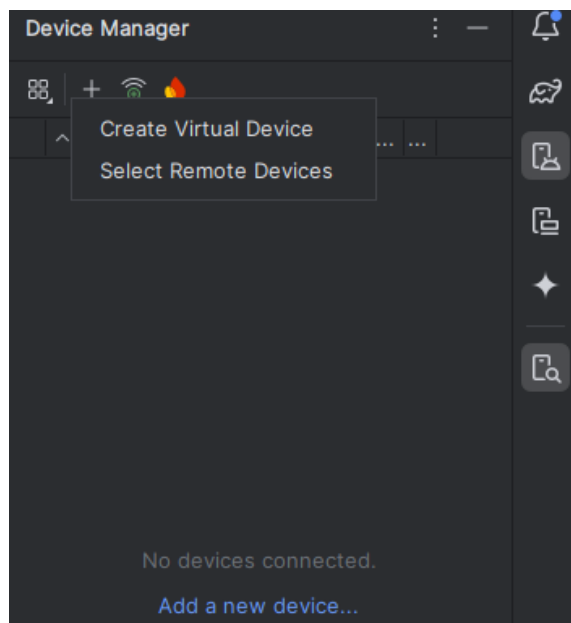


Рисунок 15 – Выбор устройства

При выборе виртуального устройства нужно определиться с тем, какое устройство будет эмулироваться. Выбор эмулируемого устройства показан на рисунке 16.

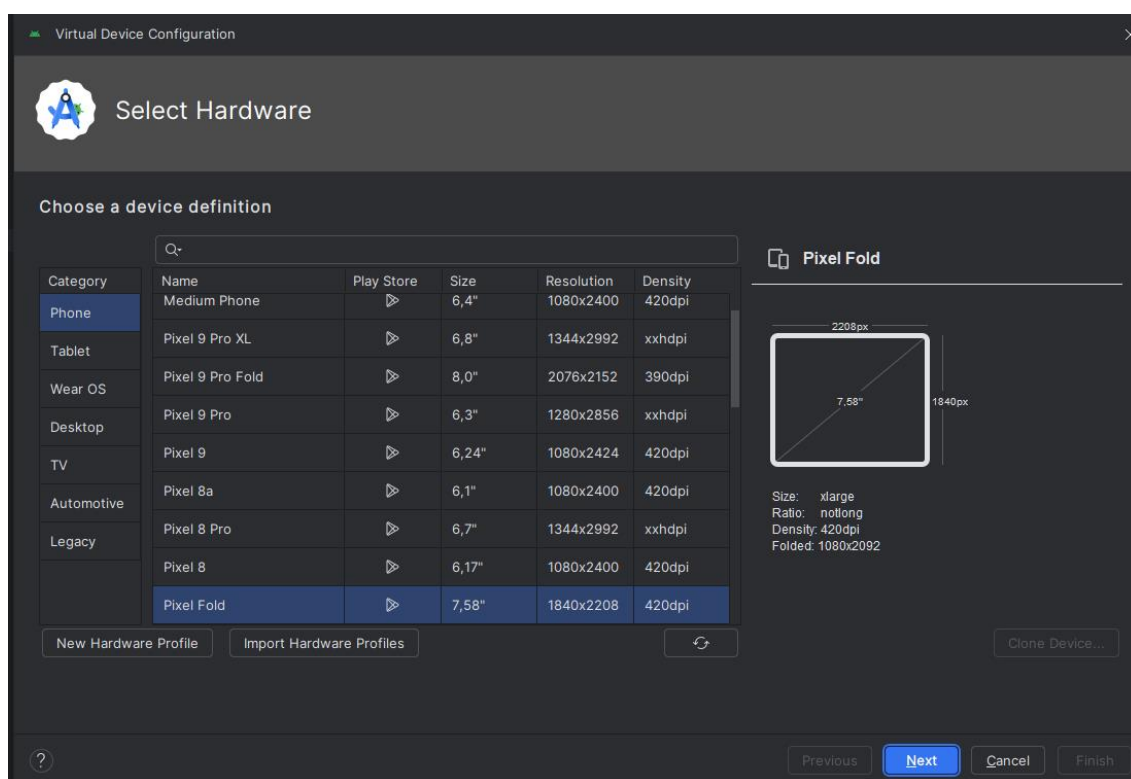


Рисунок 16 – Выбор эмулируемого устройства

1.1.8 Выбор СУБД

На данный момент существует две разновидности баз данных для хранения различной информации: реляционная и нереляционная. Если выражаться простым языком, то в реляционной БД данные хранятся в таблицах, связанных между собой, а в нереляционных БД они содержатся в неструктурированном порядке. Для управления содержанием первой разновидности используется язык SQL, а для второй – NoSQL.

К достоинствам реляционной БД можно отнести:

- чёткую структуру;
- возможность написания сложных запросов.

К недостаткам:

- проблемность в плане добавления новых полей, для этого придётся менять всю таблицу;
- медленную работу при больших массивах данных.

К плюсам нереляционной БД можно отнести:

- гибкость, возможность добавления новых полей;
- высокую скорость работы.

К минусам относится отсутствие таких связей, как в реляционной.

В любом случае выбор базы данных зависит сугубо от требований разработчика. Для проекта выбрана реляционная БД.

Теперь рассмотрим некоторые SQL системы управления баз данных: SQLite, H2 и PostgreSQL.

SQLite – компактная встраиваемая СУБД, не имеет сервера, а значит, что все данные хранятся локально.

Плюсы:

- высокая скорость работы;
- малый вес;
- встроена в Android.

Минусы:

- отсутствие возможности работы на нескольких устройствах;

- ограничение работы с большими данными.

H2 – кроссплатформенная встраиваемая СУБД, написанная на Java. К её достоинствам можно отнести, что она, в отличие от SQLite, поддерживает SQL полностью, может быть совместима с другими популярными СУБД, например, с популярной MySQL, которой, правда, нет на Android. Также возможен доступ к данным СУБД через браузер. Несмотря на свои плюсы, H2 имеет и значительные минусы: официально она не поддерживается на Android, а вместе с тем обладает небольшим количеством библиотек для данной платформы, имеет меньшую производительность, в сравнении с SQLite.

PostgreSQL – тяжёлая СУБД, подходит для всяких сложных задач, где требуется надёжность, огромный функционал и работа с большими объёмами данных, например, в банковской сфере. Поддерживает все функции SQL с расширениями, надёжная, возможны обращения к JSON, а также можно добавлять свои типы данных. Данная СУБД требует сервер и без него работать не может, обладает крайне сложными настройками, а вместе с тем её функционал избыточен для относительно простых задач.

В итоге можно сказать, что для проекта подходит именно SQLite: она встроена в Android, не обладает чрезмерным количеством функций и возможностей, которые не нужны для работы программы, имеет высокое быстродействие. Данная СУБД полностью удовлетворяет требования проекта.

1.1.9 Графический интерфейс приложения

Графический интерфейс приложения реализован с помощью разметки XML, для неё есть встроенный в Android Studio визуальный редактор под названием Layout Editor. По функционалу он похож на редактор проекта типа приложение Windows Forms, который реализован в Visual Studio, но куда менее удобный, не такой быстрый в плане выполнения задач, а также есть потребность в обязательном мобильном устройстве для теста проекта, в противном случае Android Studio будет создавать образы устройства, которые весят в районе 2–3 гигабайт, при каждой новой отладке создаваемого приложения.

В любом случае создание графического интерфейса с помощью XML- разметки подходит для освоения новичками из-за своей простоты и нужных возможностей.

1.1.10 Функциональные требования

Требования к программе складского учёта под ОС Android на основе пользовательских сценариев, разработанных ранее, приведены ниже:

- приложение должно предоставлять доступ к добавлению карточек товаров;
- приложение должно позволять редактировать и удалять карточки товаров;
- приложение должно позволять просматривать карточки товаров;
- приложение должно обладать поиском для нахождения товаров по названию;
- приложение должно обладать графиком продаж или использования, который позволит пользователю оценивать затраты, либо продажи своих товаров;
- приложение должно обладать категориями для быстрого поиска определённых товаров.

Выводы по предпроектному анализу:

- показана актуальность выбранной темы;
- описаны основные типы пользователей;
- описаны функциональные требования к разрабатываемому приложению;
- произведён анализ и выбраны инструменты для разработки.

1.2 Проектирование и разработка

Программа складского учёта состоит из БД и клиентского приложения. База данных хранит записи о товарах пользователя, а клиентское приложение служит посредником между пользователем и ней.

Приведённое на рисунке 17 программное обеспечение позволяет воспользоваться полным функционалом программы.

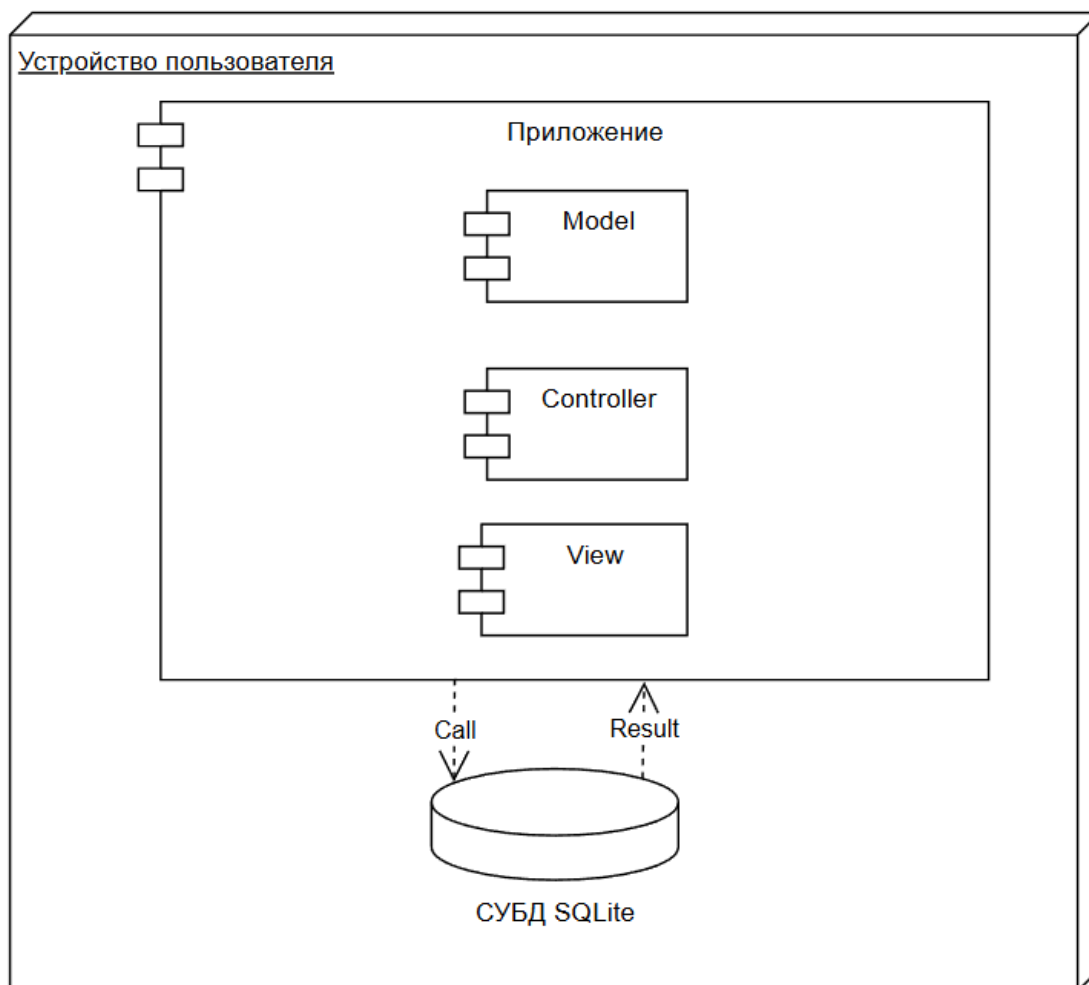


Рисунок 17 – UML-диаграмма развертывания

1.2.1 Разработка приложения

Для создания приложения был выбран язык Kotlin. В меню был выбран шаблон Empty Views Activity. Выбор шаблона для разработки показан на рисунке 18.

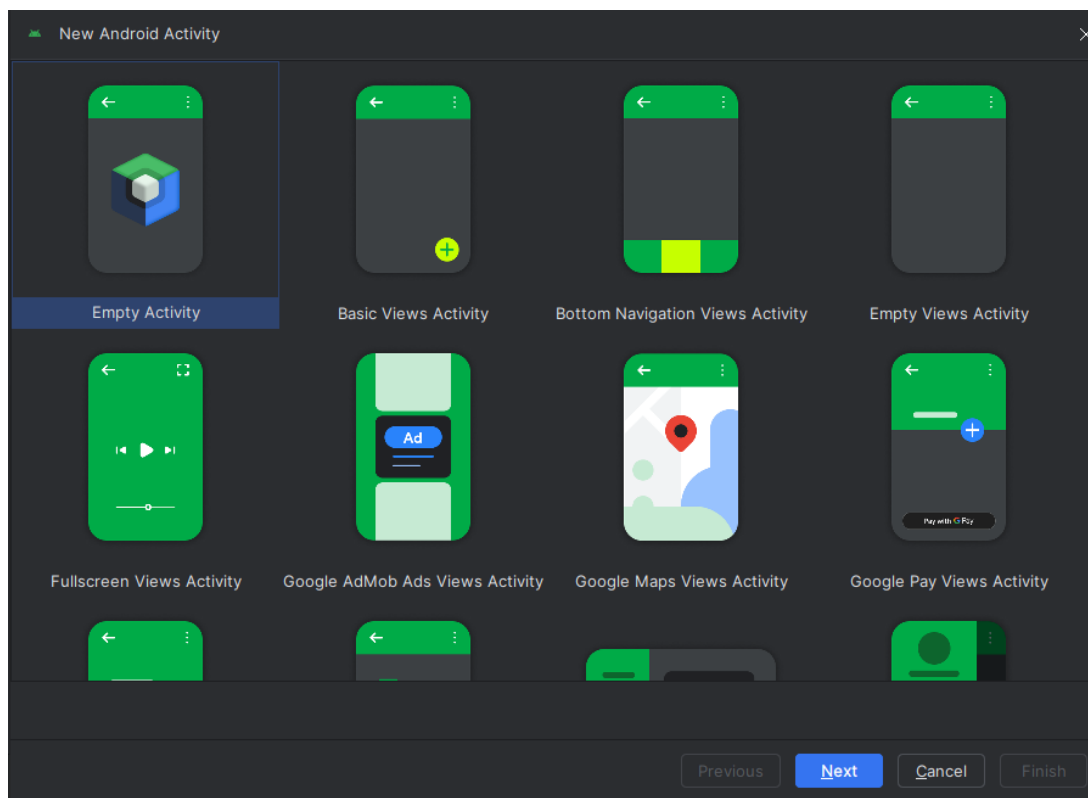


Рисунок 18 – Выбор шаблона для разработки

Принципы проектирования являются ключевыми при разработке программы. Они помогают структурировать приложение, упрощают его разработку и последующее обновление, а вместе с тем помогают эффективно разделить работу на несколько человек.

Шаблон проектирования MVC предполагает разделение данных приложения, пользовательского интерфейса и управляющей логики на три отдельных компонента: модель, представление и контроллер – таким образом, что модификация каждого компонента может осуществляться независимо. Модель (Model) предоставляет данные предметной области представлению и реагирует на команды контроллера, изменяя свое состояние. Представление (View) отвечает за отображение данных предметной области (модели) пользователю, реагируя на изменения модели. Контроллер (Controller) интерпретирует действия пользователя, оповещая модель о необходимости изменений.

В качестве View в программе выступает XML-разметка, которая выполняет роль отображения интерфейса. Код одной из кнопок представлен ниже.

```
<Button
    android:id="@+id/buttonAddProduct"
    android:layout_marginTop="8dp"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Добавить товар" />
```

В роли Controller выступают классы Activity, они отвечают за обработку действий пользователя, принятие решений. Они совершают вызовы Model для работы с данными. Пример одной из функций MainActivity приведён ниже.

```
private fun confirmAndDeleteProduct(productId: Int) {
    AlertDialog.Builder(this)
        .setTitle("Удаление товара")
        .setMessage("Вы уверены, что хотите удалить этот товар?")
        .setPositiveButton("Удалить") { _, _ ->
            if (dbHelper.deleteProduct(productId.toLong())) {
                Toast.makeText(this, "Товар удален",
                    Toast.LENGTH_SHORT).show()
                loadAllProducts()
            } else {
                Toast.makeText(this, "Ошибка при удалении",
                    Toast.LENGTH_SHORT).show() }
        }.setNegativeButton("Отмена", null)
        .show()
}
```

За Model отвечает класс DatabaseHelper. Он не зависит от классов View и Controller, обеспечивает работу с данными. Пример кода одного из методов приведён ниже.

```
// Метод для получения товара по ID
fun getProductById(id: Int): Cursor {
    val db = this.readableDatabase
    return db.rawQuery("SELECT * FROM $TABLE_WAREHOUSE WHERE
    $COLUMN_ID = ?", arrayOf(id.toString()))}
```

UML-диаграмма взаимодействия одно из классов показана на рисунке 19.

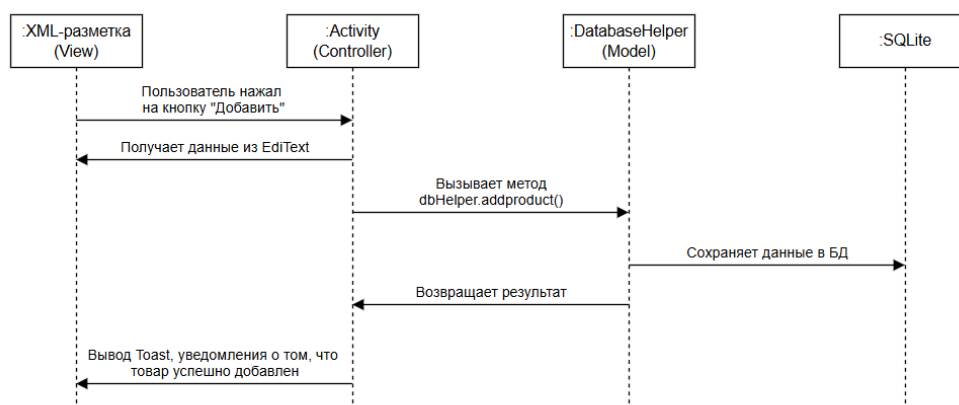


Рисунок 19 – UML-диаграмма взаимодействия

1.2.2 Архитектура пользовательского интерфейса

Для создания пользовательского интерфейса была использована XML-разметка. В приложении применены стандартные шаблоны Material Design, которые делают интерфейс интуитивным и понятным для пользователя. Material Design — это система для создания дизайна интерфейсов, которую разработала компания Google. В данной системе содержатся рекомендации по визуальному, интерактивному и динамическому оформлению элементов ПО для различных устройств и платформ.

В приложении реализованы такие паттерны проектирования пользовательского интерфейса, как:

- Mater-detail. Суть данного паттерна заключается в том, что есть две области: Master и detail. Master – главная область, где расположен список элементов, а detail – область, в которой находится информация об конкретном элементе;

- CRUD-интерфейс. CRUD – аббревиатура, обозначающая четыре основных операции управления данными: create, read, update, delete/destroy, то есть создание, чтение, обновление и удаление. Это действия, которые мы совершаем с любой информацией, в любых системах: на сайтах, в приложениях, базах данных.

Примеры шаблонов показаны на макетах, которые приведены на рисунках 20 и 21.

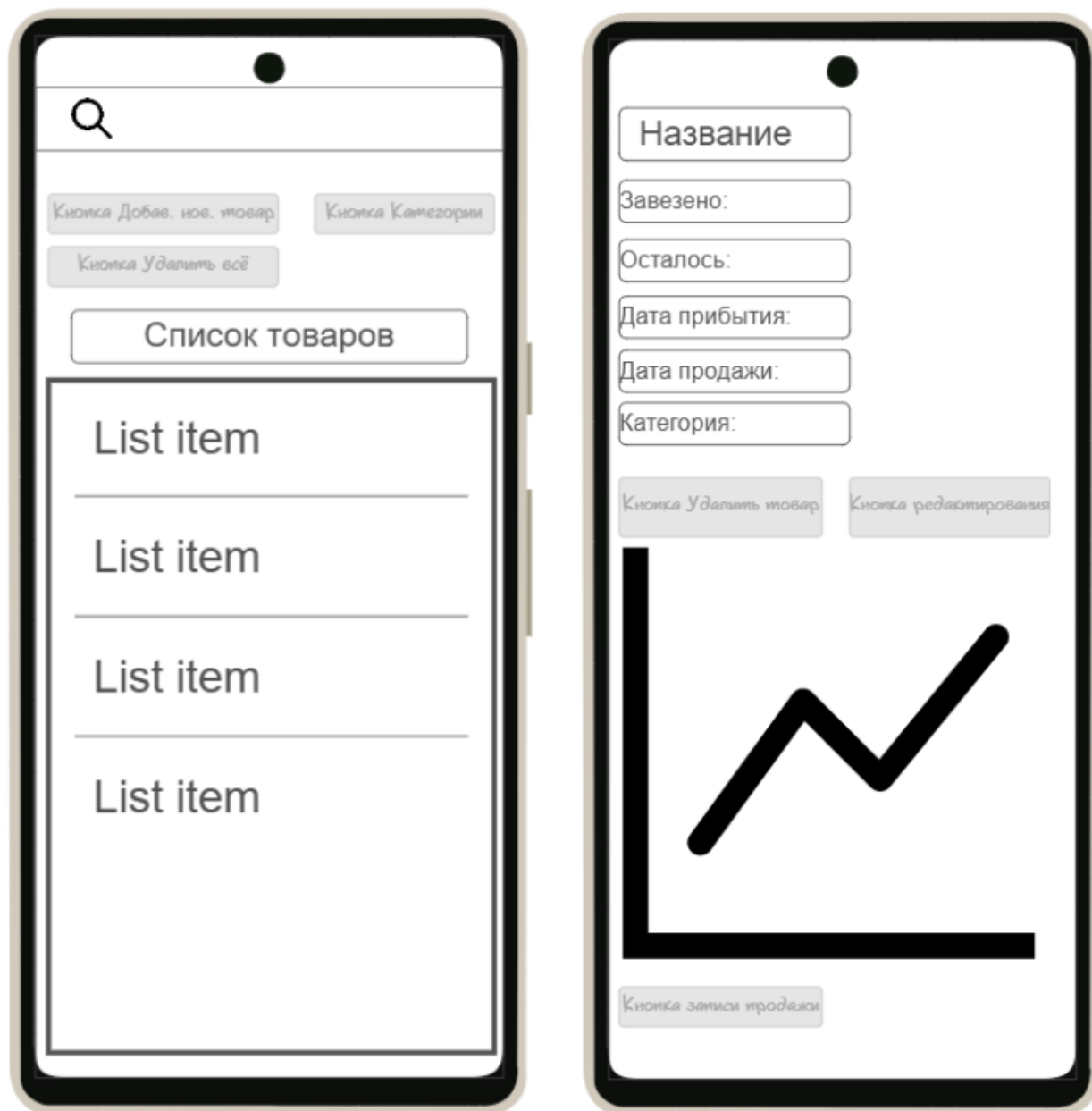


Рисунок 20 – Первый макет

Рисунок 21 – Второй макет

При создании пользовательского интерфейса были использованы такие компоненты:

- LinearLayout;
- ScrollView;
- RelativeLayout;
- EditText;
- Spinner;
- Button;

- ListView;
- SearchView;
- TextView;
- Toast;
- LineChart.

График, созданный с помощью LineChart, показан на рисунке 22.

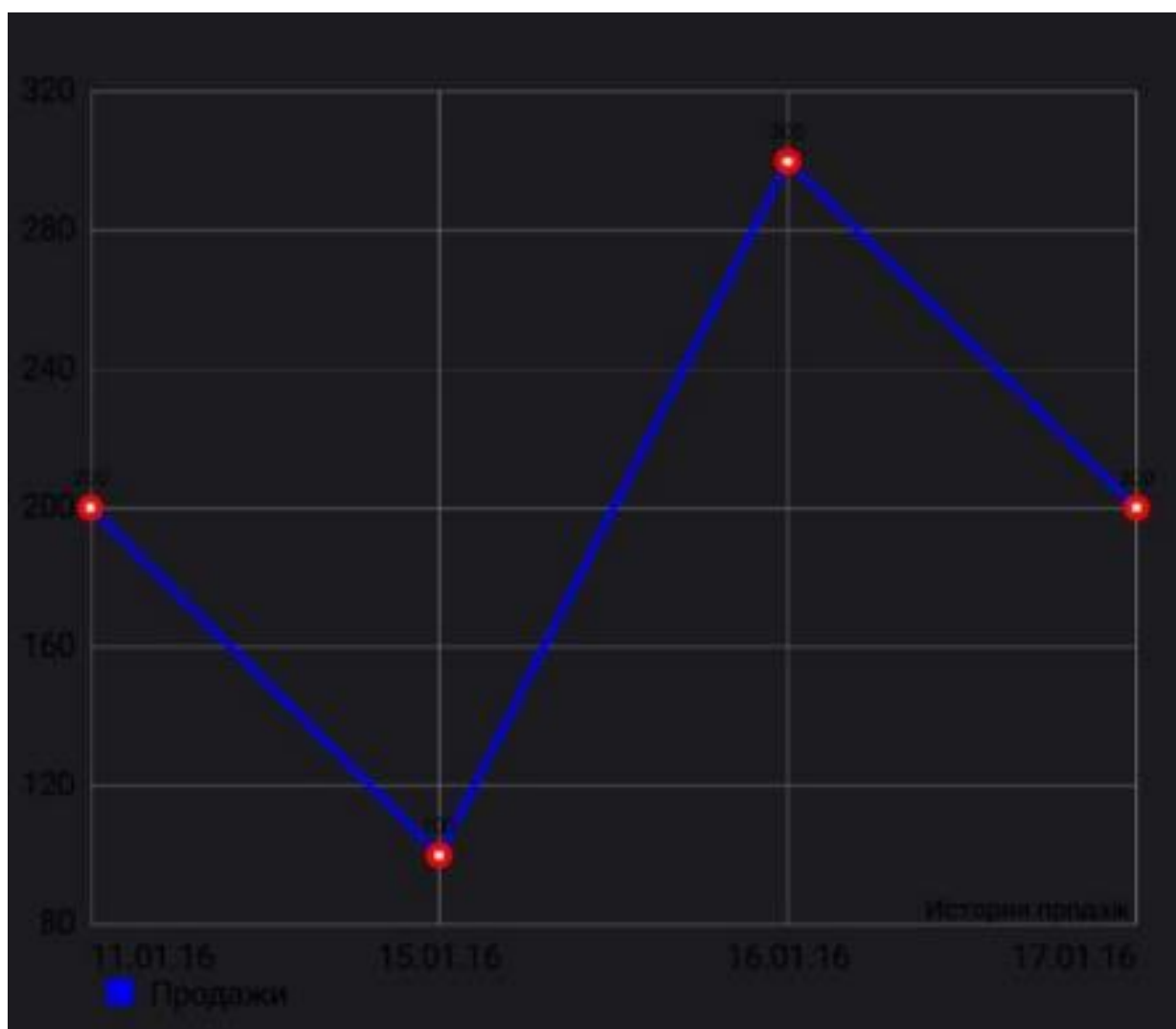


Рисунок 22 – График, созданный с помощью LineChart

Скрин маппинг представлен на рисунке 23.

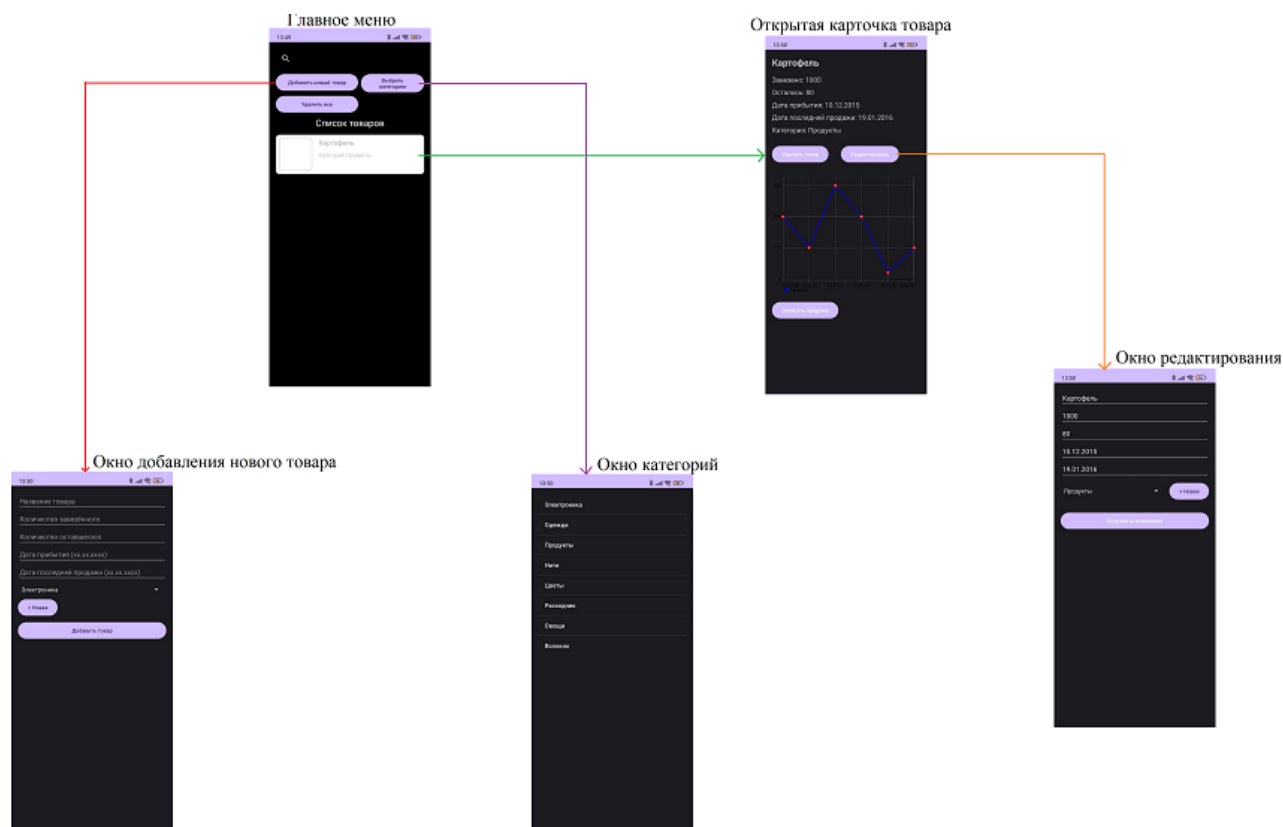


Рисунок 23 – Скрин маппинг

1.2.3 Архитектура программы

Архитектура программы состоит из трёх таблиц базы данных, а также из классов управления данными, добавления карточек товаров, отображения категорий, редактирования карточек товаров, отображения содержимого карточек товаров, главного меню. UML-диаграммы классов представлены на рисунках 24 и 25.

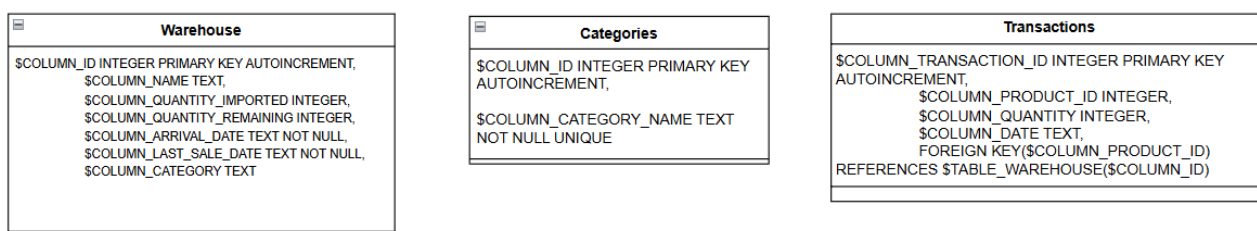


Рисунок 24 – UML-диаграмма классов, таблицы базы данных

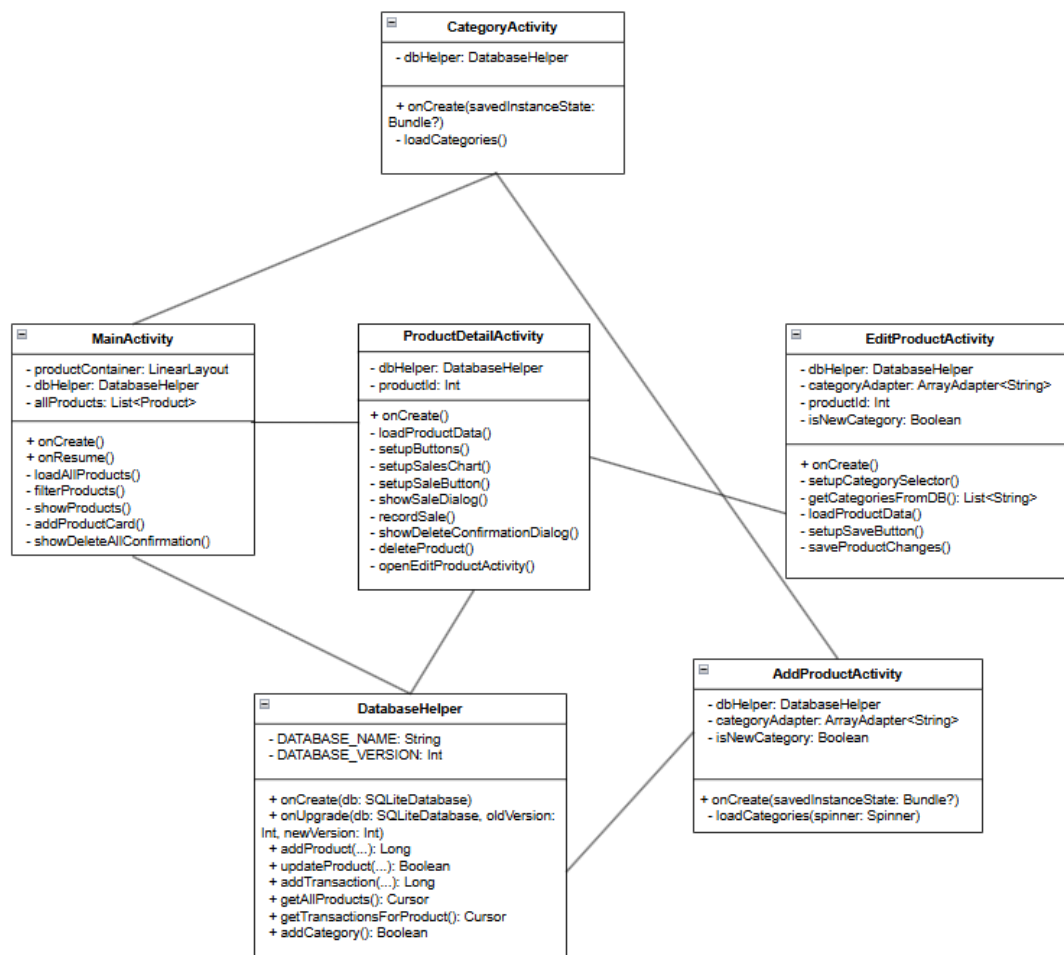


Рисунок 25 - UML-диаграмма классов, архитектура программы

MainActivity – главный класс программы, выполняет роль титульного экрана. Имеет такие главные функции:

- loadAllProducts(). Загрузка данных из БД;
- showProducts(). Отображение товаров;
- addProductCard(). Создание карточки товара;
- showDeleteAllConfirmation(). Удаление сразу группы позиций;
- confirmAndDeleteProduct(). Удаление конкретной позиции;
- setupSearchView(). Отвечает за поисковик.

ProductDetailActivity – класс, который отвечает за внутренности созданной карточки товара. Включает в себя такие функции;

- loadProductData(). Запрашивает данные товара через метод dbHelper и заполняет текстовые поля;

- setupSalesChart(). Создает график LineDataSet с осями количества продаж и дат, получает данные о продажах из БД, создает точки на графике;
- showSaleDialog(). Создает и отображает диалоговое окно для записи продажи;
- recordSale(). Фиксирует продажи, заносит их в БД;
- openEditProductActivity(). Вызов на редактирование;
- deleteProduct(). Удаление товара.

EditProductActivity – класс редактирования данных карточки товара.

Содержит такие функции:

- setupCategorySelector(). Настраивает выбор категории;
- getCagoriesFromDB(). Получение списка категорий из БД;
- loadProductData(). Загрузка данных товаров из БД;
- saveProductChanges(). Получение новых данных из полей editText и сохранение их.

CategoryActivity – класс, который служит экраном для отображения списка существующих категорий.

AddProductActivity – класс, суть которого заключается в добавлении новых в систему. Имеет такие функции:

- loadCategories(). Получение категорий из БД;
- buttonAddProduct.setOnClickListener. Обработчик добавления товаров, получает данные из полей editText и сохраняет их.

DatabaseHelper. Один из важнейших компонентов, который отвечает за создание и обновление БД, все CRUD операции, а также обеспечивает целостность данных. Содержит такие методы:

- addProduct(). Добавление нового товара;
- getProductById(). Получение товара по ID;
- updateProduct(). Обновление данных товара;
- deleteProduct(). Удаление товара;
- getAllProducts(). Все товары;
- getProductsByCategory(). Фильтрация по категории;

- addCategory(). Добавление категории с проверкой дубликатов;
- getAllCategories(). Список всех категорий;
- hasProductsInCategory(). Проверка наличия товаров в категории;
- addTransaction(). Запись операции (продажи/использования);
- getTransactionsForProduct(). История операций по товару;
- convertDateToDbFormat(). Унификация формата дат;
- searchProducts(). Поиск товаров по названию/категории.

1.2.4 Описание одного из основных методов программы

Одним из основных и важнейших методов программы является recordSale(). Данный метод реализует ключевую бизнес-логику приложения – учёт продаж товаров. Он обрабатывает продажи товара, обновляя остатки и историю транзакций.

На вход данный метод получает две переменные, введенные пользователем: date типа String в формате дд.мм.гггг, которая отвечает за дату последней продажи, и quantity типа Int, которая отвечает за количество проданного товара. Затем метод проверяет достаточно ли остатков на складе путём сравнения с основной переменной quantityRemaining, которая отвечает за количество оставшегося товара на складе, потом уменьшает её значение и обновляет lastSaleDate, которая отвечает за дату последней продажи.

```
dbHelper.updateProduct(
    id = productId,
    quantityRemaining = quantityRemaining - quantity,
    lastSaleDate = date
)
```

После происходит запись транзакции в таблицу transactions через DatabaseHelper.addTransaction(). Обе вышеперечисленные переменные обновляются и отображаются в основном интерфейсе открытой карточки товара с помощью метода loadProductData(), а график продаж – с помощью метода setupSalesChart().

UML-диаграмма последовательностей для метода recordSale() показана на рисунке 26.

					ПГУ 1.090301.21.001 ПЗ	Лист
						41
Изм.	Лист	№ докум.	Подпись	Дата		

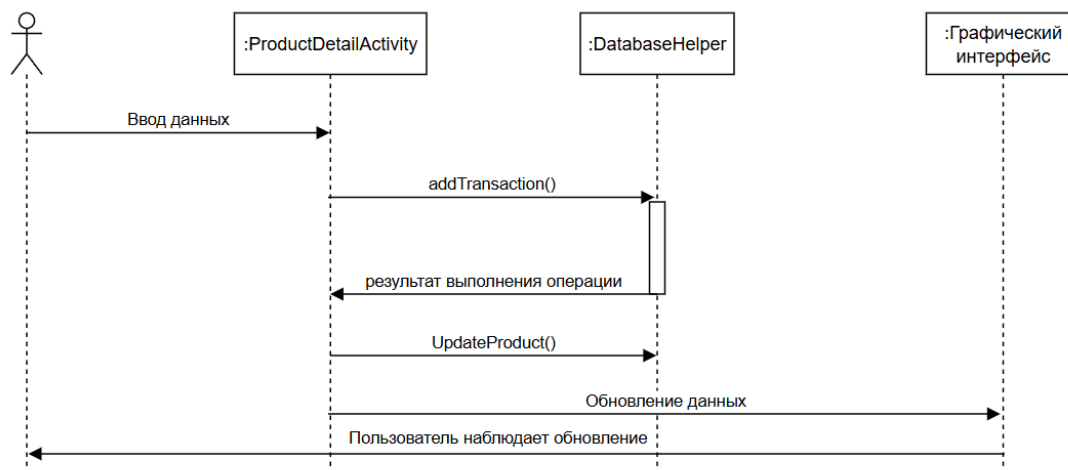


Рисунок 26 – UML-диаграмма последовательностей для метода recordSale()

1.2.5 Динамика работы системы приложения

При разработке программы создаётся UML-диаграмма состояний, которая должна помочь с пониманием того, какие состояния принимает приложение в зависимости от совершённых пользователем запросов. UML-диаграмма состояний программы показана рисунке 27.

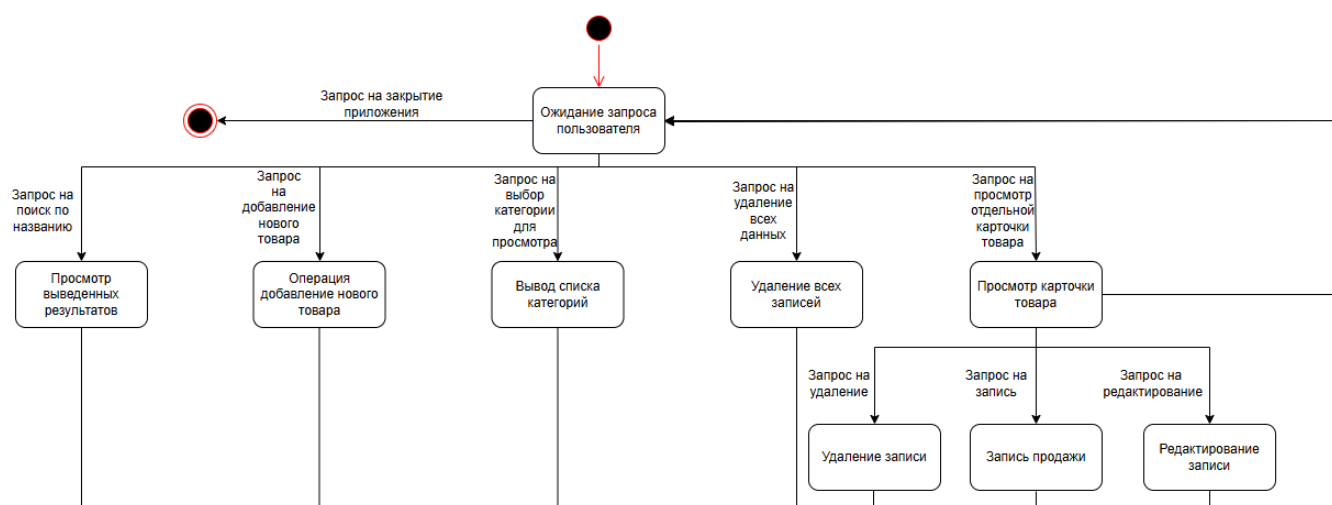


Рисунок 27 – UML-диаграмма состояний

Как видно на рисунке, приложение принимает такие состояния: просмотр выведенных результатов после запроса на поиск, добавление нового товара, вывод списка категорий, удаление всех записей, просмотр определённой записи, удаление записи, запись продажи, редактирование записи.

1.2.6 Описание AndroidManifest

AndroidManifest.xml – это центральный элемент каждого Android-приложения, который определяет его структуру, функции и взаимодействие с системой. Этот файл играет ключевую роль в процессе разработки и развертывания приложений, поскольку он управляет различными аспектами работы приложения, такими как разрешения, активности, службы и многое другое.

Файл состоит из Application Element.

```
<application
    android:allowBackup="true"
    android:dataExtractionRules="@xml/data_extraction_rules"
    android:fullBackupContent="@xml/backup_rules"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:roundIcon="@mipmap/ic_launcher_round"
    android:supportsRtl="true"
    android:theme="@style/Theme.Test3"
    tools:targetApi="31">
```

Данный блок содержит такие атрибуты:

- allowBackup. Разрешение автоматического резервного копирования;
- dataExtractionRules. Определение правил экспорта данных для Android;
- fullBackupContent. Указание файлов для полного резервного копирования;
- icon, roundIcon. Иконки для разных форм-факторов;
- supportRtl. Поддержка языков с написанием справа налево;
- theme. Базовая тема оформления интерфейса;
- targetApi. Приоритетный параметр для тестирования приложения под Android 12.

Вместе с тем файл состоит из 5 Activity Element.

Первая – MainActivity.

```

<activity
    android:name=".MainActivity"
    android:exported="true">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>

```

Атрибут `exported` отвечает за то, что данная активность может быть вызвана из других приложений.

Блок `intent filter` отвечает за то, что именно `MainActivity` будет открываться при запуске приложения (`<action android:name="android.intent.action.MAIN" />`), а также за то, чтобы система Android определяла приложение как установленное и создала ему иконку на рабочем столе (`<category android:name="android.intent.category.LAUNCHER" />`).

CategoryActivity.

```

<activity
    android:name=".CategoryActivity"
    android:exported="false" />
EditProductActivity:
<activity android:name=".EditProductActivity"/>
AddProductActivity:
<activity
    android:name=".AddProductActivity"
    android:exported="false" />
ProductDetailActivity:
<activity
    android:name=".ProductDetailActivity"
    android:exported="false" />

```

Схема взаимодействия активностей на основе манифест-файла показана на рисунке 28.

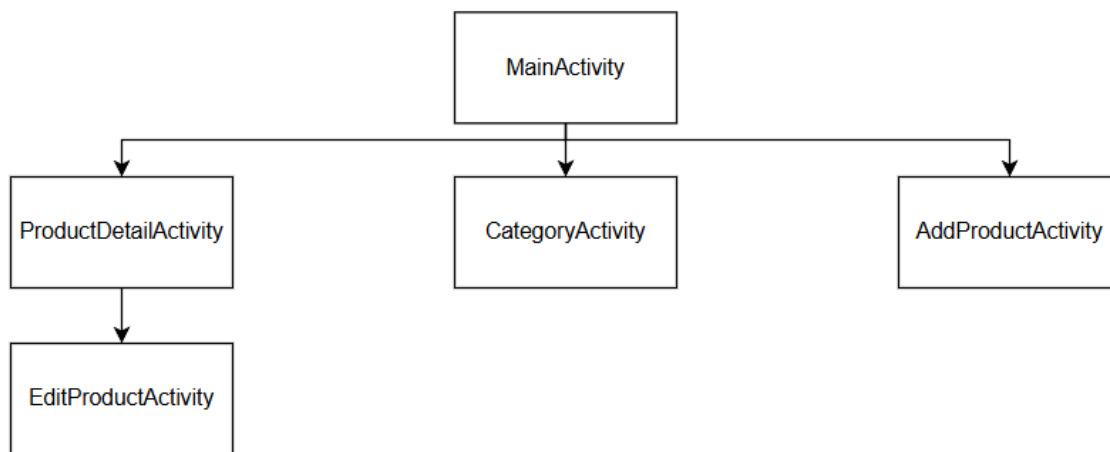


Рисунок 28 – Схема взаимодействия активностей на основе манифест-файла

1.2.7 Описание работы класса DatabaseHelper

Для работы с БД используются методы SQL, специальные команды, которые позволяют взаимодействовать, сортировать, искать и извлекать нужную информацию.

Все SQL методы расположены в одном классе программы – DatabaseHelper.

Для создания таблиц используется метод CREATE TABLE.

Код создания таблицы для хранения данных о товарах приведён ниже.

```

CREATE TABLE $TABLE_WAREHOUSE (
    $COLUMN_ID INTEGER PRIMARY KEY AUTOINCREMENT,
    $COLUMN_NAME TEXT,
    $COLUMN_QUANTITY_IMPORTED INTEGER,
    $COLUMN_QUANTITY_REMAINING INTEGER,
    $COLUMN_ARRIVAL_DATE TEXT NOT NULL,
    $COLUMN_LAST_SALE_DATE TEXT NOT NULL,
    $COLUMN_CATEGORY TEXT
)
  
```

Из особенностей данной таблицы можно отметить колонку ID с атрибутом AUTOINCREMENT. Данный атрибут гарантирует уникальность и неизменяемость ID товаров, упрощает использование, так как пользователю не нужно придумывать и вводить свои ID, программа делает всё за него.

Код создания таблицы для категорий приведён ниже.

```
CREATE TABLE $TABLE_CATEGORIES (
    $COLUMN_ID INTEGER PRIMARY KEY AUTOINCREMENT,
    $COLUMN_CATEGORY_NAME TEXT NOT NULL UNIQUE
```

Код создания таблицы транзакций приведён ниже.

```
CREATE TABLE $TABLE_TRANSACTIONS (
    $COLUMN_TRANSACTION_ID INTEGER PRIMARY KEY AUTOINCREMENT,
    $COLUMN_PRODUCT_ID INTEGER,
    $COLUMN_QUANTITY INTEGER,
    $COLUMN_DATE TEXT,
    FOREIGN KEY($COLUMN_PRODUCT_ID) REFERENCES
    $TABLE_WAREHOUSE($COLUMN_ID)
```

Последняя строка данной таблицы транзакций означает, что каждая запись из данной таблицы обязана ссылаться на существующую запись ID из таблицы Warehouse. Это позволяет избежать возможности появления лишних записей о продажах и ошибок при вычислении остатков на складе.

Добавление новой карточки товара происходит с помощью метода addProduct().

```
fun addProduct(name: String, quantityImported: Int,
    quantityRemaining: Int,
        arrivalDate: String, lastSaleDate: String,
category: String): Long {
    val db = this.writableDatabase
    val values = ContentValues().apply {
        put(COLUMN_NAME, name)
        put(COLUMN_QUANTITY_IMPORTED, quantityImported)
        put(COLUMN_QUANTITY_REMAINING, quantityRemaining)
        put(COLUMN_ARRIVAL_DATE, arrivalDate)
        put(COLUMN_LAST_SALE_DATE, lastSaleDate)
        put(COLUMN_CATEGORY, category)
    }
    return db.insert(TABLE_WAREHOUSE, null, values)
```

В параметрах функции прописываются нужные переменные с их типом данных, затем объявляется переменная db, которая является экземпляром БД с возможностью записи и изменения.

После создаются переменные ContentValues, то есть особые переменные для хранения данных, которые впоследствии будут вставлены в БД. А само занесение новых записей в БД происходит с помощью метода `db.insert(TABLE_WAREHOUSE, null, values)`.

Для удаления записей используется функция `deleteProduct`.

```
fun deleteProduct(id: Long): Boolean {  
    val db = this.writableDatabase  
    val result = db.delete(TABLE_WAREHOUSE, "$COLUMN_ID = ?",  
        arrayOf(id.toString())) > 0  
    db.close()  
    return result  
}
```

Удаление карточки происходит по ID, поэтому в параметрах функции указывается переменная `id` и её тип. Удаление происходит с помощью метода `db.delete`, в который передаётся номер записи, выбранный пользователем, и с помощью условия `WHERE` сравнивается со значениями ID внутри базы данных.

Удаление всех товаров, находящихся в определённой категории, происходит с помощью функции `deleteProductByCategory()`.

```
fun deleteProductsByCategory(category: String): Boolean {  
    val db = this.writableDatabase  
    val result = db.delete(TABLE_WAREHOUSE, "$COLUMN_CATEGORY = ?",  
        arrayOf(category)) > 0  
    db.close()  
    return result  
}
```

Удаление вообще всех товаров происходит с помощью функции `deleteAllProducts()`.

```
fun deleteAllProducts(): Boolean {  
    val db = this.writableDatabase  
    val result = db.delete(TABLE_WAREHOUSE, null, null) > 0  
    db.close()  
    return result  
}
```

Для получения всех записей из таблицы используется функция `getAllProducts()`.

```
fun getAllProducts(): Cursor {  
    val db = this.readableDatabase  
    return db.rawQuery("SELECT * FROM $TABLE_WAREHOUSE", null)  
}
```

Данная функция возвращает все записи таблицы в формате `Cursor`. `Cursor` – это своего рода указатель, который позволяет перемещаться по строкам с данными внутри таблицы SQL. Выборка происходит с помощью метода `db.rawQuery`, внутри которого есть условие, которое выбирает все данные внутри таблицы Warehouse.

Для получения всех записей определённой категории используется функция `getProductByCategory()`.

```
fun getProductsByCategory(category: String): Cursor {  
    val db = this.readableDatabase  
    return db.rawQuery("SELECT * FROM $TABLE_WAREHOUSE WHERE  
$COLUMN_CATEGORY = ?", arrayOf(category))  
}
```

Единственным отличием от `getAllProducts()` является лишь то, что в методе `db.rawQuery` появляется дополнительное условие `WHERE` с указанием на выбранную пользователем категорию.

Для получения всех записей по ID используется функция `getProductById()`.

```
fun getProductById(id: Int): Cursor {  
    val db = this.readableDatabase  
    return db.rawQuery("SELECT * FROM $TABLE_WAREHOUSE WHERE  
$COLUMN_ID = ?", arrayOf(id.toString()))  
}
```

Она тоже мало чем отличается от `getAllProducts`, в методе `db.rawQuery` лишь добавляется условие `WHERE` с указанием на выбранный пользователем ID.

Функция получения всех категорий приведена ниже.

```

fun getAllCategories(): Cursor {
    val db = this.readableDatabase
    return db.rawQuery("SELECT * FROM $TABLE_CATEGORIES", null)
}

```

Для добавления новой категории и проверки на наличие существующей при создании карточки товара используется метод `addCategory()`.

```

fun addCategory(category: String): Boolean {
    // Проверяем, есть ли уже такая категория
    val db = this.readableDatabase
    val cursor = db.query(
        TABLE_CATEGORIES,
        arrayOf(COLUMN_ID),
        "$COLUMN_CATEGORY_NAME = ?",
        arrayOf(category),
        null, null, null
    )
    val exists = cursor.count > 0
    cursor.close()
    if (exists) return true // Категория уже есть
    // Добавляем новую категорию
    val values = ContentValues().apply {
        put(COLUMN_CATEGORY_NAME, category)
    }
    return db.insert(TABLE_CATEGORIES, null, values) != -1L
}

```

В методе `db.query` идёт поиск по ID товаров категорий, если введённая категория уже существует, то `cursor.count` становится 1, если нет – процесс выполнения метода `addCategory()` продолжается. `put()` заносит в `ContentValues` новую категорию, введённую пользователем, а `db.insert()` заносит её в таблицу БД.

Для обновления данных о товарах используется метод `updateProduct()`.

```

fun updateProduct(
    id: Long,
    name: String,
    quantityImported: Int,

```

```

        quantityRemaining: Int,
        arrivalDate: String,
        lastSaleDate: String,
        category: String
    ): Boolean {
        val db = this.writableDatabase
        val values = ContentValues().apply {
            put(COLUMN_NAME, name)
            put(COLUMN_QUANTITY_IMPORTED, quantityImported)
            put(COLUMN_QUANTITY_REMAINING, quantityRemaining)
            put(COLUMN_ARRIVAL_DATE, arrivalDate)
            put(COLUMN_LAST_SALE_DATE, lastSaleDate)
            put(COLUMN_CATEGORY, category)
        }

        val result = db.update(
            TABLE_WAREHOUSE,
            values,
            "$COLUMN_ID = ?",
            arrayOf(id.toString())
        ) > 0

        db.close()
        return result
    }

```

Для поиска товаров через строку поиска используется метод `searchProducts()`.

```

fun searchProducts(query: String): Cursor {
    return readableDatabase.rawQuery(
        "SELECT * FROM $TABLE_WAREHOUSE WHERE $COLUMN_NAME LIKE ? OR $COLUMN_CATEGORY LIKE ?",
        arrayOf("%$query%", "%$query%")
    )
}

```

В методе `rawQuery` идёт условие поиска с помощью оператора `LIKE`, который ведёт сравнение с введённым пользователем набором букв, либо словом, которое подставлено на место `'%$query%'` в таблице Warehouse среди колонок названий и категорий товаров.

Добавление продажи товара происходит с помощью метода `addTransaction()`.

```
fun addTransaction(productId: Long, quantity: Int, date: String):
Long {
    val values = ContentValues().apply {
        put(COLUMN_PRODUCT_ID, productId)
        put(COLUMN_QUANTITY, quantity)
        put(COLUMN_DATE, convertDateToDbFormat(date))
    }
    return writableDatabase.insert(TABLE_TRANSACTIONS, null, values)
}
```

Для построения графика и отображения истории продаж на нём используется метод `getTransactionsForProduct()`.

```
fun getTransactionsForProduct(productId: Long): Cursor {
    return readableDatabase.query(
        TABLE_TRANSACTIONS,
        null,
        "$COLUMN_PRODUCT_ID = ?",
        arrayOf(productId.toString()),
        null, null,
        "$COLUMN_DATE ASC"
    )
}
```

Метод возвращает все продажи, введённые пользователем, а также даты продаж в порядке возрастания.

1.3 Тестирование программы

Для тестирования программы открываем её и нажимаем на кнопку «Добавить» новый товар. Главное меню программы показано на рисунке 29.

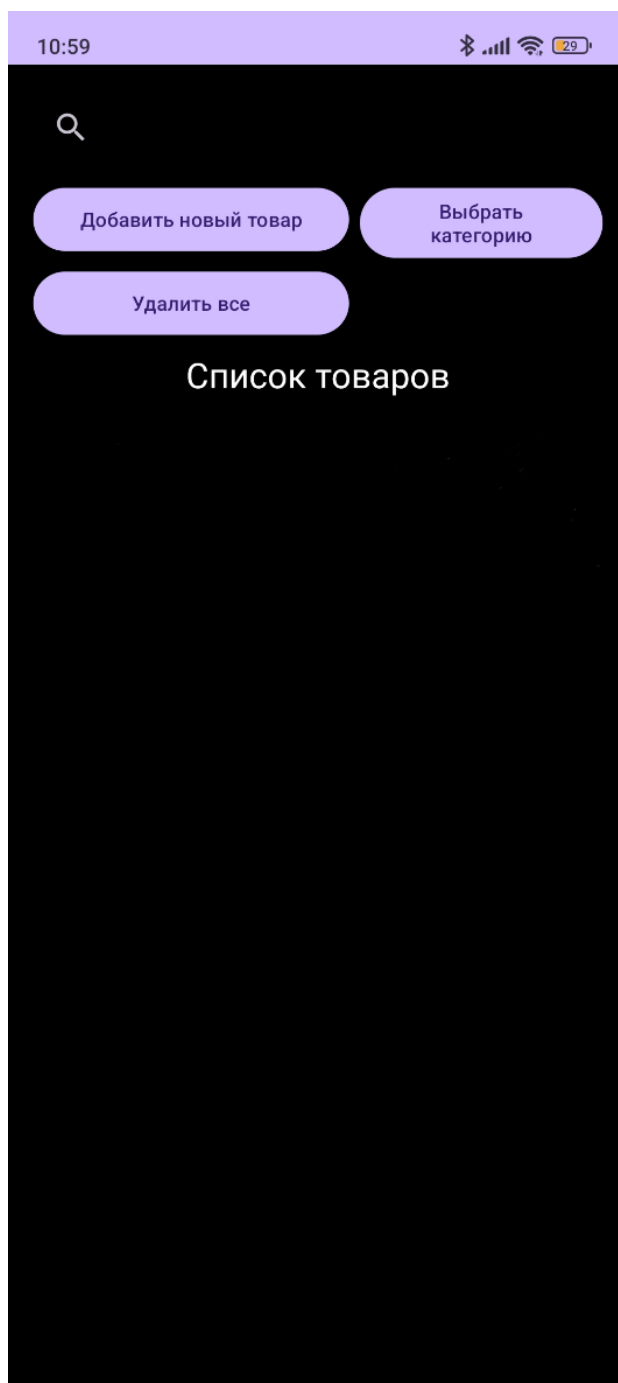


Рисунок 29 – Главное меню приложения

Заполняем все нужные поля. Форма заполнения для нового товара показана на рисунке 30.

					ПГУ 1.090301.21.001 ПЗ	Лист
						52
Изм.	Лист	№ докум.	Подпись	Дата		

Рисунок 30 – Меню добавления нового товара

Добавленная позиция теперь отображается в главном меню. Новый товар в списке представлен на рисунке 31.

					ПГУ 1.090301.21.001 ПЗ	Лист
						53
Изм.	Лист	№ докум.	Подпись	Дата		

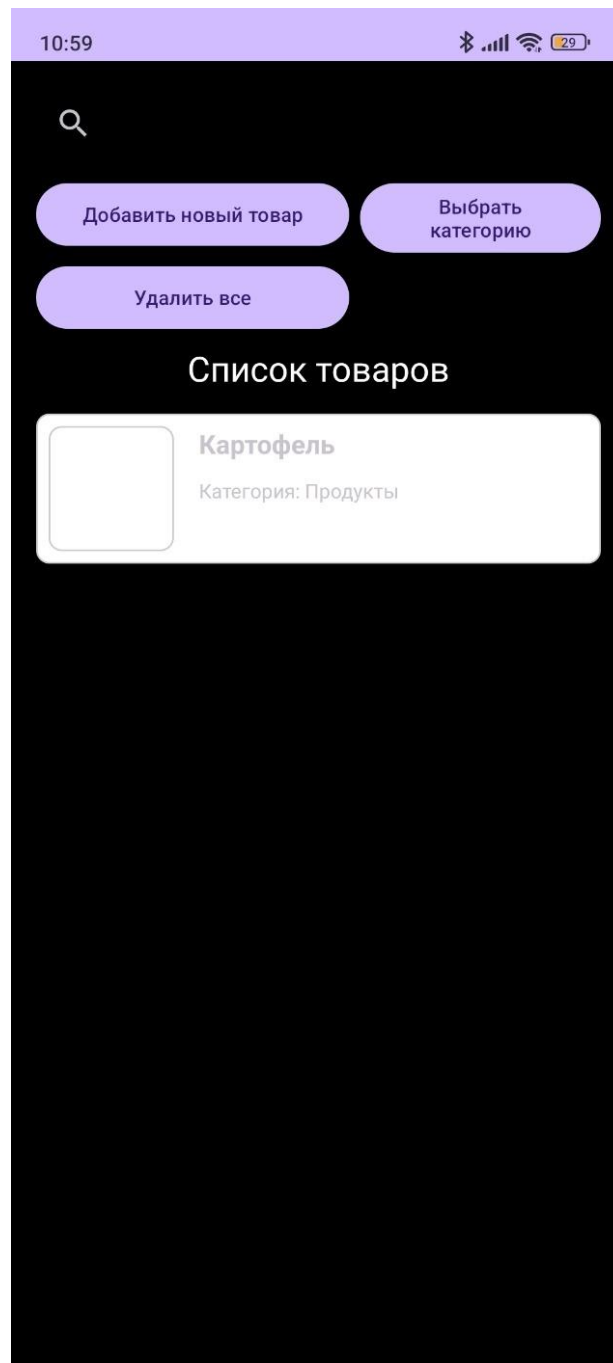


Рисунок 31 – Новая позиция в главном меню

В открытой карточке товара можно записать новую продажу и посмотреть её на графике. Были введены тестовые данные в виде даты продажи и количества проданных товаров. Тест работы показан на рисунке 32.

					ПГУ 1.090301.21.001 ПЗ	Лист
						54
Изм.	Лист	№ докум.	Подпись	Дата		

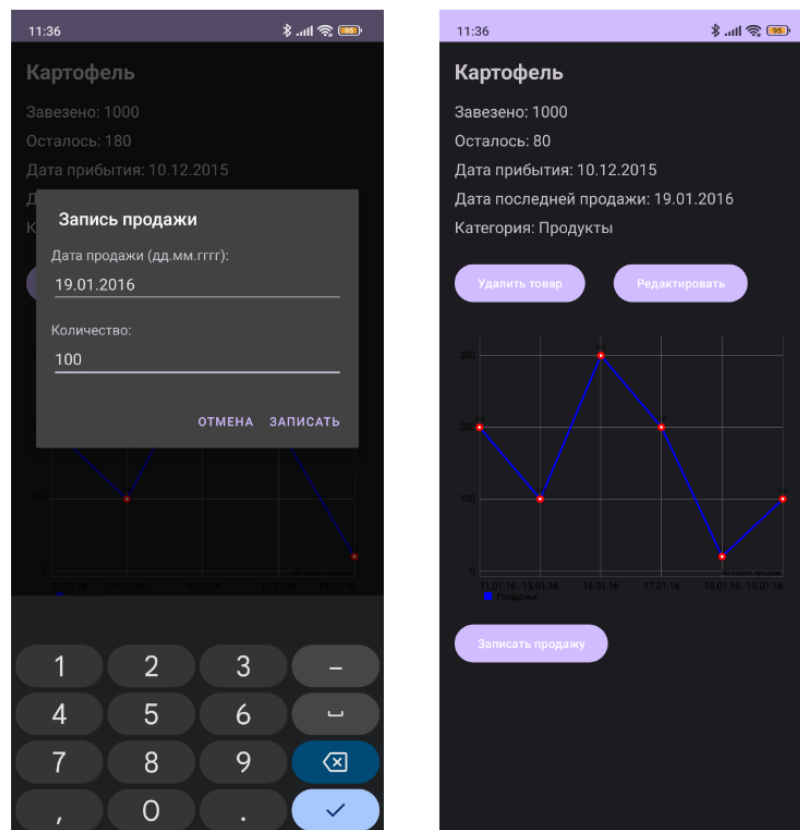


Рисунок 32 – Изменения

График с изменениями после теста показан на рисунке 33.

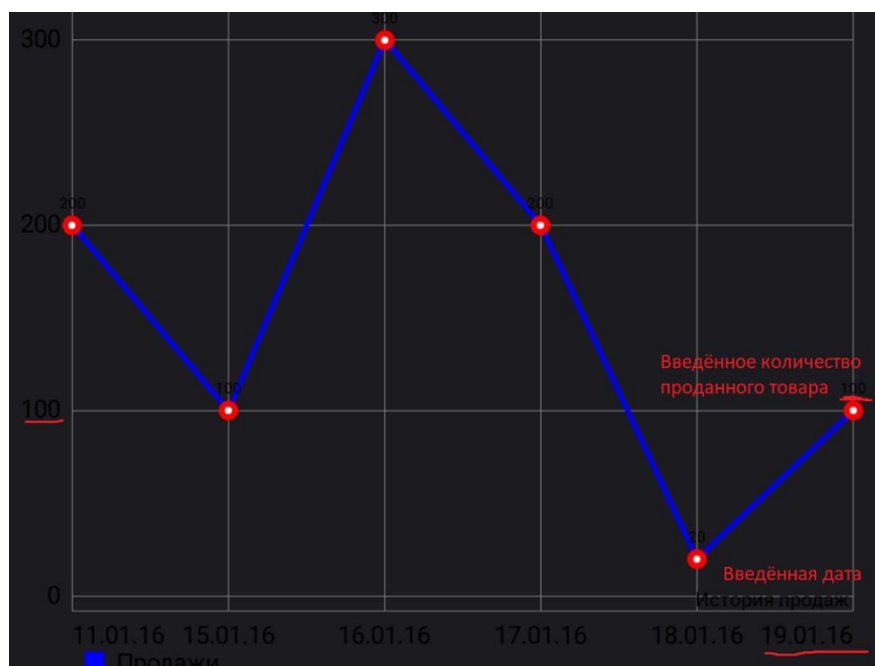


Рисунок 33 – Значения на графике

После скачивания программы для просмотра баз данных SQLite, в данном случае DB Browser for SQLite, открываем там нашу базу данных и сверяем со значениями в приложении. Проверка через программу показана на рисунке 34.

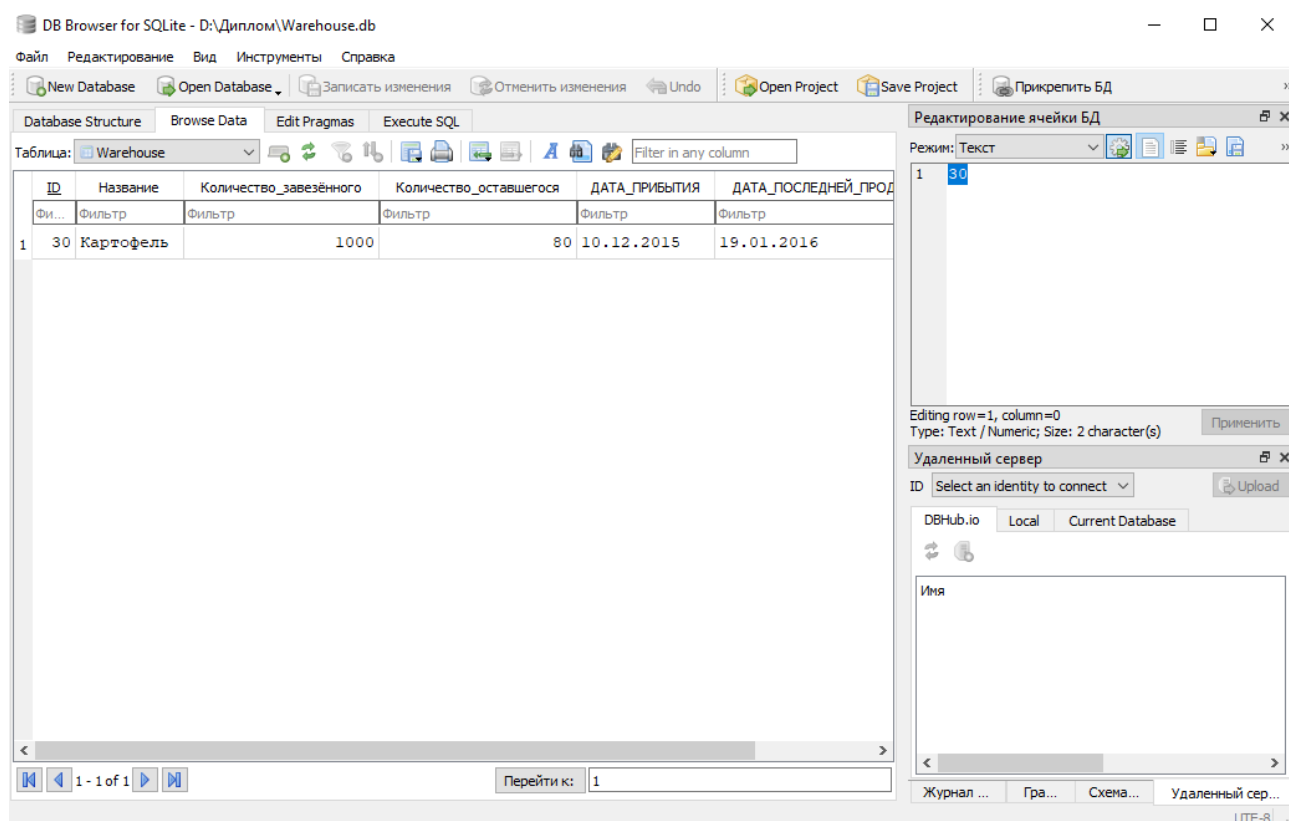


Рисунок 34 – Данные базы данных

Все значения совпадают, а это значит, что программа работает корректно.

2 Охрана труда

Данный раздел посвящён соблюдению небольшого списка правил безопасности при использовании смартфонов, будут освещены методы снижения вредоносного влияния на состояние здоровья человека при использовании мобильных устройств.

2.1 Характеристика потенциально вредоносного влияния при использовании смартфонов

Ежедневно люди используют свои мобильные телефоны не только как средства связи, но и как устройства, которые служат для получения развлекательного контента, помогают в работе, но вместе с тем они могут приносить не только пользу, но и вред. При использовании смартфонов стоит обезопасить себя тем, чтобы держать смартфон на достаточном расстоянии от глаз, постоянно не носить сотовый телефон рядом с собой и так далее.

2.2 Сводка рекомендаций для минимизации вреда со стороны мобильных устройств

Во избежание вредоносного влияния смартфонов стоит придерживаться рекомендаций, приведённых ниже:

- продолжительность разговора по мобильному телефону не должна превышать двух минут, а между звонками следует выдерживать паузу не менее 15 минут. Более безопасным способом является отправка сообщений, нежели держать телефон близко к уху;
- телефон рекомендуется держать вертикально, охватывая нижнюю часть корпуса и немного отодвигая от уха. Поскольку интенсивность радиоволн уменьшается обратно пропорционально квадрату расстояния, даже смещение на один сантиметр значительно снижает воздействие излучения на мозг — примерно в четыре раза;
- следует прикладывать телефон к уху лишь после того, как вызов будет

					ПГУ 1.090301.21.001 ПЗ	Лист
						57
Изм.	Лист	№ докум.	Подпись	Дата		

принят на другом конце линии. Во время набора номера устройство функционирует на максимальной мощности, независимо от качества сигнала. Однако спустя 10–20 секунд после соединения уровень излучения падает до минимально необходимого. Кроме того, сразу прикладывать трубку к уху нецелесообразно, поскольку первый длинный гудок появляется с задержкой;

- во время разговора рекомендуется снимать очки с металлической оправой, так как металлические детали могут усиливать электромагнитное поле, воздействующее на пользователя;

- рекомендуется не оставлять включённый мобильный телефон рядом с собой во время сна;

- постоянное ношение телефона при себе, например, в кармане брюк, стоит избегать;

- контакт с мобильным телефоном следует свести к минимуму, особенно если в этом нет особой необходимости;

- лучше всего хранить мобильный телефон в сумке; длительное ношение его на груди, поясе или в нагрудном кармане не рекомендуется.

2.3 Классификация помещения, в котором происходит работа

Современные исследования в области охраны труда выделяют малоподвижный характер работы с компьютерной техникой как значительный фактор риска для здоровья. Длительное статическое напряжение опорно-двигательного аппарата в сочетании с постоянной зрительной нагрузкой приводит к кумулятивному эффекту, проявляющемуся в снижении продуктивности, повышенной утомляемости и развитии профессионально обусловленных заболеваний.

Для минимизации негативных последствий компьютерной работы разработан комплекс организационно-технических мероприятий:

Оптимизация рабочего процесса включает:

- антропометрическая настройка рабочего места – подбор мебели с учетом особенностей телосложения пользователя позволяет сохранить естественные

изгибы позвоночника и обеспечить физиологичное положение конечностей;

- эргономичное размещение оборудования – визуальный дисплей следует устанавливать на расстоянии 60–70 см от оператора, при этом верхний край экрана должен находиться на 10-15° ниже горизонтальной линии взгляда. Особое внимание уделяется исключению паразитной засветки экрана;

- регламентация рабочего времени – научно обоснованный режим труда предполагает циклы работы продолжительностью 45–50 минут с последующими 10–15-минутными перерывами для выполнения профилактических упражнений;

- контроль осанки – систематический мониторинг положения тела во время работы предотвращает развитие мышечных дисбалансов и дегенеративных изменений позвоночника;

- техника безопасности – обязательное соблюдение нормативов по электробезопасности и противопожарной защите рабочего места.

2.3.1 Эргономические стандарты организации рабочего пространства

Современные нормативные документы регламентируют параметры, которые представлены ниже.

Оптимальная зона визуального восприятия: 60–70 см от глаз пользователя до поверхности монитора при угле наклона 10-15°.

Параметры рабочей мебели:

- регулируемая высота рабочей поверхности (680–800 мм);
- стандартная высота при отсутствии регулировки – 725 мм.

Минимальные габариты подстольного пространства:

- высота в размере 600 мм;
- глубина на уровне колен в размере 450 мм;
- глубина для вытянутых ног в размере 650 мм.

Рациональное размещение устройств ввода:

- клавиатура располагается на расстоянии 10–30 см от края стола;
- предпочтительно использование выдвижных панелей для клавиатуры.

Критерии правильной рабочей позы:

- нейтральное положение кистей (без изгиба в запястьях);
- естественное положение плечевого пояса;
- оптимальный угол в локтевых суставах (90-100°);
- полноценная опора для предплечий.

					ПГУ 1.090301.21.001 ПЗ	Лист
						60
Изм.	Лист	№ докум.	Подпись	Дата		

Заключение

В рамках данной выпускной квалификационной работы было разработано приложение для учёта склада под ОС Android. Приложение было реализовано на языке Kotlin в IDE Android Studio, база данных реализована с помощью СУБД SQLite.

В ВКР реализованы следующие функции: предоставление к добавлению новых товаров, редактирование и удаление карточек товаров, предоставление доступа к просмотру карточек товаров, наличие и возможность использования поисковой строки, наличие и использование графика продаж, возможность просмотра категорий.

Данное приложение может быть доработано и улучшено в дальнейшем путём добавления функций сортировки, фильтров, возможности экспорта данных в Excel-таблицы, возможности добавления сканера штрих-кодов.

В процессе разработки было проведено тестирование, на основе которого можно говорить о корректности работы приложения. Анализ результатов тестирования показал успешное соответствие программы заявленным требованиям, а её базовые функции окажут поддержку в учёте складов.

					ПГУ 1.090301.21.001 ПЗ	Лист
						61
Изм.	Лист	№ докум.	Подпись	Дата		

Список использованных источников

1. Обобщённый Model-View-Controller. Каркас на основе шаблона проектирования MVC в исполнении Generic Java и C#, Сергей Рогачёв, ООО “Лукойл-Информ” г.Пермь, 23.03.2007.
2. Material Desig [Электронный ресурс] – URL: <https://media.contented.ru/glossary/materialdesign/?ysclid=maqm4x3z3y219903352> (дата обращения 10.05.2025).
3. Что такое CRUD, простыми словами об использовании подхода [Электронный ресурс] – URL: <https://orbitsoft.com/ru/blog/crud/> (дата обращения 10.05.2025).
4. ScrollView [Электронный ресурс] – URL: <https://metanit.com/java/android/3.11.php?ysclid=maqpcahtkg214431495> (дата обращения 12.05.2025).
5. Toast [Электронный ресурс] – всплывающие сообщения – URL: <https://developer.alexanderklimov.ru/android/toast.php?ysclid=mapdb5nws5626147974> (дата обращения 12.05.2025).
6. LineChart [Электронный ресурс] – URL: <https://github.com/Teddko/LineChart> (дата обращения 12.05.2025).
7. SQLite на Android [Электронный ресурс] – URL: <https://developer.alexanderklimov.ru/android/sqlite/android-sqlite.php> (дата обращения 14.05.2025).
8. Что такое AndroidManifest.xml, его структура и важность [Электронный ресурс] – URL: <https://habr.com/ru/articles/844612/> (дата обращения 14.05.2025).
9. Санитарные правила и нормы. СанПиН 2.2.2.542-96 [Электронный ресурс] – URL: https://www.consultant.ru/document/cons_doc_LAW_18895/df3eaac6ab127894002172933f9cd660e61df962/ (дата обращения 20.05.2025).
10. О рекомендациях по безопасному использованию мобильных телефонов [Электронный ресурс] – URL: <https://71.rospotrebnadzor.ru/content/591/90401/?ysclid=mbgq3jrn5b858843087> (дата обращения 20.05.2025).

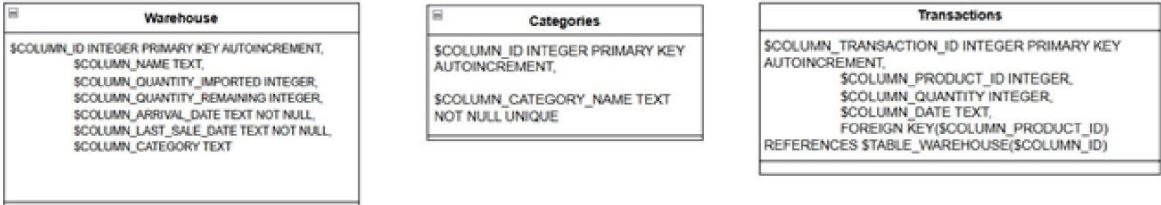
Приложение А

(обязательное)

Программа учёта склада под ОС Android

UML-диаграммы

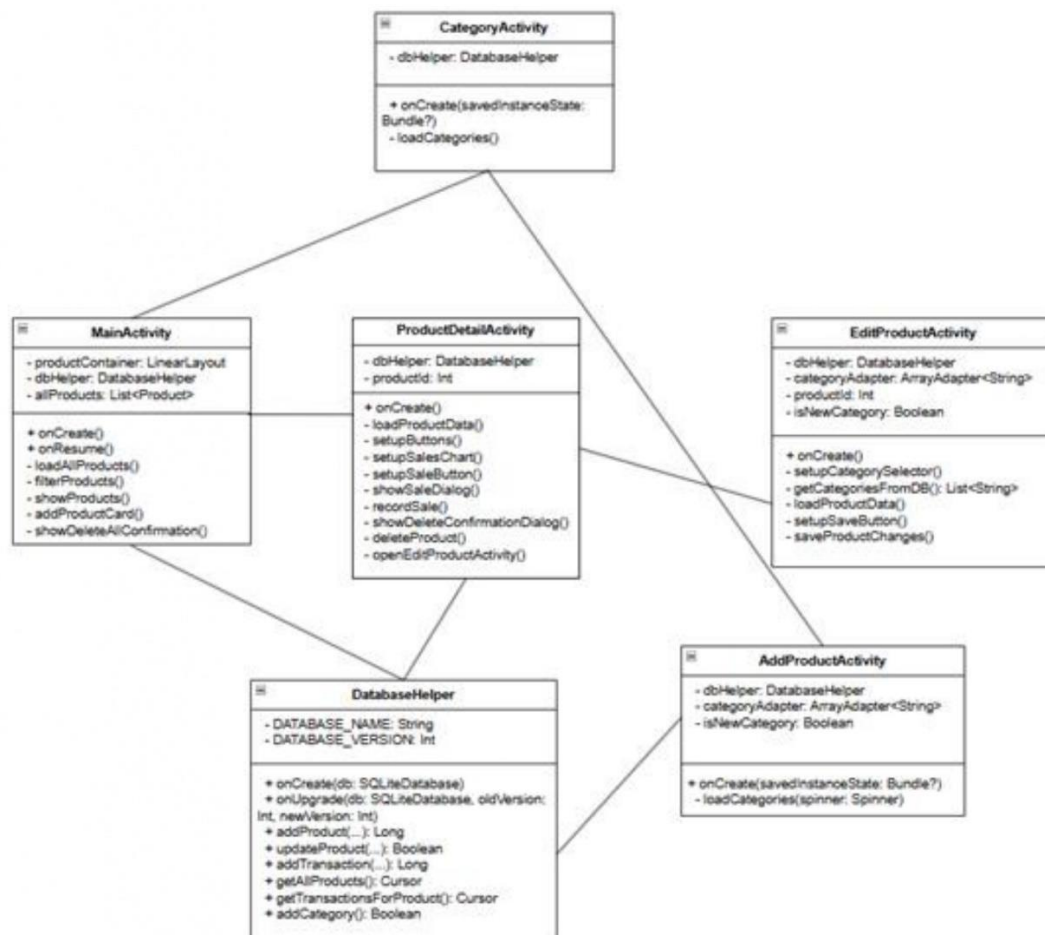
Приложение А.1 – UML-диаграмма классов



					ПГУ 1.090301.01 001 Д11			
Изм.	Лист	Версия	Измен.	Дата	Дополнительная информация по ГОСТ-ISO 9001:2015 – Дополнительная информация	Исх.	Внесено	Модифицировано
Разработчик	Исполнитель	Проверен	Утвержден	Дата		Исх.	Внесено	Модифицировано
Создан	Исполнен	Проверен	Утвержден	Дата		Исх.	Внесено	Модифицировано
Разработчик	Исполнитель	Проверен	Утвержден	Дата		Исх.	Внесено	Модифицировано
И. комп.	Байков А.С.				Ф87, стр. 216887			
Упр.	Митронов М.А.							

Изм.	Лист	№ докум.	Подпись	Дата

Приложение А.2 – UML-диаграмма классов



ПГУ 1.090301.01 002 Д11					Лист		Всего	Масштаб
Дир. ПГУ	Мухомов	Григорьев	Дрозд	Программисты	Лист	Всего	Масштаб	
Инженер	Борисенко Е.А.			Инженер	Лист	Всего	Масштаб	
Инженер	Томаров А.И.			Инженер	Лист	Всего	Масштаб	
Инженер	Павлов			Инженер	Лист	Всего	Масштаб	
Инженер	Ворожцов			Инженер	Лист	Всего	Масштаб	
Инженер	Мерзляков			Инженер	Лист	Всего	Масштаб	

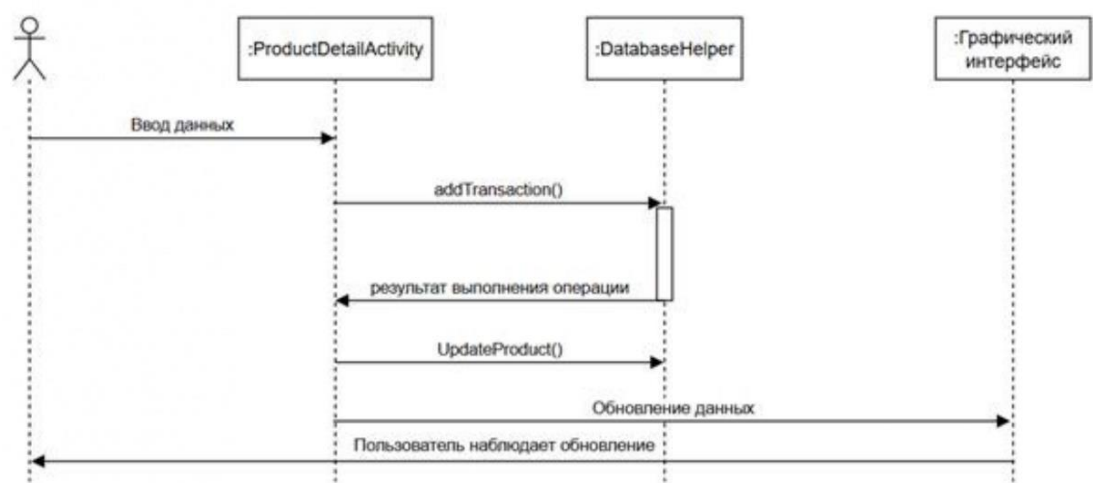
Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090301.21.001 ПЗ

Лист

65

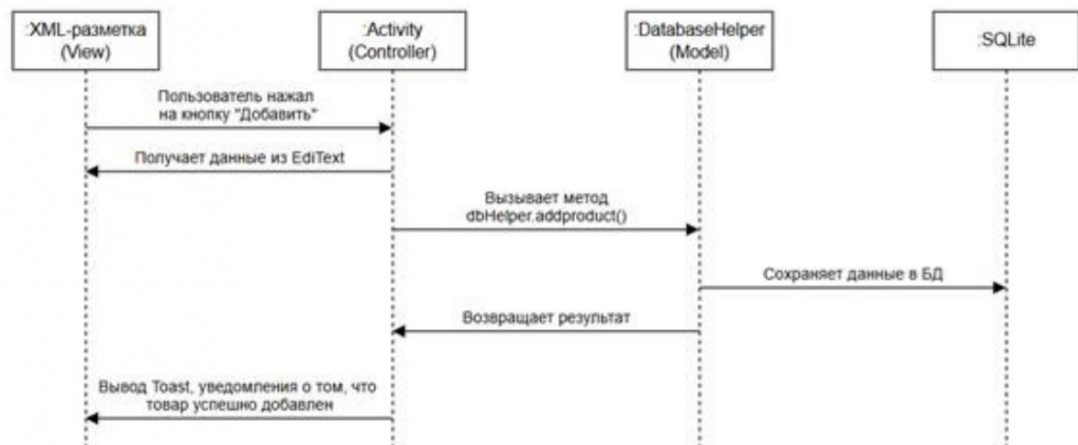
Приложение А.3 – UML-диаграмма последовательностей



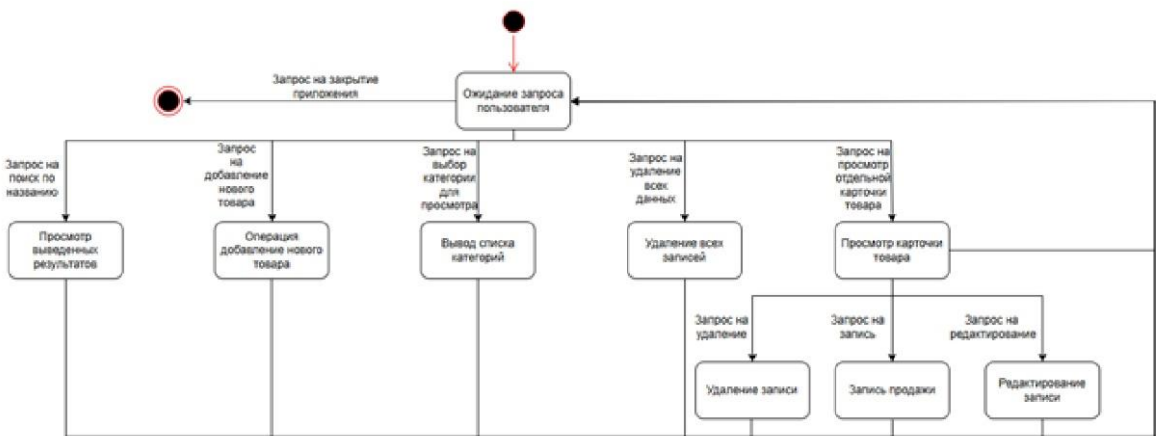
ПГУ 1.090301.01 003 Д14									
Имя	Долг	Место	Полном.	Долг	Дирекция учебно-исследовательской работы ПГУ, кафедра ИТМ. Документы по последовательности				
Риски	Корректировка	С.А.							
Срок	Полном.	С.А.							
С. номер									
Риски					ФЕТ. № 216881				
С. номер	Каждый С.								
Имя	Место	С.А.							

Изм.	Лист	№ докум.	Подпись	Дата

Приложение А.4 – UML-диаграмма последовательностей

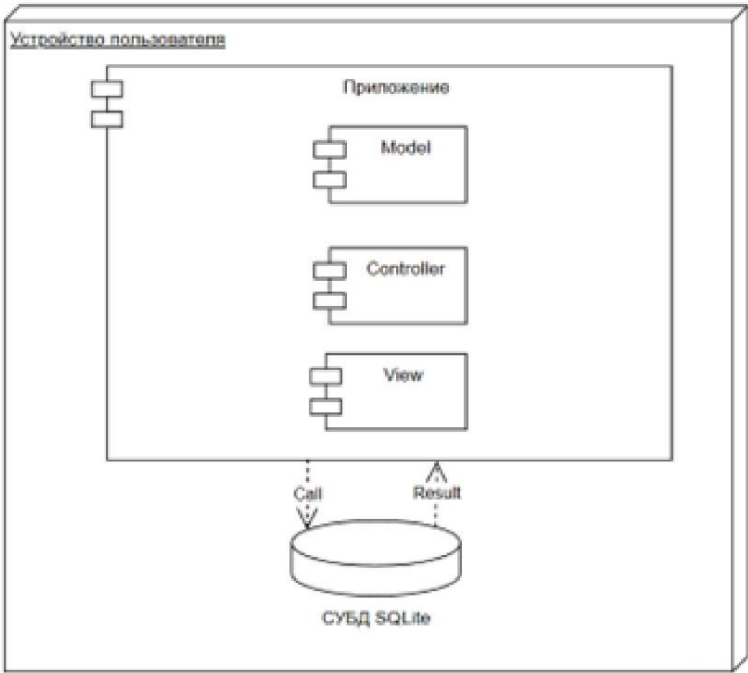
[illegible]

Приложение А.5 – UML-диаграмма состояний



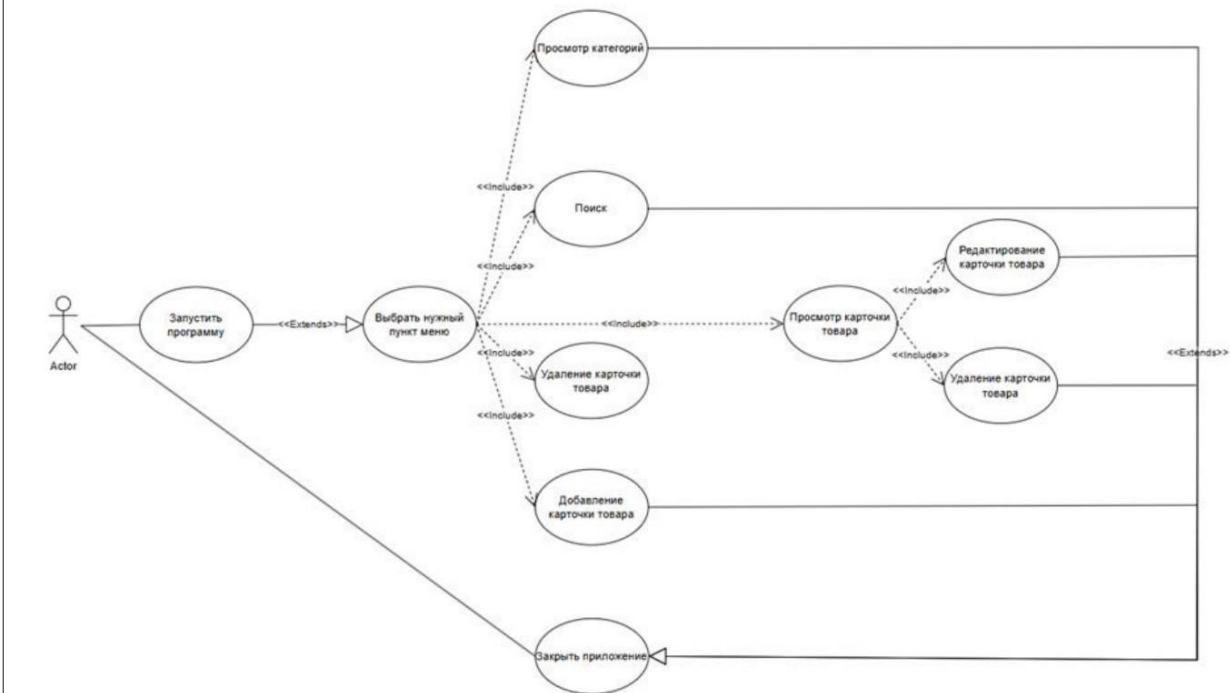
ПГУ 1.090301.01 005 Д16									
Введ. Загл.	Исх. док.	Подпись	Дата	Подписано и датировано			Лист	Место	Масштаб
Рисунки	Утвержденный акт			визы ОГ, подлинный текст					
Текст	Визы			Утвержденный текст			Лист	Место	
Лист	Визы						ФБТ, стр. 210807		
Итого	Утвержденный акт								

Приложение А.6 – UML-диаграмма развертывания



ПГУ 1.090301.01 006 Д19									
Изм.	Лист	Вариант	Полное	Дата	Приложение входит в состав ПГУ 1.090301.01 006 Д19 – Доклад о результатах				
Изм.	Лист	Вариант	Полное	Дата					
Изм.	Лист	Вариант	Полное	Дата					
Изм.	Лист	Вариант	Полное	Дата					
Изм.	Лист	Вариант	Полное	Дата					
Изм.	Лист	Вариант	Полное	Дата					
Изм.	Лист	Вариант	Полное	Дата					
Изм.	Лист	Вариант	Полное	Дата					
Изм.	Лист	Вариант	Полное	Дата					

Приложение А.7. – UML-диаграмма вариантов использования



					ПГУ 1.090301.01 007 Д13				
Имя	Долг	Вид	Полное	Дата	Имя	Долг	Вид	Полное	Дата
Имя	Долг	Вид	Полное	Дата	Имя	Долг	Вид	Полное	Дата
Имя	Долг	Вид	Полное	Дата	Имя	Долг	Вид	Полное	Дата
Имя	Долг	Вид	Полное	Дата	Имя	Долг	Вид	Полное	Дата
Имя	Долг	Вид	Полное	Дата	Имя	Долг	Вид	Полное	Дата
Имя	Долг	Вид	Полное	Дата	Имя	Долг	Вид	Полное	Дата
Имя	Долг	Вид	Полное	Дата	Имя	Долг	Вид	Полное	Дата
Имя	Долг	Вид	Полное	Дата	Имя	Долг	Вид	Полное	Дата
Имя	Долг	Вид	Полное	Дата	Имя	Долг	Вид	Полное	Дата

Приложение Б

(обязательное)

Программа для учёта склада под ОС Android

Листинг программы

					ПГУ 1.090301.21.001 ПЗ	Лист
						71
Изм.	Лист	№ докум.	Подпись	Дата		

Файл DatabaseHelper

```
package com.example.test3

import android.content.ContentValues
import android.content.Context
import android.database.Cursor
import android.database.sqlite.SQLiteDatabase
import android.database.sqlite.SQLiteOpenHelper
import java.text.SimpleDateFormat
import java.util.Date
import java.util.Locale
import java.util.Calendar

class DatabaseHelper(context: Context) : SQLiteOpenHelper(context,
    DATABASE_NAME, null, DATABASE_VERSION) {

    companion object {

        private const val DATABASE_NAME = "Warehouse.db"
        private const val DATABASE_VERSION = 4

        // Название таблиц и колонок
        const val TABLE_WAREHOUSE = "Warehouse"
        const val TABLE_CATEGORIES = "Categories"
        const val COLUMN_ID = "ID"
        const val COLUMN_NAME = "Название"
        const val COLUMN_QUANTITY_IMPORTED =
            "Количество_завезённого"
        const val COLUMN_QUANTITY_REMAINING =
            "Количество_оставшегося"
        const val COLUMN_ARRIVAL_DATE = "ДАТА_ПРИБЫТИЯ"
        const val COLUMN_LAST_SALE_DATE = "ДАТА_ПОСЛЕДНЕЙ_ПРОДАЖИ"
        const val COLUMN_CATEGORY = "Категория"
        const val COLUMN_CATEGORY_NAME = "Название_категории"
```

					ПГУ 1.090301.21.001 ПЗ	Лист
						72
Изм.	Лист	№ докум.	Подпись	Дата		

```

const val TABLE_TRANSACTIONS = "transactions"
    const val COLUMN_TRANSACTION_ID = "transaction_id"
    const val COLUMN_PRODUCT_ID = "product_id"
    const val COLUMN_QUANTITY = "quantity"
    const val COLUMN_DATE = "date"
    const val COLUMN_TYPE = "type" // "sale" или "usage"
}

override fun onCreate(db: SQLiteDatabase) {
    // Создаем все таблицы внутри транзакции
    db.beginTransaction()
    try {
        // 1. Таблица Warehouse
        db.execSQL("""
            CREATE TABLE $TABLE_WAREHOUSE (
                $COLUMN_ID INTEGER PRIMARY KEY AUTOINCREMENT,
                $COLUMN_NAME TEXT,
                $COLUMN_QUANTITY_IMPORTED INTEGER,
                $COLUMN_QUANTITY_REMAINING INTEGER,
                $COLUMN_ARRIVAL_DATE TEXT NOT NULL,
                $COLUMN_LAST_SALE_DATE TEXT NOT NULL,
                $COLUMN_CATEGORY TEXT
            )
        """).trimIndent())
    db.execSQL("""
        CREATE TABLE $TABLE_CATEGORIES (
            $COLUMN_ID INTEGER PRIMARY KEY AUTOINCREMENT,
            $COLUMN_CATEGORY_NAME TEXT NOT NULL UNIQUE
        )
    """).trimIndent())
    db.execSQL("""
        CREATE TABLE $TABLE_TRANSACTIONS (
            $COLUMN_TRANSACTION_ID INTEGER PRIMARY KEY AUTOINCREMENT,
            $COLUMN_PRODUCT_ID INTEGER,
            $COLUMN_QUANTITY INTEGER,
            $COLUMN_DATE TEXT,

```

```

        FOREIGN KEY($COLUMN_PRODUCT_ID) REFERENCES
$TABLE_WAREHOUSE($COLUMN_ID)
    )
    """.trimIndent())

    // Добавляем начальные данные
    addInitialCategories(db)
    db.setTransactionSuccessful()
} finally {
    db.endTransaction()
}
}

override fun onUpgrade(db: SQLiteDatabase, oldVersion: Int,
newVersion: Int) {
    when (oldVersion) {
        1 -> {
            // Миграция с версии 1 на 2 - создаем таблицу Categories
            db.execSQL("""
            CREATE TABLE $TABLE_CATEGORIES (
                $COLUMN_ID INTEGER PRIMARY KEY AUTOINCREMENT,
                $COLUMN_CATEGORY_NAME TEXT NOT NULL UNIQUE
            )
            """.trimIndent())
            addInitialCategories(db)
        }
        2, 3 -> {
            // Миграция на версию 4 - создаем таблицу Transactions
            db.execSQL("""
            CREATE TABLE IF NOT EXISTS $TABLE_TRANSACTIONS (
                $COLUMN_TRANSACTION_ID INTEGER PRIMARY KEY
                AUTOINCREMENT,
                $COLUMN_PRODUCT_ID INTEGER,
                $COLUMN_QUANTITY INTEGER,
                $COLUMN_DATE TEXT,

```

```

FOREIGN KEY($COLUMN_PRODUCT_ID) REFERENCES
$TABLE_WAREHOUSE($COLUMN_ID)
    )
    """.trimIndent())
    }
}

private fun addInitialCategories(db: SQLiteDatabase) {
    val categories = listOf("Электроника", "Одежда", "Продукты")
    categories.forEach { category ->
        db.insert(TABLE_CATEGORIES, null, ContentValues().apply {
            put(COLUMN_CATEGORY_NAME, category)
        })
    }
}

private fun createWarehouseTable(db: SQLiteDatabase) {
    db.execSQL("""
        CREATE TABLE $TABLE_WAREHOUSE (
            $COLUMN_ID INTEGER PRIMARY KEY AUTOINCREMENT,
            $COLUMN_NAME TEXT,
            $COLUMN_QUANTITY_IMPORTED INTEGER,
            $COLUMN_QUANTITY_REMAINING INTEGER,
            $COLUMN_ARRIVAL_DATE TEXT NOT NULL,
            $COLUMN_LAST_SALE_DATE TEXT NOT NULL,
            $COLUMN_CATEGORY TEXT
        )
    """.trimIndent())
}

private fun createCategoriesTable(db: SQLiteDatabase) {
    db.execSQL("""
        CREATE TABLE $TABLE_CATEGORIES (
            $COLUMN_ID INTEGER PRIMARY KEY AUTOINCREMENT,
            $COLUMN_CATEGORY_NAME TEXT NOT NULL UNIQUE
        )
    """.trimIndent())
}

```

```

private fun createTransactionsTable(db: SQLiteDatabase) {
    db.execSQL("""
        CREATE TABLE $TABLE_TRANSACTIONS (
            $COLUMN_TRANSACTION_ID INTEGER PRIMARY KEY
AUTOINCREMENT,
            $COLUMN_PRODUCT_ID INTEGER,
            $COLUMN_QUANTITY INTEGER,
            $COLUMN_DATE TEXT,
            FOREIGN KEY($COLUMN_PRODUCT_ID) REFERENCES
$TABLE_WAREHOUSE($COLUMN_ID)
        )
    """).trimIndent()
}

fun addProduct(name: String, quantityImported: Int,
quantityRemaining: Int,
            arrivalDate: String, lastSaleDate: String, category:
String): Long {
    val db = this.writableDatabase
    val values = ContentValues().apply {
        put(COLUMN_NAME, name)
        put(COLUMN_QUANTITY_IMPORTED, quantityImported)
        put(COLUMN_QUANTITY_REMAINING, quantityRemaining)
        put(COLUMN_ARRIVAL_DATE, arrivalDate)
        put(COLUMN_LAST_SALE_DATE, lastSaleDate)
        put(COLUMN_CATEGORY, category)
    }
    return db.insert(TABLE_WAREHOUSE, null, values)
}

fun deleteProduct(id: Long): Boolean {
    val db = this.writableDatabase
    val result = db.delete(TABLE_WAREHOUSE, "$COLUMN_ID = ?",
arrayOf(id.toString())) > 0
    db.close()
    return result
}

```

```

// Метод для удаления всех товаров в категории
fun deleteProductsByCategory(category: String): Boolean {
    val db = this.writableDatabase
    val result = db.delete(TABLE_WAREHOUSE, "$COLUMN_CATEGORY = ?",
arrayOf(category)) > 0
    db.close()
    return result
}

// Метод для удаления всех товаров
fun deleteAllProducts(): Boolean {
    val db = this.writableDatabase
    val result = db.delete(TABLE_WAREHOUSE, null, null) > 0
    db.close()
    return result
}

// Метод для получения всех товаров
fun getAllProducts(): Cursor {
    val db = this.readableDatabase
    return db.rawQuery("SELECT * FROM $TABLE_WAREHOUSE", null)
}

// Метод для получения товаров по категории
fun getProductsByCategory(category: String): Cursor {
    val db = this.readableDatabase
    return db.rawQuery("SELECT * FROM $TABLE_WAREHOUSE WHERE
$COLUMN_CATEGORY = ?", arrayOf(category))
}

// Метод для получения товара по ID
fun getProductById(id: Int): Cursor {
    val db = this.readableDatabase
    return db.rawQuery("SELECT * FROM $TABLE_WAREHOUSE WHERE
$COLUMN_ID = ?", arrayOf(id.toString()))
}

```

```

// Метод для получения всех категорий
fun getAllCategories(): Cursor {
    val db = this.readableDatabase
    return db.rawQuery("SELECT * FROM $TABLE_CATEGORIES", null)
}

// Метод для добавления новой категории
fun addCategory(category: String): Boolean {
    // Проверяем, есть ли уже такая категория
    val db = this.readableDatabase
    val cursor = db.query(
        TABLE_CATEGORIES,
        arrayOf(COLUMN_ID),
        "$COLUMN_CATEGORY_NAME = ?",
        arrayOf(category),
        null, null, null
    )
    val exists = cursor.count > 0
    cursor.close()

    if (exists) return true // Категория уже есть
    // Добавляем новую категорию
    val values = ContentValues().apply {
        put(COLUMN_CATEGORY_NAME, category)
    }
    return db.insert(TABLE_CATEGORIES, null, values) != -1L
}

// Метод для проверки существования товаров в категории
fun hasProductsInCategory(category: String): Boolean {
    val db = this.readableDatabase
    val cursor = db.rawQuery(
        "SELECT COUNT(*) FROM $TABLE_WAREHOUSE WHERE $COLUMN_CATEGORY = ?",
        arrayOf(category)
    )

```

```

cursor.moveToFirst()
    val count = cursor.getInt(0)
    cursor.close()
    return count > 0
}

fun updateProduct(
    id: Long,
    name: String,
    quantityImported: Int,
    quantityRemaining: Int,
    arrivalDate: String,
    lastSaleDate: String,
    category: String
): Boolean {
    val db = this.writableDatabase
    val values = ContentValues().apply {
        put(COLUMN_NAME, name)
        put(COLUMN_QUANTITY_IMPORTED, quantityImported)
        put(COLUMN_QUANTITY_REMAINING, quantityRemaining)
        put(COLUMN_ARRIVAL_DATE, arrivalDate)
        put(COLUMN_LAST_SALE_DATE, lastSaleDate)
        put(COLUMN_CATEGORY, category)
    }
    val result = db.update(
        TABLE_WAREHOUSE,
        values,
        "$COLUMN_ID = ?",
        arrayOf(id.toString())
    ) > 0

    db.close()
    return result
}

```

```

fun searchProducts(query: String): Cursor {
    return readableDatabase.rawQuery(
        "SELECT * FROM $TABLE_WAREHOUSE WHERE $COLUMN_NAME LIKE ? OR
        $COLUMN_CATEGORY LIKE ?",
        arrayOf("%$query%", "%$query%")
    )
}

fun addTransaction(productId: Long, quantity: Int, date: String):
Long {
    val values = ContentValues().apply {
        put(COLUMN_PRODUCT_ID, productId)
        put(COLUMN_QUANTITY, quantity)
        put(COLUMN_DATE, convertDateToDbFormat(date))
    }
    return writableDatabase.insert(TABLE_TRANSACTIONS, null, values)
}

private fun convertDateToDbFormat(dateStr: String): String {
    return try {
        val inputFormat = SimpleDateFormat("dd.MM.yyyy",
        Locale.getDefault())
        val outputFormat = SimpleDateFormat("yyyy-MM-dd",
        Locale.getDefault())
        val date = inputFormat.parse(dateStr)
        outputFormat.format(date ?: Date())
    } catch (e: Exception) {
        SimpleDateFormat("yyyy-MM-dd",
        Locale.getDefault()).format(Date())
    }
}

fun getTransactionsForProduct(productId: Long): Cursor {
    return readableDatabase.query(
        TABLE_TRANSACTIONS,
        null,
        "$COLUMN_PRODUCT_ID = ?",
        arrayOf(productId.toString()),

```

```

        null, null,
        "$COLUMN_DATE ASC"
    )
}
}

```

Файл EditProductActivity

```

package com.example.test3

import android.content.Intent
import android.database.Cursor
import android.os.Bundle
import android.view.View
import android.widget.AdapterView
import android.widget.AdapterView.OnItemClickListener
import android.widget.ArrayAdapter
import android.widget.ListView
import androidx.appcompat.app.AppCompatActivity

class CategoryActivity : AppCompatActivity() {

    private lateinit var dbHelper: DatabaseHelper

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_category)

        dbHelper = DatabaseHelper(this)
        // Получаем все категории из базы данных
        val categoriesCursor: Cursor = dbHelper.getAllCategories()
        val categories = mutableListOf<String>()
        if (categoriesCursor.moveToFirst()) {
            do {
                val categoryName =
                    categoriesCursor.getString(categoriesCursor.getColumnIndexOrThrow(DatabaseHelper.COLUMN_CATEGORY_NAME))
                categories.add(categoryName)
            } while (categoriesCursor.moveToNext())
        }
    }
}

```

					ПГУ 1.090301.21.001 ПЗ	Лист
						81
Изм.	Лист	№ докум.	Подпись	Дата		

```

}

        categoriesCursor.close()

        // Отображаем категории в ListView
        val listViewCategories: ListView =
findViewById(R.id.listViewCategories)
        val adapter = ArrayAdapter(this,
android.R.layout.simple_list_item_1, categories)
        listViewCategories.adapter = adapter

        // Обработчик нажатия на категорию
        listViewCategories.setOnItemClickListener =
AdapterView.OnItemClickListener { _, _, position, _ ->
            val selectedCategory = categories[position]
            val intent = Intent(this,
MainActivity::class.java).apply {
                putExtra("SELECTED_CATEGORY", selectedCategory)
            }
            startActivity(intent)
        }
    }
}

```

Файл activity_product_detail.xml

```

<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".ProductDetailActivity">

    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"

```

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090301.21.001 ПЗ

Лист

82

```
android:orientation="vertical"
android:padding="16dp">
```

```
<TextView
    android:id="@+id/productNameDetail"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Название товара"
    android:textSize="24sp"
    android:textStyle="bold" />
```

```
<TextView
    android:id="@+id/productQuantityImportedDetail"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@id/productNameDetail"
    android:text="Завезено: "
    android:textSize="18sp"
    android:layout_marginTop="16dp" />
```

```
TextView
    android:id="@+id/productQuantityRemainingDetail"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@id/productQuantityImportedDetail"
    android:text="Осталось: "
    android:textSize="18sp"
    android:layout_marginTop="8dp" />
```

```
<TextView
    android:id="@+id/productArrivalDateDetail"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Дата прибытия: "
    android:layout_below="@id/productQuantityRemainingDetail"
    android:textSize="18sp"
    android:layout_marginTop="8dp" />
```

					ПГУ 1.090301.21.001 ПЗ	Лист
						83
Изм.	Лист	№ докум.	Подпись	Дата		

```

<TextView
    android:id="@+id/productLastSaleDateDetail"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@id/productArrivalDateDetail"
    android:text="Дата последней продажи: "
    android:textSize="18sp"
    android:layout_marginTop="8dp" />

<TextView
    android:id="@+id/productCategoryDetail"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@id/productLastSaleDateDetail"
    android:text="Категория: Электроника"
    android:textSize="18sp"
    android:layout_marginTop="8dp" />

<Button
    android:id="@+id/btnDeleteProduct"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@id/productCategoryDetail"
    android:text="Удалить товар"
    android:layout_marginTop="24dp"
    android:textColor="#FFFFFF"/>

<Button
    android:id="@+id/btnEditProduct"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@id/productCategoryDetail"
    android:layout_marginLeft="30dp"
    android:layout_marginTop="24dp"
    android:layout_toRightOf="@id/btnDeleteProduct"
    android:text="Редактировать"
    android:textColor="#FFFFFF" />

```

```

<com.github.mikephil.charting.charts.LineChart
    android:layout_below="@id/btnEditProduct"
    android:id="@+id/salesChart"
    android:layout_width="match_parent"
    android:layout_height="300dp"
    android:layout_marginTop="16dp"/>

<Button
    android:id="@+id/btnRecordSale"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@id/salesChart"
    android:layout_marginTop="16dp"
    android:text="Записать продажу"
    android:textColor="#FFFFFF"/>

</RelativeLayout>
</androidx.constraintlayout.widget.ConstraintLayout>

```

					ПГУ 1.090301.21.001 ПЗ	Лист
						85
Изм.	Лист	№ докум.	Подпись	Дата		

Приложение В

(обязательное)

Программа для учёта склада под ОС Android

Презентация

					ПГУ 1.090301.21.001 ПЗ	Лист
						86
Изм.	Лист	№ докум.	Подпись	Дата		



Программа для учёта склада под ОС Android

Автор работы:

Хорошильцев Егор Александрович

Руководитель:

доцент Токарев Андрей Николаевич

Цели и применение

- Программы складского учёта – неотъемлемое средство для ведения записей о товарах, материалах и прочих заказах, которые делает предполагаемый заказчик.

Данный тип софта включает в себя следующие цели для выполнения:

1. Оптимизация трудовых процессов

Ведение вручную подобного рода данных может оказаться проблематичным: начиная от того, что заполнение каких-нибудь заметок или условных Excel-таблиц может занимать немало времени, и заканчивая увеличивающейся возможностью критичной ошибки в процессе работы. Программы складского учёта упрощают работу человека со всех сторон в его роде деятельности.

2. Учёт операций

Приложение содержит встроенные функции для вычисления остатков на складе.

3. Безопасность данных

В основном все программы складского учёта обладают своими базами данных, которые не так просто случайно удалить или испортить: они хранятся в корневых папках приложения, так что обычный пользователь не сможет так просто ненамерочно их испортить.

4. Мобильность

Так как приложение разработано под ОС Android, а смартфоны есть у большинства людей на сегодняшний день, то пользоваться софтом можно в разных удалённых точках без надобности в ПК.

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090301.21.001 ПЗ

Лист

87



Актуальность

В условиях роста мобильности бизнеса складские приложения для Android становятся ключевым инструментом для оперативного управления запасами, благодаря их доступности и мобильности. Особенно востребованы в малом и среднем бизнесе, где важно сокращать издержки без покупки специализированного оборудования. Поддержка офлайн-режима делает их незаменимыми на складах с нестабильным интернетом.

Обзор аналогов

Team: Торговля + Склад + Производство



Team: Торговля + Склад + Производство – это комплексное приложение для учёта товаров, услуг, продаж, закупок, производства и бухгалтерии бизнеса на 1С. Имеет множество разных функций, которые не имеются в других приложениях.

Warehouse Management



Warehouse Management – приложение для учёта и анализа склада от компании Microsoft, которое входит в состав Dynamic 365 Supply Chain Management. Предназначено для автоматизации складских процессов.

Изм.	Лист	№ докум.	Подпись	Дата

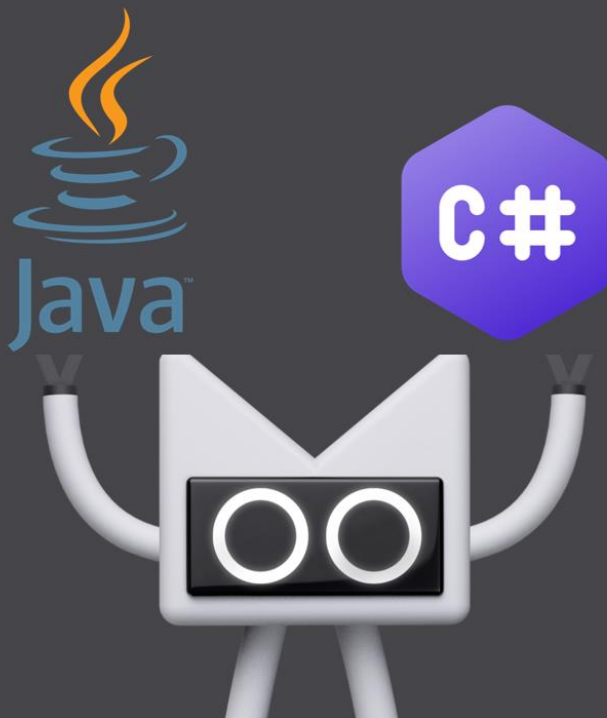
ПГУ 1.090301.21.001 ПЗ

Лист

88

Выбор языка программирования

- Java – один из самых популярных языков программирования в мире, занимает четвертое место в списке популярности именитого сайта [tiobe.com](https://www.tiobe.com).
- Kotlin – язык программирования, основанный на Java и вместе с тем совместимый с ней. Хотя и не входит в топ популярных языков программирования, но вместе с тем больше всех используется для создания мобильных приложений.
- C# + .NET MAUI/Xamarin – объектно-ориентированный язык программирования от компании Майкрософт. Также является одним из популярнейших языков программирования в мире, занимая место после Java. *Xamarin* – фреймворк для разработки мобильных приложений под iOS и Android, появившийся в 2011 году и использовавшийся до 2022 года. .NET MAUI – апгрейд *Xamarin*, начавший свою работу с .NET 6+.



Выбор IDE и СУБД

- В качестве интегрированной среды разработки была выбрана Android Studio, а в качестве СУБД - SQLite

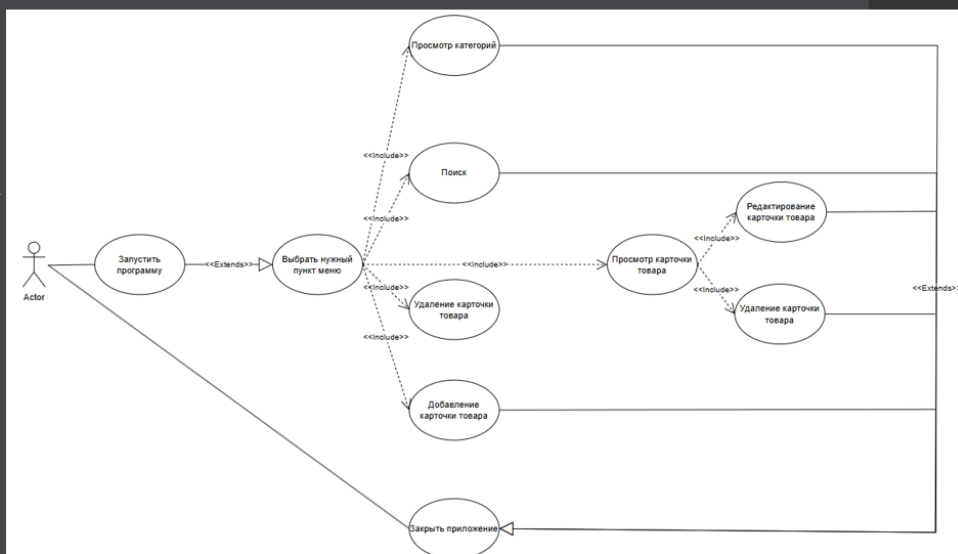


Изм.	Лист	№ докум.	Подпись	Дата

Диаграмма вариантов использования

• Пример сценария:

1. открытие приложения;
2. добавление товаров в базу данных;
3. указание даты прихода;
4. редактирование данных в случае продаж;
5. просмотр остатков после рабочего дня.



Заключение

- Разработано приложение для учёта склада под ОС Android.
- Данное приложение может быть доработано и улучшено в дальнейшем путём добавления функций сортировки, фильтров, возможности экспорта данных в Excel- таблицы, возможности добавления сканера штрих-кодов.



Спасибо за внимание!

					ПГУ 1.090301.21.001 ПЗ	Лист
						91
Изм.	Лист	№ докум.	Подпись	Дата		