Протокол анализа Отчета подобия Научным руководителем

Заявляю, что я ознакомился(-ась) с Полным отчетом подобия, который был сгенерирован Системой выявления и предотвращения плагиата в отношении работы:

Автор: Еркинбекова Жанель

Название: Еркинбековой Жанель

Координатор:Гульнара Омарова

Коэффициент подобия 1:11,3

Коэффициент подобия 2:6,2

Тревога:14

После анализа Отчета подобия констатирую следующее:

☐ обнаруженные в работе заимствования являются добросовестными и не обладают признаками плагиата. В связи с чем, признаю работу самостоятельной и допускаю ее к защите;

☐ обнаруженные в работе заимствования не обладают признаками плагиата, но их чрезмерное количество вызывает сомнения в отношении ценности работы по существу и отсутствием самостоятельности ее автора. В связи с чем, работа должна быть вновь отредактирована с целью ограничения заимствований;

☐ обнаруженные в работе заимствования являются недобросовестными и обладают признаками плагиата, или в ней содержатся преднамеренные искажения текста, указывающие на попытки сокрытия недобросовестных заимствований. В связи с чем, не допускаю работу к защите.

Обоснование:

.....
.....
.....
.....
.....
.....

16.05.19.
.....

Дата

.....

Подпись Научного руководителя

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН
СӘТБАЕВ УНИВЕРСИТЕТІ

ОТЗЫВ

НАУЧНОГО РУКОВОДИТЕЛЯ

на дипломный проект
(наименование вида работы)
Еркинбекова Жанель
(Ф.И.О. обучающегося)
5В070400 – Вычислительная техника и программное обеспечение
(шифр и наименование специальности)

Тема: «Разработка мобильного приложения для автоматизации процессов мерчендайзинга на платформе Android»

Актуальность данного проекта заключается в том, что он упрощает работу мерчендайзерам. В дипломном проекте позволяет сократить время посещения торговых точек, при этом многократно улучшить качество собираемых данных и таким образом сделать их пригодными для последующего анализа и принятия управленческих решений. Дипломный проект состоит из 4-х глав, включая введение, заключение и приложение.

В первой главе представлено общее описание. Проведен анализ и описание предметной области, а также раскрыты цели разработки системы.

Во второй главе представлены технологии для создания программного обеспечения. Проведено сравнение всех используемых технологий для разработки мобильного приложения и выбраны наилучшие для данного ПО. Также описано техническое задание.

В третьей главе описана проектная часть дипломного проекта. Представлены логическая, физическая модели, подробно описаны сущности и связи между ними. Помимо этого, имеется диаграмма вариантов использования, которая демонстрирует различные способы взаимодействия пользователя с системой.

В четвертой главе представлено проектирование пользовательского интерфейса. Наглядно продемонстрированы и описаны все окна мобильного приложения.

В целом данный дипломный проект выполнен с учетом всех требований, предъявляемых к дипломному проекту по специальности 5В070400 – «Вычислительная техника и программное обеспечение», студентка Еркинбекова Ж. рекомендована к защите дипломного проекта и заслуживает присвоения академической степени «бакалавра» по специальности 5В070400 – «Вычислительная техника и программное обеспечение».

Научный руководитель
лектор
(должность, уч. степень, звание)
Омарова Г. А.
(подпись)
«16» мая 2019г.

АКТ

о внедрении результатов дипломного проекта
студента (ки) специальности 5В070400 – «Вычислительная техника и программное обеспечение» СӘТБАЕВ УНИВЕРСИТЕТІ

Еркинбековой Жанель Айдаровны
(имя, фамилия, отчество студента (ки))

на тему "Разработка мобильного приложения для автоматизации процессов мерчендайзинга на платформе Android".
(тема выпускной квалификационной работы)

Автоматизированная система процессов мерчендайзинга на платформе Android, разработанная в рамках дипломного проекта студенткой группы ВПб-15-1р СӘТБАЕВ УНИВЕРСИТЕТІ Еркинбековой Жанель Айдаровны, соотвествует всем требованиям, принята в промышленную эксплуатацию и активно используется в компании Raimbek Group, клиента компании ТОО "Vela IT"

Директор «Vela IT»
(должность, ученая степень, звание)

_____ Жиленко А.С.
(подпись)

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН

СӘТБАЕВ УНИВЕРСИТЕТІ

Институт информационных и телекомм уникационных технологий

Кафедра "Программная инженерия"

Еркинбекова Ж.А.

Разработка мобильного приложения для автоматизации процессов мерчендайзинга на платформе Android

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к дипломному проекту

Специальность 5В070400 – Вычислительная техника и программное обеспечение

Алматы 2019

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН

СӘТБАЕВ УНИВЕРСИТЕТІ

Институт информационных и телекоммуникационных технологий

Кафедра "Программная инженерия"

ДОПУЩЕН К ЗАЩИТЕ
Заведующий кафедрой ПИ
канд. техн. наук, доцент,
ассистент-профессор
_____ Р. Юнусов
"_16_" _____ 2019 г.

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к дипломному проекту

На тему: " Разработка мобильного приложения для автоматизации процессов мерчендайзинга на платформеAndroid "

по специальности 5В070400 – Вычислительная техника и программное обеспечение

Выполнила
Еркинбекова Ж.А.

Научный руководитель
лектор
_____ Г.А. Омарова
"_16_" _____2019 г.

Алматы 2019

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН

СӘТБАЕВ УНИВЕРСИТЕТІ

Институт информационных и телекоммуникационных технологий

Кафедра "Программная инженерия"

5В070400 – Вычислительная техника и программное обеспечение

УТВЕРЖДАЮ
Заведующий кафедрой ПИ
канд. техн. наук, доцент,
ассистент-профессор
_____ Р. Юнусов
"16" мая 2019 г.

ЗАДАНИЕ
на выполнение дипломного проекта

Обучающемуся *Еркинбековой Жанель Айдаровне*.
Тема: *Разработка мобильного приложения для автоматизации процессов мерчендайзинга на платформе Android*.
Утверждена *приказом проректора по академической работе № 1162 - б от "16" октября 2018 г.*
Срок сдачи законченного проекта *"17" мая 2019 г.*
Исходные данные к дипломному проекту: *Паспорт проекта, техническая документация по применению технологии, техническое задание, описание БД для хранения информации в виде ER-диаграммы.*.
Перечень подлежащих разработке в дипломном проекте вопросов:
а) разработка классовой модели реляционной базы данных SQLite согласно представленной диаграмме;.
б) реализация многозвенной архитектуры мобильного приложения в соответствие с концепцией MVVM;
в) проектирование и разработка пользовательского интерфейса;
.
г) разработка, отладка, тестирование программного комплекса..
Перечень графического материала (с точным указанием обязательных чертежей): *представлены 15 слайда презентации.*.
Рекомендуемая основная литература: *из 11 наименований.*

ГРАФИК

подготовки дипломного проекта

Наименование разделов, перечень разрабатываемых вопросов	Сроки представления научному руководителю и консультантам	Примечание
1. Анализ предметной области, разработка технического задания на проектирование программного комплекса	18.10.18	примечаний нет
2. Реализация классовой структуры модели базы	2.11.18	примечаний нет
3. Разработка пользовательского интерфейса	28.11.18	примечаний нет
4. Разработка загрузки контрагентов, каталога компании, детальной информации товара.	24.12.18 - 3.02.19	примечаний нет
5. Разработка фотоотчета, последней страницы приложения. Тестирование приложения	4.02.19 - 14.03.19	примечаний нет
6. Написание пояснительной записки к дипломному проекту	01.05.19 - 06.05.19	примечаний нет

Подписи

консультантов и нормоконтролера на законченный дипломный проект
с указанием относящихся к ним разделов проекта

Наименования разделов	Консультанты, И.О.Ф. (уч. степень, звание)	Дата подписания	Подпись
Программное обеспечение	З.М.Өмірбекова ассистент	6.05.19	
Нормоконтролер	Г.А.Омарова лектор	6.05.19	

Научный руководитель _____ Г.А. Омарова

Задание принял к исполнению обучающийся _____ Ж.А. Еркинбекова

Дата "6" мая 2019 г.

АННОТАЦИЯ

Этот дипломный проект предназначен для разработки приложения на платформе Android в целях автоматизации работы мерчендайзинга. Разработанное мобильное приложение позволит работнику быстро сформировать отчет. Удобство данного решения состоит в сокращении времени посещения торговых точек, сбора точной и оперативной информации, избавлении от бумажных анкет. Система позволяет контролировать мерчендайзера, который работает непосредственно на выезде в торговые точки. Автоматизированный мерчендайзинг многократно улучшает качество собираемых данных и таким образом делает их пригодными для последующего анализа и принятия управленческих решений.

В этой пояснительной записке описывается разработка реляционной базы данных SQLite с технологией картирования (ORM) – ORMlite, использование архитектуры MVVM, библиотеки для работы с REST API – Retrofit. ПО разрабатывалось в интегрированной среде Android Studio с использованием языка объектного программирования Java.

АҢДАТПА

Бұл диплом жобасы мерчандайзинг қызметін автоматтандыру үшін бағдарламалық жасақтама әзірлеуге арналған. Әзірленген мобильді қосымша қызметкерге тез арада есеп шығаруға мүмкіндік береді. Бұл шешімнің ыңғайлылығы сауда нүктесіге бару уақытын азайтуда, нақты және уақтылы ақпарат жинауда, қағаз сауалнамаларынан құтылуда тұрады. Жүйе дүкенге шығуға тікелей жұмыс істейтін мерчензайдерді басқаруға мүмкіндік береді. Автоматтандырылған мерчендайзинг жиналған деректердің сапасын бірнеше рет жақсартады және оларды болашақ талдау және басқару шешімдеріне қолайлы етеді.

Бұл түсіндірме жазбада жүйесін дамыту үшін SQLite реляциялық деректер база мен ORMlite картографиялау технология дамуы, MVVM архитектурасын пайдалануы, Retrofit REST API-мен жұмысы сипатталған. Бағдарламалық жасақтама Android Studio бағдарламасының Java бағдарламалау тілінің көмегімен жасалған.

ANNOTATION

This diploma project is intended to develop software to automate the work of merchandising. The developed mobile application will allow the employee to quickly generate a report. The convenience of this solution is to reduce the time to visit stores, collect accurate and operative information, getting rid of paper questionnaires. The system allows you to control the merchandiser, which works directly on the store. Automated merchandising repeatedly improves the quality of the data collected and thus makes them suitable for further analysis and management decisions.

This explanatory note describes the development of SQLite relational database with mapping technology (ORM) – ORMlite, using MVVM architecture, a library for working with REST API – Retrofit. The software was developed in the integrated environment of Android Studio using the object programming language Java.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В настоящее время с широким разнообразием брендов заставить потребителя приобрести именно ваш товар не так уж и легко. Витрины стали местом борьбы за выбор покупателей. Правильная выкладка и подача продукта на полках магазинов с целью привлечения внимания клиента для дальнейшей покупки этого товара является реализацией грамотного мерчендайзинга.

С усиливающейся конкурентной борьбой компании пытаются найти как можно больше данных о соперниках, дабы предложить покупателю что-то лучше. Такая информация способствуют эффективному продвижению продукта.

Торговые представители занимаются сбором данных, на которых основывается анализ работы конкурента, планирование продаж и т.д. К этому относится мониторинг цен, фейсинг, текущий остаток товара, наличие рекламных материалов, выкладка. Для максимального контроля и сбора информации требуется автоматизация мерчендайзинга.

Автоматизация работы мерчендайзера – это единая система сбора, анализа, обработки, контроля и передачи мерчендайзинговых данных.

Основными задачами автоматизации мерчендайзинга являются:

- комфортный, эффективный контроль сортаментом в торговых точках;
- формирование отчета по проделанной работе, собранных данных и проведение последующего анализа;
- повышение точности и оперативности полученной информации;
- отсутствие бумажных анкет.

Таким образом, автоматизированный мерчендайзинг позволяет сократить время посещения торговых точек, при этом многократно улучшить качество собираемых данных и таким образом сделать их пригодными для последующего анализа и принятия управленческих решений.

Актуальность данного проекта заключается в том, что он упрощает работу мерчендайзерам, облегчает процесс формирования отчетов, минимизирует время визита торговых точек, создает фотоотчет с невозможностью его фальсификации, фиксирует место, дату и время создания фотоотчета, повышает эффективность контроля и планирование рабочего процесса. Кроме того, приложение может быть модернизировано и расширено с появлением новых требований, а также использовано при решении других схожих задач.

1 Общее описание

1.1 Цель разработки системы

Необходимо создать автоматизированную систему работы мерчендайзинга с возможностью просмотра, фильтрации и выбора товара из каталога; фиксации необходимых данных о товаре в электронном виде: наличие, фейсинг, цена, остаток на складе торговой точки, комментарий; создания фотоотчета с фиксацией места, даты и времени; формирования отчета; отправки документа на сервер.

Достоинство данного решения состоит в сокращении времени посещения торговых точек, сбора точной и оперативной информации, избавлении от бумажных анкет. Система позволяет контролировать мерчендайзера, который работает непосредственно на выезде в торговые точки.

Работник заполняет некоторый электронный отчет, вместо бумажных анкет вводимый на карманном персональном компьютере. При этом торговый представитель может добавлять необходимые комментарии о выкладке, наличии или отсутствии товара с дальнейшей оптимизацией и исправлением ошибок для улучшения продвижения продукта.

1.2 Определения, термины и сокращения

В таблице 1 сформулированы все сокращения и термины, используемые в предметной области разрабатываемого проекта, а также специфические термины, связанные с программной реализацией проекта и используемыми технологиями при разработке.

Таблица 1 – Сокращения, термины и их определения

Сокращение или термин	Определение
API	Application Programming Interface
БД	База данных
СУБД	Система управления базами данных
КПК	Карманный персональный компьютер
ПО	Программное обеспечение
IDE	Integrated Development Environment
ПО	Программное обеспечение
JAR	Java ARchive
ORM	Object Relational Mapping

Продолжение таблицы 1

Сокращение или термин	Определение
JSON	JavaScript Object Notation
SQL	Structured Query Language
UML	Unified Modeling Language
URL	Uniform Resource Locator
ER	Entity Relationship

1.3 Предметная область

1.3.1 Происхождение мерчендайзинга

Перед разработкой приложения необходимо изучить предметную область и понять, что же такое “мерчендайзинг”. В процессе изучения, выяснилось, что это слово происходит от английского merchandise, что означает подготовку продукции, какого-либо товара к продаже, сбыту. Мерчендайзинг – это своего рода психология потребителей, которая создает и улучшает систему распределения товаров в соответствии с потребностями людей.

Мерчендайзинг состоит из оценки и выбора продукции и товаров за счет привлекательности упаковки и притягательности дизайна; достоверности информации и маркировки товара; выбора и оценки стиля, формы и способа торговли товаром; общения с потребителями; методов завлечения потенциальных покупателей.

Ежедневно люди ходят в супермаркеты, магазины, торговые центры, на рынок, покупают необходимую им продукцию. С развитием розничной торговли и производства между компаниями возрастает острая борьба за наиболее выгодное место на витрине. Чтобы обогнать конкурентов, заняв лучшую позицию в магазине, применяются различные методы мерчендайзинга.

1.3.2 Определения мерчендайзинга

Мерчендайзинг относительно новое явление на рынке, при всем при том, уже сложилось достаточно разнообразных определений «мерчендайзинга», которые по большей части заимствованы из зарубежной литературы.

1. Оформление продукции и ее расположение в торговой точке, всячески сподвигающее потребителя к спонтанной покупке.

2. Мерчендайзинг – комплекс методов, которые направлены на увеличение объема продаж в торговой точке, включающий оформление торговых полок, прилавков, витрин, распределение продукции в торговом зале.

3. Мерчендайзинг – это ряд мероприятий для построения результативных связей между потребителем и товаром в зонах торговли, способствующий новым приобретениям.

В западных странах уже давно уделяется особое внимание грамотному представлению товара, так как конкуренция только ужесточается. Качество производства и условия примерно одинаковы, поэтому на разнице в цене особо не выиграешь. Так и появился процесс мерчендайзинга, решающий эту проблему, предлагая продать товар более искусно и тонко.

Продукцию надо не просто так привезти в торговую точку, приклеить ценники и продать как можно дороже. Ее необходимо правильно представить покупателям, чтобы они без раздумий захотели купить. Тут важно все: красивая, привлекающая внимание упаковка, реклама, представление продавцам.

1.3.3 Мерчендайзинг и маркетинг

Мерчендайзинг является неотъемлимой часть маркетинга, который традиционно характеризует рыночные отношения. Но вернее было бы сказать, что маркетинг является деятельностью по управлению системы торговли, подготовки продукции для сбыта.

Одними из основных маркетинговых действий являются стимулирование продаж и продвижение.

Продвижение – это целенаправленные, специальные действия, применяемые компанией-продавцом для представления необходимых данных потребителям о своей продукции с воздействием на сознание покупателей в целях продажи им товара. Продвижение включает в себя размещение на упаковке продукции различных инструментов сейлзпромоушн (например, портретов известных актеров, певцов, спортсменов, героев мультфильмов, т.п.), проведение специализированных ивентов непосредственно на месте продаж для различных видов стимуляции сбыта.

Стимулирование продаж характеризуется особенностью привязки мероприятий непосредственно с местами продаж (торговые точки, магазины, супермаркеты) - в этом и заключается главное отличие стимулирования продаж от рекламы.

Персональная продажа – это лдин из видов продвижения продукции, который состоит в устном представление потенциальным покупателям в целях продажи.

Плюсами персональной продажи являются: проведение диалога между клиентами и продавцами, обратная связь, незначительные затраты, возможность

продемонстрировать товар. Однако зачастую ни клиент, ни продавец не слушают, пропускают мимо ушей и вовсе отказываются вступать в диалог.
В совокупности мерчендайзинг и продвижение дают очень хороший результат, стимулирующий потребителей на совершение покупки.

Проще говоря, мерчендайзинг – искусство реализации, сбыта и продажи товара.

1.3.4 Задачи мерчендайзинга

К задачам мерчендайзинга относится:

1) управление и организация торгово-технологическими процессами в торговле. Первое впечатление о магазине у покупателя состаляется за пару секунд, при этом он не должен заметить влияние мерчендайзинга на свой выбор. В магазине потребителю должно быть понятно и без помощи продавца.

2) грамотно продуманная планировка торгового помещения и размещение торгово-холодильного оборудования, удобное для обзора покупателю;

3) реализация правильной выкладки, стимулирующая клиента на покупку товара, создание особой ауры. Учитываются многие факторы: освещение, музыка, запахи;

4) использование результативных технологий продвижения товаров, при которых одни товары стимулирует покупку другого товара. В этом случае надо решать проблему расположения товара-лидера рядом с товаром-новинкой. Например, конфеты кладут рядом с чаем или кофе;

5) обильное использование инструментов рекламной поддержки, ориентированных на повышение внимания покупателей к товару. Как отмечают профессионалы, товар покупается глазами, а не руками;

6) улучшение ценовой политики, создание условия закупок для торговых точек и пластичной ценовой стратегии;

2 Требования к разработке

2.1 Технологии для создания программного обеспечения

2.1.1 Среда разработки

Для разработки Android существует несолько видов IDE приложений на Java: NetBeans, Intellij IDEA, Eclipse, Ansdroid Studio.

NetBeans – это IDE для разработки приложений на PHP, Python, C++, C, Java, JavaScript, и даже Ада. Не самая известная интегрированная среда разработки, разработанная компанией Oracle. Малое количество разработчиков ее используют специально для платформы Android. Из-за концепции «всё в одном» низкое быстродействие относится к недостаткам.

Intellij IDEA – одна из наиболее известных IDE, которая разработана компанией JetBrains, которая позволяет создавать программы на таких языках как Python,PHP, Java, JavaScript, Ruby, Groovy, Scala, C, C++. Более умная и удобная, чем Eclipse. Однако за ее использование приходится платить деньгами и ресурсами вашего компьютера. Разработка в ней практически невозможна на слабых машинах. Хромает производительность.

Eclipse – популярная среда разработки, ориентированная на работу с Java, известна широким выбором внешних модулей, значительно расширяющих функциональность. Не такая удобная, как Intellij IDEA, но зато вполне подойдет для новичка и менее прожорлива. Совсем недавнего была официальной IDE для разработки приложений для Android. Недостатками является нехватка документации, отсутствие единого сообщества разработчиков.

Android Studio – это одна из ответвлений Intellij IDEA, разработанное компанией Google. Стремительно развивающаяся и относительно молодая IDE, ориентированная на разработку приложений под Android. Эта IDE предназначена только для Android, в отличие от Intellij IDEA, которая также используется для разработки серверных приложений. К плюсам Android Studio относится отсутствие необходимости установки дополнительных плагинов.

В течение долгого времени Eclipse являлся предпочтительной средой разработки (IDE) для разработчиков Android. Однако в настоящее время он сталкивается с серьезной конкуренцией со стороны Android Studio от Google. Существует общее мнение, что Android Studio постепенно «затмевает» популярность Eclipse – благодаря его более дружественному характеру. Здесь проведем сравнительное исследование Android Studio и Eclipse, основанное на нескольких ключевых параметрах, и выясним, какой из них выходит на первое место:

1. Пользовательский интерфейс (UI). Для новичков больше подойдет Android Studio, разработка в ней быстрее, понятнее и проще, т.к. она является нативной IDE Android в отличие от Eclipse

2. Инструмент сборки (Gradle vs Apache Ant) – Ant основан на XML и занимает первое место по показателям надежности и производительности. Тем не менее, совершенно новая система сборки Gradle в Android Studio (вместе с Groovy DSL) выглядит более продвинутой и предлагает больше удобств для разработчиков мобильных приложений.

3. Функция дополнения кода – и Eclipse, и Android Studio предлагают достаточно высококачественную функцию автозаполнения. Функция дополнения кода в Android Studio намного лучше благодаря всесторонней поддержке IntelliJ IDEA, которая делает завершение кода менее подверженным ошибкам. Google также разрешает рефакторинг кода в Android Studio. Это дает ему преимущество над Eclipse.

4. Drag-and-Drop – графический интерфейс пользователя (GUI) есть в Android Studio, а в Eclipse нет. В Android Studio это новая функция, но в Eclipse ее отсутствие не имеет значения.

5. Системные требования и стабильность - Eclipse, по сравнению с Android Studio, является гораздо большей IDE. Тем не менее, он обеспечивает более стабильную гарантию производительности, чем Eclipse, а системные требования также ниже.

6. IDE Workflow – это то, где Android Studio превосходит Eclipse. Всякий раз, когда разработчикам/кодировщикам Android-приложений приходится переключаться с одного рабочего пространства на другое, Eclipse следует перезапускать. Если в одном рабочем пространстве имеется более трех проектов, проект необходимо очищать (и среда IDE может перезапускаться) с частыми интервалами. Копирование проектов в рабочих пространствах в Eclipse также может быть сложным, особенно для тех, кто не знаком с разработкой приложений для Android. Поток «Project and Module» (в большинстве случаев одно приложение будет составлять модуль) в Android Studio определенно менее сложен и более удобен для пользователя. Также могут быть отдельные модули для скриптов сборки и библиотек.

7. Зависимости (Dependencies) – еще один тупик в Android Studio против Eclipse. Разработчики и кодеры мобильных приложений должны создавать ссылки на сторонние файлы JAR, работая с любой из двух IDE. Эти JAR-файлы остаются в каталоге «libs» - еще одна функция, которую совместно используют Eclipse и Android Studio. Если разработчик переключается с Eclipse на Android Studio, все JAR-зависимости должны быть добавлены в новые файлы сборки Gradle.

8. Поддержка облачной платформы – Android Studio поставляется с собственной поддержкой Google Cloud. Еще раз, однако, это не дает ему каких-либо существенных дополнительных преимуществ - поскольку Eclipse имеет плагин Google, который служит точно такой же цели. Поддержка внутреннего сервера от Google удобна для тестирования мобильных приложений и общих задач интеграции приложений. Если вам не очень удобно работать с плагинами IDE, тогда Android Studio лучший выбор.

9. Фактор скорости – в этом отношении Eclipse проигрывает Android Studio. В среднем на сборку окончательных версий проектов уходит около 2-3 минут. Выполнение той же задачи в Android Studio занимает не более 40 секунд (даже для кодов, которые являются относительно длинными и сложными).

10. Тестирование и отладка приложений. Надлежащее тестирование приложений перед выпуском является большой проблемой разработчиков. Именно благодаря этому функциональность юнит-тестов Android Studio получила яркую обратную связь. Настроив тестовые классы и включив их в конфигурацию запуска проектов, можно легко обнаруживать и устранять ошибки программы в Android Studio. Модульные тесты могут быть запущены, когда приложения находятся в стадии сборки (это гарантирует, что ошибки не вступят в производственную фазу). Eclipse не имеет аналогичного инструмента для удобного тестирования приложений. Это еще один недостаток в более старой IDE.

Рассмотрев все за и против обеих сред, можно с легкостью сделать выбор в пользу разработки приложения в Android Studio.

2.1.2 Базы данных

Есть несколько баз данных для разработки мобильного приложения Android: UnQLite, Berkeley DB, Couchbase Lite, LevelDB, SQLite, Realm. Самые популярные из всех SQLite и Realm, их и будем сравнивать.

SQLite – это встроенная база данных, разработанная, традиционно используемая в Android для хранения разных типов данных. При это эта база данных бесплатна и в открытом доступе. Основная идея SQLite – избавиться от сервер-клиентской архитектуры и хранить всю информацию о приложении непосредственно на мобильном устройстве. SQLite представляет гибкость, надежность, наличие хорошей документации и большое количество образцов. Таким образом, большинство разработчиков Android уже более 20 лет используют его для проектов.

Однако совсем недавно популярной стала новая база данных Realm, которую определяют как полноценную альтернативу SQLite. Собственное ядро Realm написано на C++. Realm – это система управления нереляционной базой данных. Realm в основном используется для разработки мобильных приложений. Чтобы понять, почему Realm стал настолько популярным и востребованным, надо более тщательно рассмотреть общие проблемы, с которыми сталкивается молодой разработчик, когда он впервые использует SQLite. После выхода Realm тысячи разработчиков почувствовали его преимущества перед SQLite, потому что:

1. База данных Realm чрезвычайно проста по сравнению с SQLite. Строчек кода при использовании Realm значительно меньше, чем при использовании SQLite.

2. Скорость. Как мы уже заметили, разработка Realm для Android действительно очень быстрая. В некоторых случаях он работал даже быстрее, чем чистый SQLite.

3. Дополнительные функции. Поскольку Realm является гораздо более молодым инструментом по сравнению с SQLite, он может похвастаться множеством новых функций, которые являются отличным бонусом для всех разработчиков. Например, вы можете получить поддержку JSON, шифрование и свободный API при использовании Realm в проектах Android. Более того, в Realm доступна функция уведомлений об изменении данных.

Но для решения всех этих проблем совместно с SQLite начали использовать ORM – технология программирования, связывающая базы данных с объектно-ориентированными концепциями языков программирования, при этом реализовывать «виртуальную базу данных», которая минимизируют создание огромного количества шаблонного кода, занимающее много времени. Эта технология обеспечивает уровень абстракции над SQLite, что делает работу с системой намного проще и эффективнее для разработчика. Разработчикам удалось не только догнать, но даже опередить Realm в некоторых случаях.

Тем не менее, если разработка не просто хобби, рано или поздно можно понять, что SQLite и ORM намного лучше подходят для больших и сложных проектов. Поэтому выбор в плане базы данных пал на SQLite.

2.1.3 Технология ORM

Итак, в предыдущем пункте речь зашла о технологии ORM, которая значительная облегчила работу с SQLite. Имеется много вариантов для использования:

1. ORMLite. Простой и легкий в использовании, одна из самых быстрых библиотек - встроенное кэширование и отложенная инициализация, хорошая документация. поддерживает вызовы API-интерфейсов баз данных Android SQLite, а его основной веб-сайт содержит отдельные документы только для Android.

2. SugarORM. Из плюсов можно отметить простоту в использовании, которой недостаточно в ORMLite, и все тут. Имеет отдельный недостаток, связанный с подходом к реализации - он не «дружелюбен» к Instant Run, поэтому при работе с ним его нужно отключить.

3. Freezer. Относительно молодая библиотека не получила широкого распространения. Комфортно, но нельзя совершать никаких сложных манипуляций с ним. Кроме того, скорость работы, честно говоря, подводит.

4. DBFlow. Один из лучших по параметрам, изложенным в этой статье. Одной из самых быстрых операций записи / чтения является средняя скорость обновления/удаления. Он является лучшим на уровне Realm.

5. GreenDao. Это гибкое и удобное использование связи один ко многим, но очень неприятная генерация кода, в результате чего ваши классы будут заполнены кучей методов и комментариев. Кроме того, вам необходимо подключить плагин Gradle, что значительно увеличивает время сборки. Также разработчики представили свою базу данных – ObjectBox – объектную базу данных, которую они позиционируют как самую быструю, сравнимую с GreenDao, ORM.

6. Sprinkles. Каждая запись сопровождается ненужной выборкой. От sql-запросов тоже не спасает. Показатели эффективности являются худшими из представленных. Поэтому нет никаких оснований использовать его для своих проектов.

7. Room. Достойное решение, представленное в Google I / O 2017, как оптимальное для работы с базами данных на ОС Android. Несмотря на то, что нужно применять sql-запросы, библиотека как оказалась довольно удобная. По производительности лидирует.

В итоге, можно сказать, что наиболее подходящей библиотекой оказалась ORMLite, благодаря своей быстроте, простоте и легкости интегрирования.

2.1.4 Архитектура приложения

MVC, MVP и MVVM – три популярных шаблона проектирования в разработке ПО.

На самом деле MVP и MVVM есть ничто иное, как производное от MVC. Основное различие между MVC и его производными заключается в зависимости каждого слоя от других уровней, а также от связи друг с другом.

Model View Controller (MVC). Паттерн проектирования MVC делит приложение на три главных аспекта: модель (Model), представление (View) и контроллер (Controller) (см. рис. 2.1).

Model. Модель означает данные, которые требуются для отображения в представлении. Модель представляет из себя набор классов, описывающих бизнес-логику (бизнес-модель и модель данных). Он также определяет бизнес-правила для средств данных, то, как данные могут быть обработаны и изменены.

View. View представляет компоненты интерфейса, такие как XML, HTML и т. д. Представление отображает данные, полученные от контроллера в качестве результата. В шаблоне MVC View контролирует модель на предмет любых изменений состояния и отображает обновленную модель. Model и View коммуницируют друг с другом, применяя шаблон Observer.

Controller. Контролер, отвечающий за обработку входящих запросов, обрабатывает данные пользователя через модель и передает полученный результат обратно в представление. Обычно он предсталяет из себя связь между представлением(View) и моделью(Model).

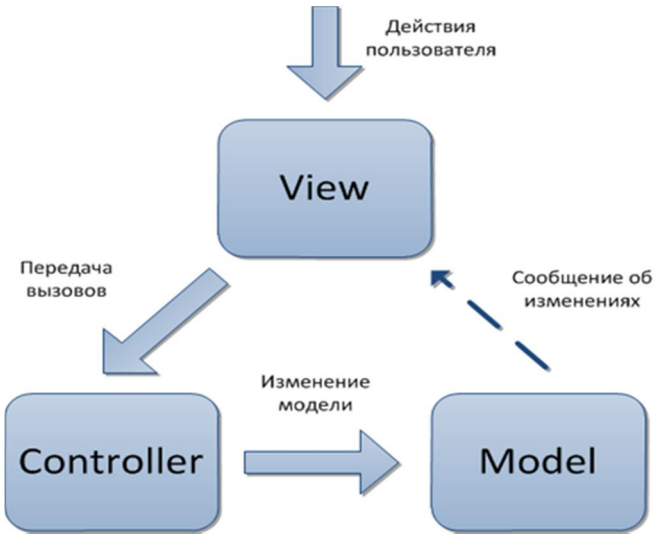

Рисунок 2.1 – Шаблон проектирования MVC

Model View Presenter (MVP). Шаблон MVP аналогичен шаблону MVC. Он получен из шаблона MVC, в котором контроллер заменяется презентатором. Этот шаблон делит приложение на три основных аспекта: модель, представление и презентатор (см. рис. 2.2).

Model. Модель представляет собой набор классов, описывающих бизнес-логику и данные. Также он определяет бизнес-правила для данных, что означает, как данные могут быть изменены.

View. Представление – это компонент, который напрямую взаимодействует с пользователем, например, XML, Activity, фрагменты. Он не содержит никакой реализованной логики.

Presenter. Presenter получает входные данные от пользователей через View, затем обрабатывает данные пользователя с помощью Model и передает полученный результат обратно во View. Ведущий общается с просмотром через интерфейс. Интерфейс определяется в классе презентатора, в который он передает необходимые данные. Activity/Fragment или любой другой компонент представления реализуют этот интерфейс и отображают данные так, как они хотят.

Рисунок 2.1 – Шаблон проектирования MVP

В шаблоне проектирования MVP Presenter управляет моделью, а также обновляет представление. В MVP View и Presenter

полностью отделены друг от друга и взаимодействуют друг с другом через интерфейс. Потому что, если разделение макета представления проще, и модульное тестирование приложений, использующих шаблон проектирования MVP над шаблоном проектирования MVC, будет намного проще.

Model View View-Model (MVVM)

Шаблон MVVM поддерживает двустороннее связывание данных между View и View-Model. Это позволяет автоматически распространять изменения внутри состояния View-Model на View. Как правило, View-Model использует шаблон наблюдателя для информирования об изменениях View-Model в Модели.

View–Model. Он отвечает за предоставление методов, команд и других свойств, которые помогают поддерживать состояние представления, манипулировать моделью в результате действий над представлением и инициировать события в самом представлении. View имеет ссылку на View-Model, но View-Model не имеет информации о View. Между View и View-Model есть связь «многие-к-одному», поэтому многие виды могут быть сопоставлены с одной View-Model. Это полностью не зависит от просмотров.

Двунаправленная привязка данных между представлением и моделью представления обеспечивают синхронизацию моделей и свойств в модели представления с представлением. Шаблон проектирования MVVM хорошо подходит для приложений, которым требуется поддержка двунаправленной привязки данных.

Проанализировав все виды архитектур, наиболее подходящей оказалась MVVM.

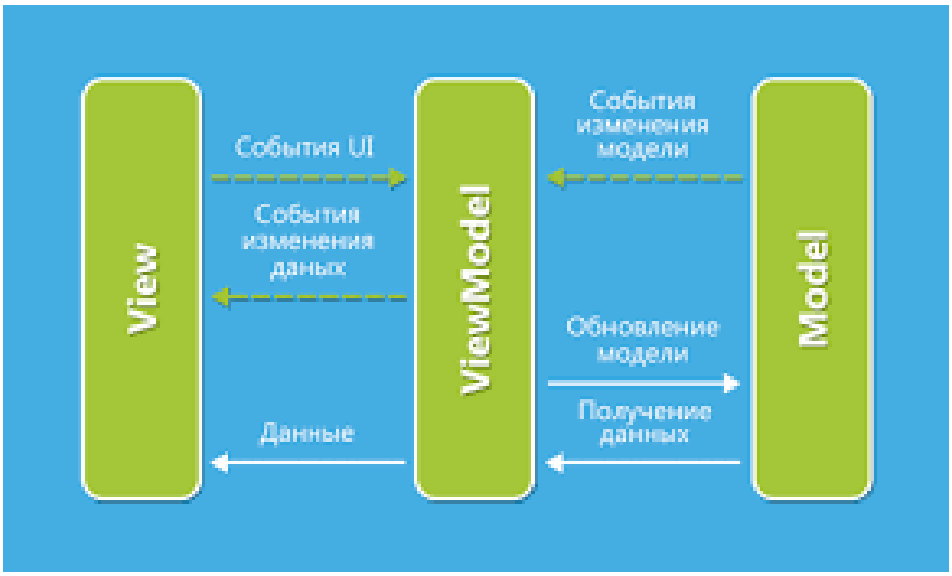

Рисунок 2.3 – Шаблон проектирования MVVM

2.1.5 Библиотека для работы с REST API

Retrofit и Volley – отличные сетевые библиотеки для современных приложений для Android, и у каждого из них есть свои достойные стороны. Retrofit можно использовать, если наше приложение использует стандартный REST API с ответами JSON и не слишком много пользовательских требований с точки зрения кэширования, приоритизации запросов, повторных попыток и т.д.

А Volley можно использовать, если имеются необычные / детальные требования или если понадобится большая гибкость от сетевого уровня в будущем за счет большего количества кода.

Как уже ранее упоминалось, Retrofit используется для парсинга и получения JSON из некоторых веб-служб, или же для отправки данных. Более того, Retrofit значительно упрощает работу с фоновыми потоками и HTTP-запросы, так как является типобезопасным HTTP-клиентом.

В свою очередь Volley – это создание Google, использующееся в приложениях Play Store и разных Google-приложениях. Volley рассматривается как тандем Retrofit + Picasso, и может использоваться для альтернативы нескольким библиотекам. Однако недостатком этой библиотеки является мало документации.

- Целью Retrofit является упрощение использования веб-сервисов RESTful, в то время как целью Volley является решение всех ваших сетевых задач, особенно для Android.
- Retrofit намного проще в настройке
- Volley обеспечивает поддержку обработки сетевых изображений.
- Volley может интегрироваться с большинством популярных HTTP-клиентов, включая OkHttp, где Retrofit (насколько я знаю) опирается на OkHttp.
- Retrofit остается актуальной. Volley может полагаться на библиотеку, поставляемую в комплекте с вашей ОС, поэтому обновление сетевого клиента не вариант.
- Retrofit значительно упрощает настройку перехватов HTTP (если вы хотите что-то сделать до или после вызова HTTP).
- Volley может обеспечить точный контроль над стратегией кэширования, но его сложнее настроить, чем Retrofit.
- Retrofit опирается на OkHttp, который опирается на Okio, что эффективно делает эту библиотеку огромной по сравнению с базовой конфигурацией Volley.
- Retrofit подходит больше, если приложению требуется использование сервисов RESTful, и вы не хотите зацикливаться на конфигурации Volley. Если нужно абстрагировать свой код, используйте Retrofit.

В результате сравнения для приложения мне подходит Retrofit, по выше описанным причинам.

2.2 Техническое задание

1) Цели и задачи приложения

Целью разработки приложения является автоматизация процессов мерчендайзинга.

Первой задачей приложения является предоставление пользователю компании возможности начать визит в торговую точку и заполнить все необходимые данные о товарах в этой торговой точке. Для этого пользователь должен:

- выбрать контрагента из списка и начать визит;
- выбрать товар из каталога (с возможностью отфильтровать товары);
- заполнить данные о товаре (наличие, цена, остаток, фейсинг, комментарий), после чего товар попадает в “Корзину” (список заполненных товаров).

После заполнения информации о продукте, пользователь может:

- просматривать список в “Корзине”;
- редактировать, изменять данные о товаре;
- удалять продукты из “Корзины”;

Кроме того, пользователь должен иметь возможность делать фотоотчет. После заполнения информации о товаре пользователь может:

- открыть камеру КПК и сделать снимок;
- просматривать список фотографий;
- указывать наименования брендов, представленных на фотографии;
- просматривать конечный результат в виде отчета;
- отправка отчета на сервер при подключении к интернету.

2) Портрет целевой аудитории

Основной целевой аудиторией приложения являются существующие и потенциальные клиенты компании, физические и юридические лица. С помощью приложения пользователи могут произвести визит в торговую точку и заполнить анкету в электронном виде.

3) Основные понятия, используемые в техническом задании

Пользователь – человек, использующий данное приложение для выполнения определенных функций.

Мерчендайзинг – комплекс методов, которые направлены на увеличение объема продаж в торговой точке, включающий оформление торговых полок, прилавков, витрин, распределение продукции в торговом зале.

Фейсинг – процесс, задача по выкладке на полке товарных единиц продукции (фейсов), видимых и доступных покупателю в магазинах самообслуживания с полочной выкладкой продукции.

Фотоотчет – фото товара, представленного на полках.

Контрагент – юридическое или физическое лицо, которое берет на себя определенные обязательства по договору (торговая точка).

4) Хранение информации в приложении

Информация необходимая для функционирования приложения загружается при первом использовании приложения, в случае необходимости обновляется. В приложении хранится информация о контрагентах, подробная информация о товарах в оффлайн режиме (при отсутствии подключения к интернету).

5) Язык реализации

Приложение будет реализовано на русском языке. В последующих версиях возможно добавление английского языка.

6) Алгоритм работы с приложением

Пользователь загружает приложение на устройство и получает доступ к следующему функционалу:

- просмотр списка контрагентов;
- выбор одного из контрагентов для начала визита;

После выбора контрагента пользователю становятся доступны следующие функции:

- просмотр каталога товаров;
- возможность отфильтровать товары из каталога;
- выбрать товар и заполнить все необходимые данные о нем (наличие, цена, остаток, фейсинг, комментарий).

После заполнения информации о товаре, он добавляется в новый список “Корзина”, теперь пользователь может:

- просматривать список товаров в “Корзине”;
- редактировать данные о товаре;
- удалять товары из “Корзины”;
- просматривать конечный результат в виде отчета;
- отправка отчета на сервер при подключении к интернету.

7) Механизм работы приложения при подключении к интернету

При отсутствии подключения к интернету пользователю не доступны:

- Загрузка данных из БД;
- Обмен данными;

8) Обратная связь при работе с приложением

При работе с приложением пользователь должен получать обратную связь от своих действий. В приложении это реализуется следующими способами:

- внешний вид иконок изменяется при нажатии на них (иконки отображаются вдавленными или меняют оттенок на более темный);
- поля, заполняемые пользователем, выделяются цветом;
- у вновь добавленных полей или блоков фон является более темным, спустя заданное время (например, 5 сек.) фон становится однородным;
- при отправке информации из приложения на сервер в нижней части экрана отображается прогресс бар.

3 Проектная часть

3.1 Логическая и физическая модели

Логическая модель – это абстрактный взгляд на данные, на нем данные представляются так, как выглядят в реальном мире, и могут называться так, как они называются в реальном мире. Объекты модели, представляемые на логическом уровне, называются сущностями и атрибутами.

Список требований:

– сотрудник компании записывает визит в каждую торговую точку, то есть заполняет один экземпляр документа “Маркетур” для каждой торговой точки отдельно;

– в документе “Маркетур” могут быть включены сразу несколько разных наименований товаров, значит и несколько фотоотчетов;

– у документа “Маркетур” есть уникальный номер, дата и время создания, комментарий и ответное сообщение;

– документу соответствует только один пользователь, контрагент, визит контрагента и статус документа.

Диаграмма ER (логическая модель), полученная из требований, показана на рисунке 3.1:

1. Сущность «Контрагент» идентифицируется своим Id_контрагент, поэтому мы используем его в качестве первичного ключа.
2. Сущность «Визит контрагента» идентифицируется по его Id_визит_контрагента. Начало и конец визита определяются атрибутами “Начало_визита” и “Конец_визита” соответственно. Имеются отдельные атрибуты долготы и широты для обозначения местоположения.
3. Сущность «Пользователь» имеет свой первичный ключ и необходимые атрибуты с информацией о пользователе.
4. Сущность «Статус документа» также идентифицируется своим первичным ключом Id_статус_документа и атрибутами названия, псевдонима статуса документа.
5. Сущность «Мерчендайзинг_контент» идентифицируется по Id_ мерчендайзинг_контент. Дата создания фиксируется своим атрибутом “Дата создания”. Параметры мерчендайзинга определяются атрибутами: “Цена”, “Фейсинг”, “Статус_остатка”, “Остаток”. Комментарий учитывается атрибутом “Комментарий”.
6. Сущность «Фото_контент» идентифицируется по Id_ фото_контент. Дата создания фиксируется своим атрибутом “Дата создания”. Путь и url определяются своими соответствующими атрибутами. Комментарий учитывается атрибутом “Комментарий”.
7. Сущность «Маркетур» идентифицируется по своему Id. Фиксируется дата создания, комментарий и ответное сообщение. В маркетуре могут быть несколько товаров, а значит несколько их фотография, поэтому связь сущности

«Маркетур» с сущностями «Мерчендайзинг_контент», «Фото_контент» имеет кардинальность 1: N. Остальные связи имееют кардинальность 1:1.

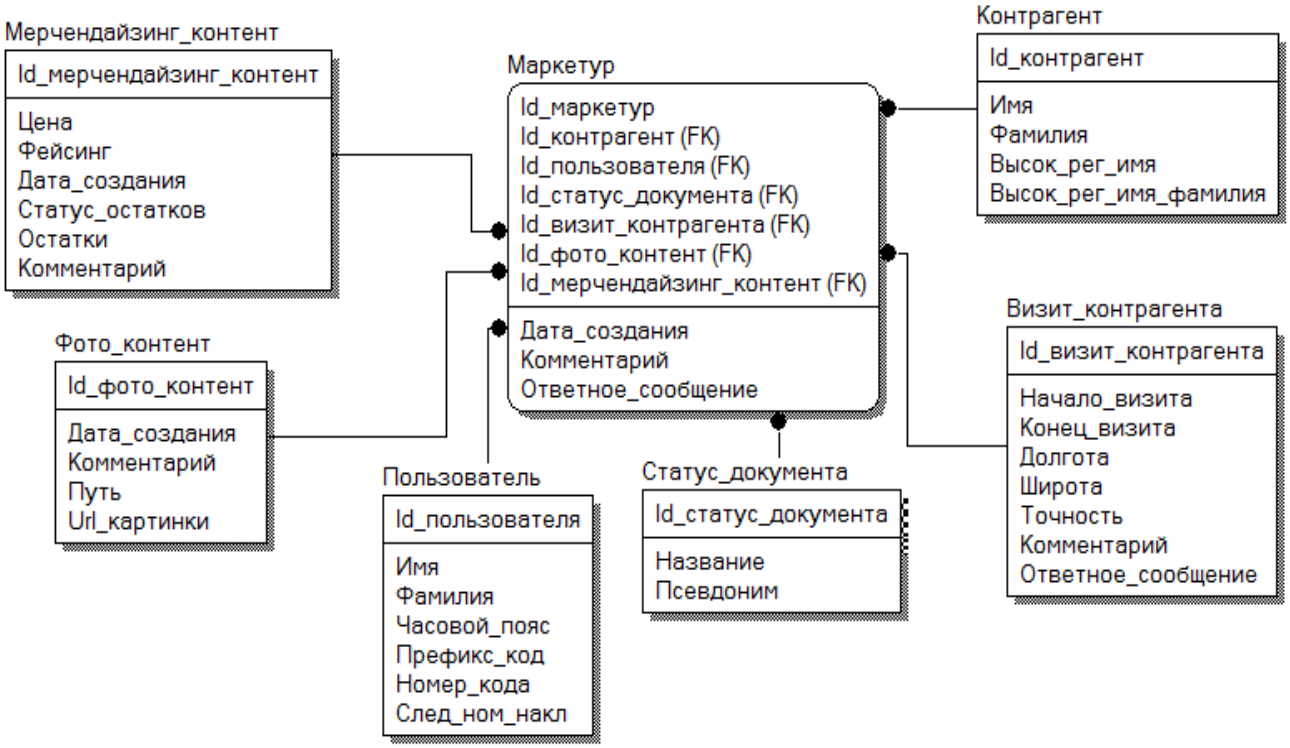

Рисунок 3.1 – Логическая модель базы данных

Физическая модель – логическая модель базы данных, выраженная в терминах языка описания данных конкретной СУБД, т.е. зависит от конкретной СУБД. В физической модели содержится информация о всех объектах БД. Физическая модель – фактически таблицы (см. рис.3.2).

Физическая модель базы данных содержит все детали, необходимые конкретной СУБД для создания базы: наименования таблиц и столбцов, типы данных, определения первичных и внешних ключей и т.п.

/

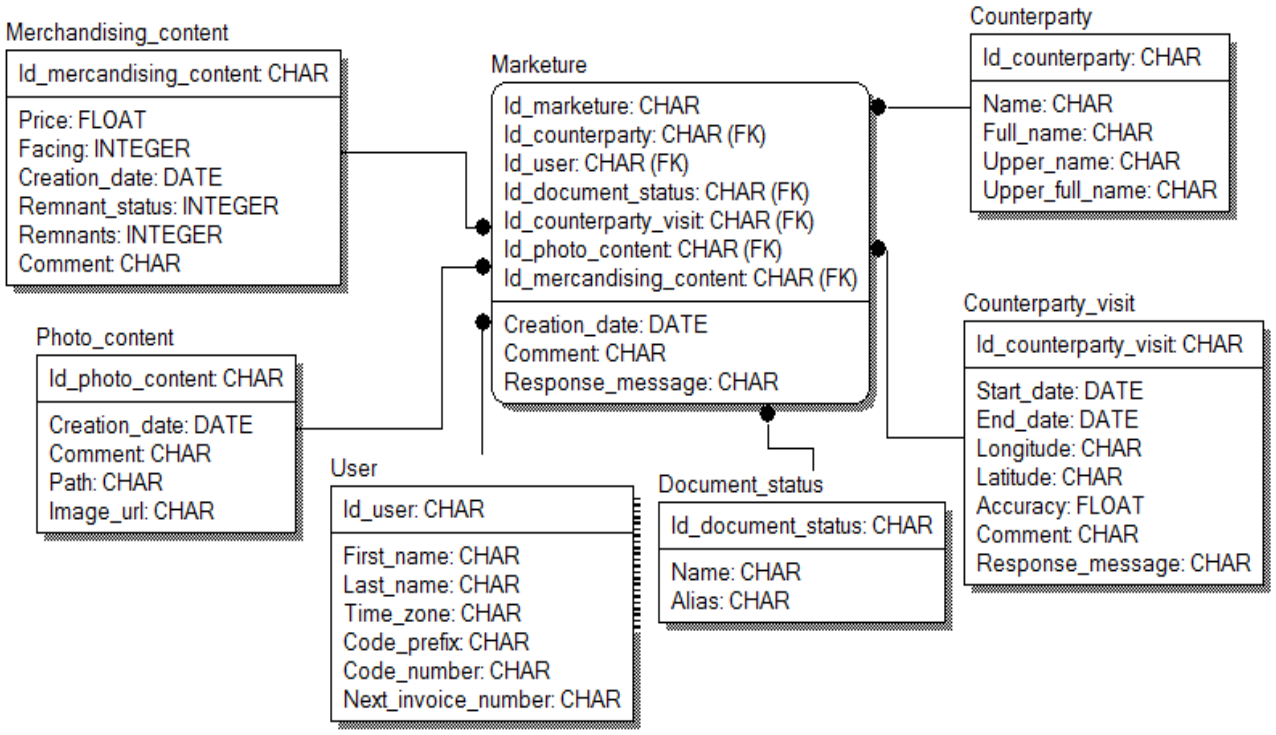

Рисунок 3.2 – Физическая модель базы данных

3.2 Диаграмма вариантов использования (Use-case diagram)

Целью диаграммы вариантов использования в UML является демонстрацией различных способов взаимодействия пользователя с системой. Диаграмма вариантов использования изображает общий обзор взаимосвязей между сценариями использования, субъектами и системами.

Таким образом в нашей системе актером представлен пользователь, который может просматривать список контрагентов, при этом имеет возможность поиска по списку, после чего обязательно выбирает одного контрагента для продолжения работы других функции программы.

Пользователь также просматривает каталог товаров с возможностью фильтрации для упрощения поиска, к тому же может детально просмотреть параметры товара, нажав на один из списка.

После заполнения данных о товаре, он автоматически добавляется в корзину, где пользователь может отредактировать или удалить товар из списка.

Затем необходимо сделать снимки товаров на полке, с возможностью добавления комментария и брендов, просмотра во весь экран и удаления сделанных фотографий.

В конце пользователь может посмотреть результат проделанной работы в виде таблицы (см.рис.3.3).

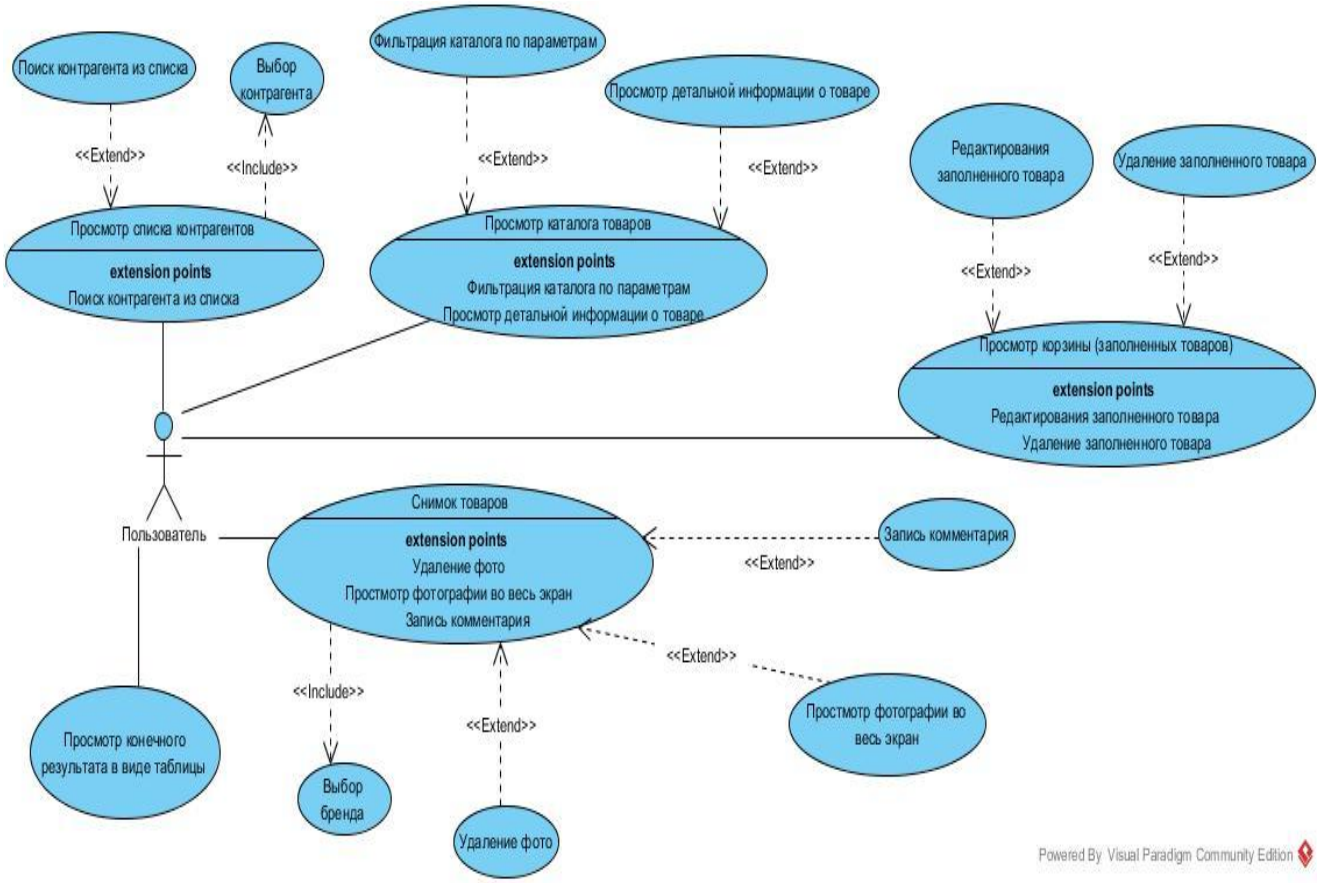

Рисунок 3.3 – Use-case диаграмма

4 Проектирование пользовательского интерфейса

Мобильное приложение состоит из 5 главных страниц. На рисунке 4.1 представлена первая страница приложения, которая открывается после его открытия.

В меню представлены четыре кнопки: кнопка поиска по списку, календарь, дополнительное меню и кнопка назад. Ниже расположены вкладки с другими Activity (окна). Еще ниже recyclerview (список) с контрагентами и кнопкой начать/закончить визит. (см. рис.4.1)

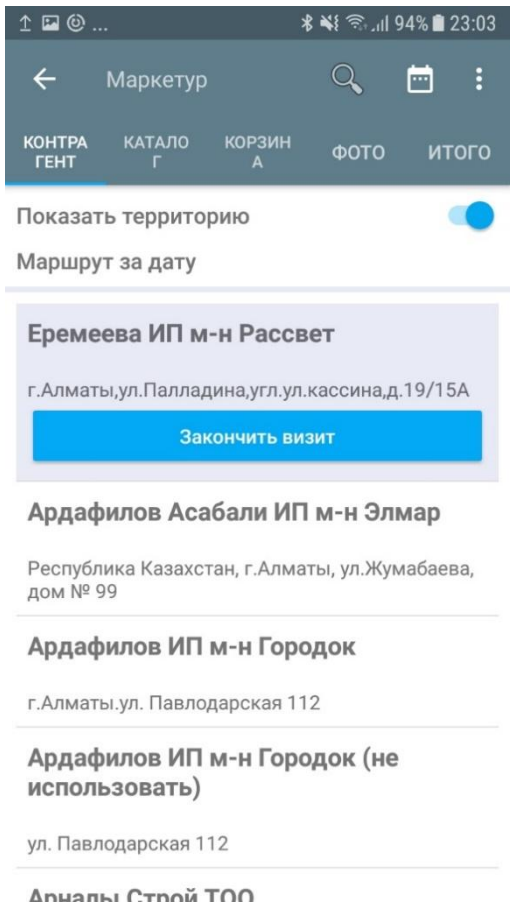

Рисунок 4.1 – Первая страница приложения

Во второй вкладке приложения загружается каталог всех товаров (recyclerview) и внизу располагается кнопка с фильтром, с помощью которой можно отсортировать список и меню с тремя кнопками: кнопка назад, поиск по каталогу и дополнительные параметры меню. (рис. 4.2)

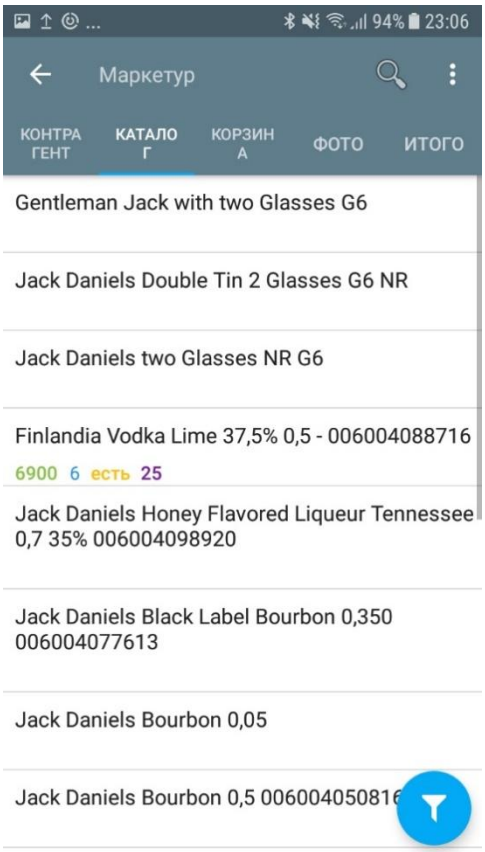

Рисунок 4.2 – Вторая страница приложения

Нажав на кнопку со значком фильтра открывается непосредственно окно фильтра для сортировки товара по брендам (см. рис. 4.3).

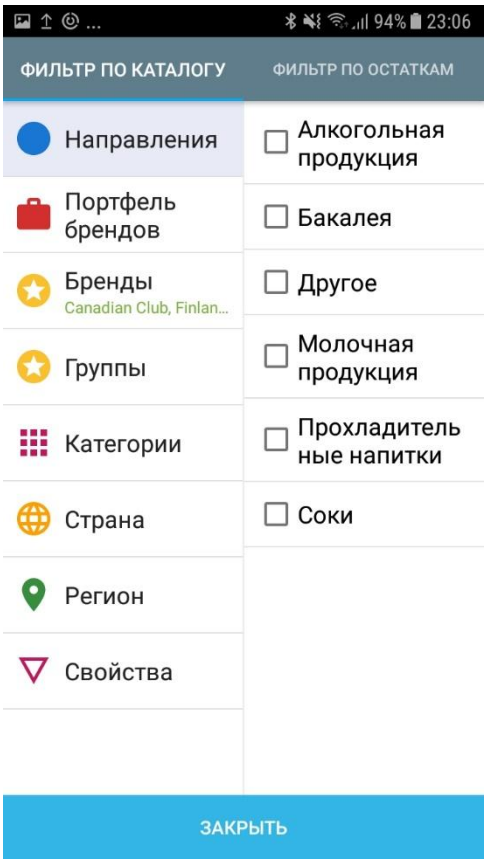

Рисунок 4.3 – Третья страница приложения

При нажатии на один товар из списка открывается страница для заполнения данных о товаре: наличие, остатки на складе, цена, фейсинг и комментарий. Сверху имеется кнопка в виде стрелки для перехода в предыдущее окно. В самом низу располагается кнопка для сохранения данных (см. рис. 4.4).

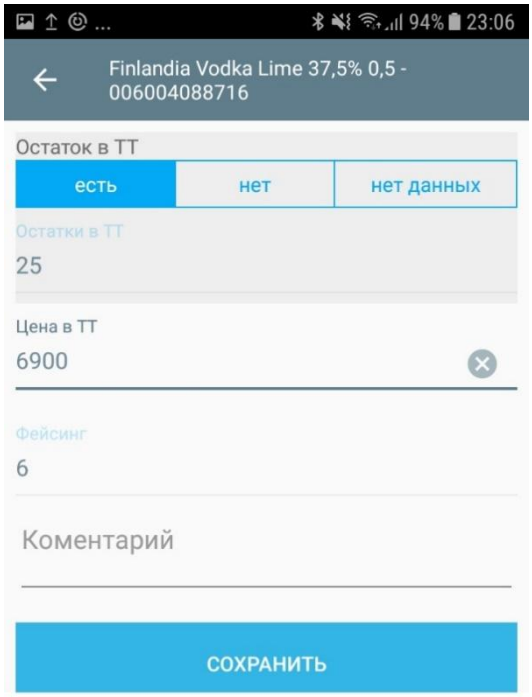

Рисунок 4.4 – Заполнения данных о товаре

Следующее вкладка содержит окно со списком из заполненных товаров и кнопки для снятия фотографий, сверху в меню имеются кнопка с возвращением в предыдущую активность и дополнительно меню (рис. 4.5).

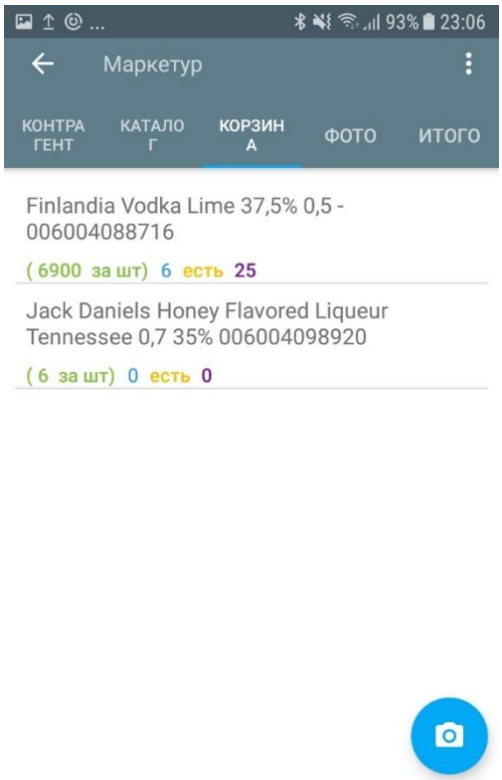

Рисунок 4.5 – Список с заполненными товарами и камера

Это окно открывается при нажатии кнопки для того, чтобы сделать фотографию (см. рис.4.6).

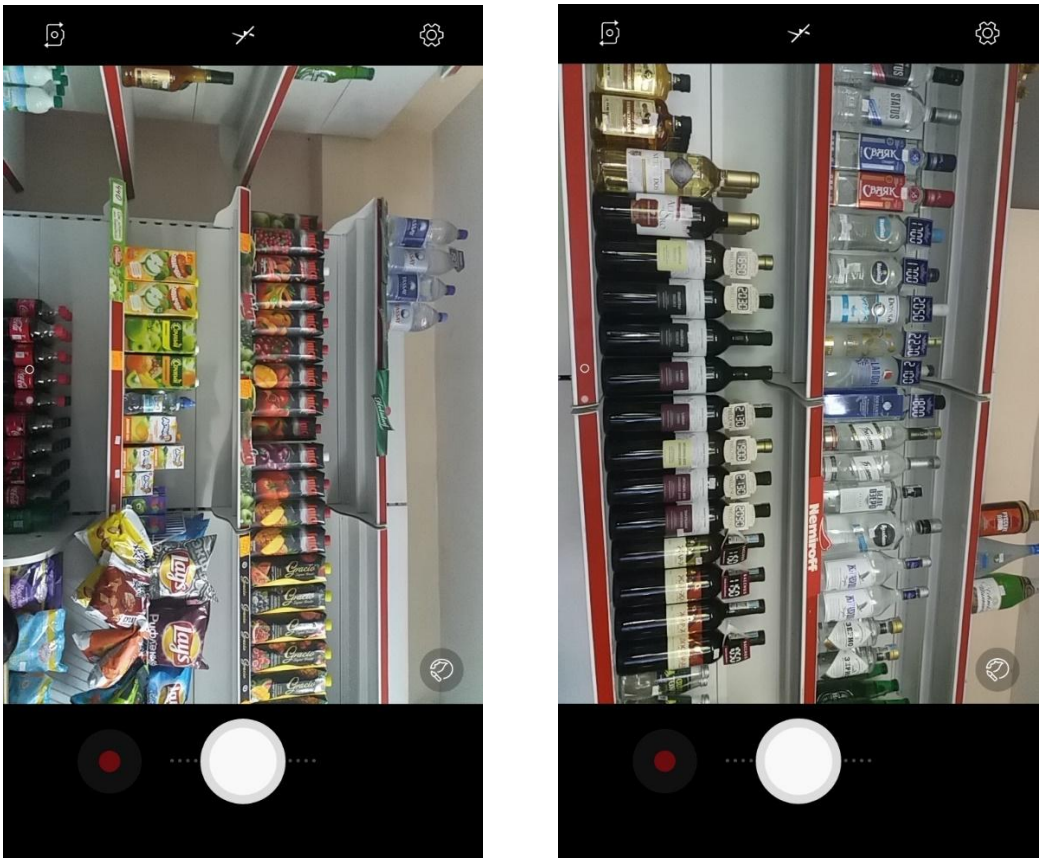

Рисунок 4.5 – Окно для снятия снимка

Предпоследняя страница состоит из списка сделанных фотографий и кнопок фильтра и камеры. С помощью фильтра можно выбрать список брендов, представленных на фотографии. Сверху кнопка с возвращением в предыдущую активность и дополнительно меню (см. рис .4.6).

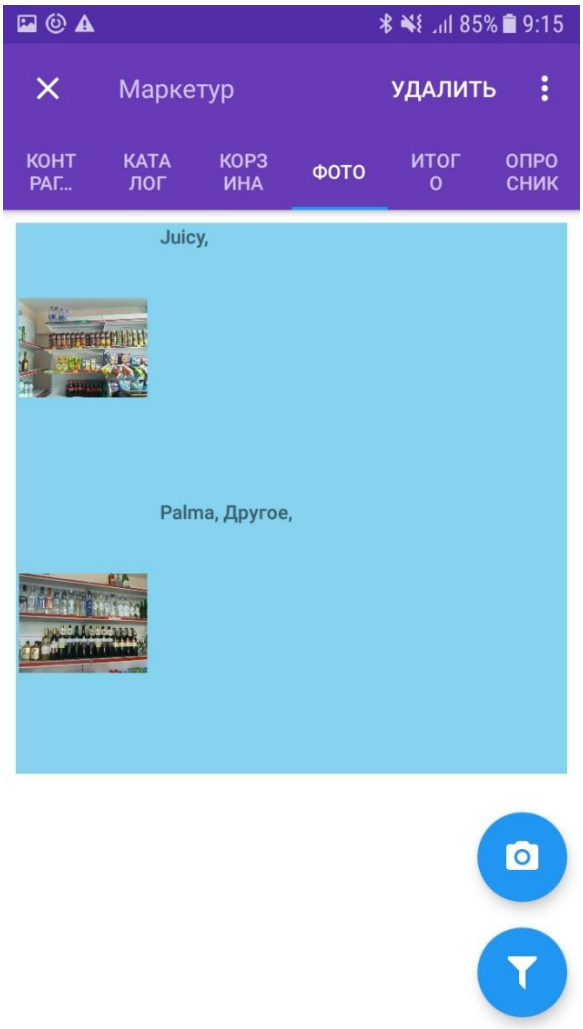

Рисунок 4.6 – Список сделанных фотографий

Последнее окно с ранее заполненными данными в виде документа. Сверху кнопка для закрытия маркетура и дополнительно меню (рис. 4.7).

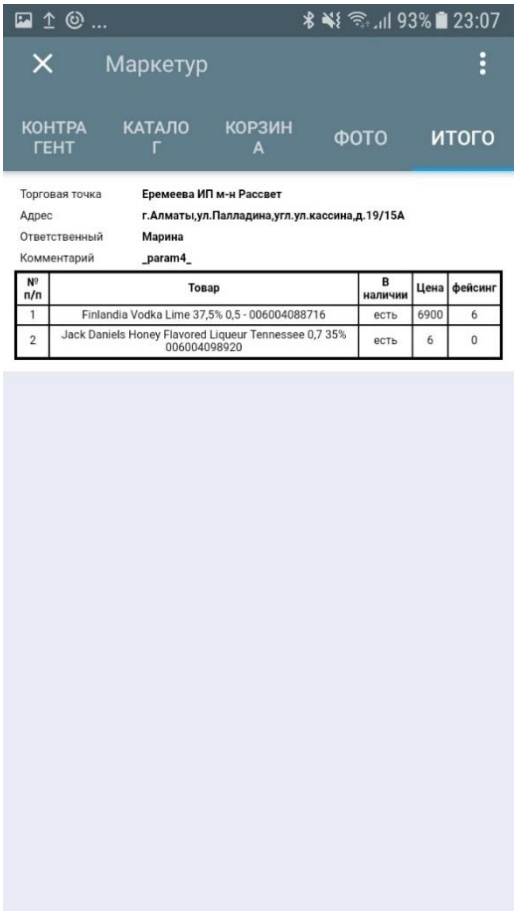

Рисунок 4.7 – Последнее окно

ЗАКЛЮЧЕНИЕ

Результатом дипломного проекта является автоматизированная информационная система процессов мерчендайзинга на платформе Android. Программный продукт ориентирован сугубо для внутрикорпоративного применения и не является свободно распространяемым ресурсом. Основным требованием к системе явилось создание системы электронного учёта мерчендайзинга предприятия, которое является основной функцией и целью создания системы.

В процессе проектирования уточнялись требования к системе и продукт дополнялся новыми функциями. Кроме системы мерчендайзинга была предусмотрена автоматизация фотоотчета, загрузка каталога товаров и списка контрагентов.

В процессе исследования предметной области был проведён анализ сравнительных характеристик информационных систем и сделан следующий вывод: рынок показывает наличие большого количества конкурентных торговых компаний. Все системы представляют собой комплекс защитных мер и не поставляются отдельно, в основном это физические заграждения виде турникетов и шлагбаумов, применяемых к организации безопасности целого здания, а не отдельных офисов или объектов. Все импортируемые системы носят закрытый характер и не могут дорабатываться в соответствии с требованиями заказчика. Для установки новой системы требуются специалисты извне или требуется специальное обучение.

Автоматизированная система мерчендайзинга является открытой для разработчиков системой и представляет законченный программный продукт, поэтому легко дорабатывается в соответствие с новыми требованиями, что, в свою очередь, может явиться причиной быстрого развития системы. Кроме того, программный продукт не несёт специализированного оборудования и не предъявляет особенных требований к аппаратной или программной составляющей. Приложение является кроссплатформенным, что делает его более мобильным и не ориентированным на аппаратный комплекс заказчика.

Данное приложение является ядром для аппаратной доработки и программного усовершенствования. В дальнейшем планируется внедрение автоматического распознования товаров. Это увеличит уровень цифровизации данного процесса, сводя вмешательство человеческого фактора со стороны пользователя к нулю.

Сотрудникам больше не придётся сверять сотни фотографий товаров на полках в день, а просто наблюдать за результатами работ системы в рамках специальнно разработанных отчетов.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Введение в MVVM // Электронная версия на сайте https://proandroiddev.com/mvvm-architecture-viewmodel-and-livedata-part-1-604f50cda1 .
2. Логинова, Е. Мерчендайзинг [Текст]/ А.С. Якорева, М.С. Клочкова.– Москва: Март, 2006. – 300 с.
3. OrmLite - Lightweight Object Relational Mapping (ORM) Java Package // Электронная версия на сайте http://ormlite.com/ .
4. ORMLite –Простая работа с базой данных // Электронная версия на сайте http://akutepov.ru/ru/blog/ormlite-prostaia-rabota-s-bazoi-dannykh/ .
5. Android ORMLite по примеру SQLite // Электронная версия на сайте https://riptutorial.com/ru/android/example/24951/android-ormlite-over-sqlite-example .
6. Model–View–View-Model // Электронная версия на сайте https://www.wintellect.com/model-view-viewmodel-mvvm-explained/ .
7. Руководство по SQLite // Электронная версия на сайте https://proglib.io/p/sqlite-tutorial/ .
8. LiveData // Электронная версия на сайте https://startandroid.ru/ru/courses/architecture-components/27-course/architecture-components/525-urok-2-livedata.html .
9. LiveData overview // Электронная версия на сайте https://developer.android.com/topic/libraries/architecture/livedata .
10. Android Spinner // Электронная версия на сайте http://developer.alexanderklimov.ru/android/views/spinner.php .
11. Работа с камерой на Android // Электронная версия на сайте https://startandroid.ru/ru/uroki/vse-uroki-spiskom/266-urok-133-kamera-delaem-snimok-i-pishem-video.html .

Приложение А

Текст программы

```
//код модели Marketure:
package velait.raimbek_mobile_trade.Model;


import com.google.gson.annotations.SerializedName;
import com.j256.ormlite.dao.ForeignCollection;
import com.j256.ormlite.field.DatabaseField;
import com.j256.ormlite.field.ForeignCollectionField;
import com.j256.ormlite.table.DatabaseTable;

import java.sql.SQLException;
import java.util.ArrayList;
import java.util.Date;
import java.util.UUID;

import velait.raimbek_mobile_trade.App;
import velait.raimbek_mobile_trade.Utils.DateTimeUtil;

@DatabaseTable(tableName = Marketure.TABLE_MARKETURE)
public class Marketure extends BaseModel {
   public static final String TABLE_MARKETURE = "market_tour";
   public static final String ID_MARKETURE    = "id_market_tour";
   public static final String CREATION_DATE   = "creation_date";
   public static final String COMMENT         = "comment";
   public static final String RESPONSE_MESSAGE    = "response_message";
   public static final int  AVAILABLE     = 1;  // 1-есть
   public static final int  NOT_AVAILABLE = 0;  // 0-нет
   public static final int  NOT_DATA      = -1; // -1-нет данных

   @SerializedName(ID_MARKETURE)
   @DatabaseField(columnName = ID_MARKETURE, id = true)
   public String id;

   @SerializedName(DocumentStatus.TABLE_DOCUMENT_STATUS)
   @DatabaseField(columnName = DocumentStatus.ID_DOCUMENT_STATUS,
foreign = true, foreignAutoRefresh = true, index= true)
   public DocumentStatus  documentStatus;
```

Продолжение приложения А

```
 @SerializedName(COMMENT)
    @DatabaseField(columnName = COMMENT)
    public String comment;

    @SerializedName(Counterparty.TABLE_COUNTERPARTY)
    @DatabaseField(columnName = Counterparty.ID_COUNTERPARTY, foreign =
true, foreignAutoRefresh = true, index = true)
    public Counterparty counterParty;

    @SerializedName(CounterpartyVisit.TABLE_COUNTER_PARTY_VISIT)
    @DatabaseField(columnName =
CounterpartyVisit.ID_COUNTER_PARTY_VISIT, foreign = true,
foreignAutoRefresh = true, index = true)
    public CounterpartyVisit counterpartyVisit;

    @SerializedName(User.TABLE_USER)
    @DatabaseField(columnName = User.ID_USER, foreign = true,
foreignAutoRefresh = true, index = true)
    public User user;

    @SerializedName(RESPONSE_MESSAGE)
    @DatabaseField(columnName = RESPONSE_MESSAGE)
    public String responseMessage;

    @SerializedName(PhotoContent.TABLE_PHOTO_CONTENT)
    @ForeignCollectionField(columnName = PhotoContent.ID_PHOTO_CONTENT,
eager = true)
    public ForeignCollection<PhotoContent> photoContentList;


@SerializedName(MerchandisingContent.TABLE_MERCHANDISING_CONENT)
    @ForeignCollectionField(columnName =
MerchandisingContent.ID_MERCHANDISING_CONTENT, eager = true)
    public ForeignCollection<MerchandisingContent> merchandisingContentList;

    // todo костыль для форминрования json
    public void clearForJSON() {
        for (PhotoContent photoContent : photoContentList)
            photoContent.clearForJSON();

        for (MerchandisingContent merchandisingContent : merchandisingContentList)
            merchandisingContent.clearForJSON();
```

```
        counterParty.clearForJSON();

        if (counterpartyVisit != null)
            counterpartyVisit.counterparty.clearForJSON();
    }

    public boolean canUpload() {
//        if (  someFieldNull()  ) return false;
//        if(counterparty.getLastVisitEndTime() == null)  return false;// Если визит не
завершен то запрещаем отправку заказа

        if (  documentStatus  == null )  return true;

        if (getDocumentStatusName().equals("uploaded")) return false;
        if (getDocumentStatusName().equals("sent"))     return false;
        if (getDocumentStatusName().equals("draft"))    return false;

        if (getDocumentStatusName().equals("approved")) return true;
        if (getDocumentStatusName().equals("error"))    return true;

        return false;
    }

    public String getCreationDate() {
        return creationDate == null ? "" :  DateTimeUtil.formatDateTime0(
creationDate.getTime() );
    }

    public Marketure() {
    }


    public Marketure(Counterparty counterparty) {
        id              =  UUID.randomUUID().toString();
        creationDate      =  new Date();
        user            = App.sessionInfo.user;
        this.counterParty  = counterparty;
        counterpartyVisit             = counterParty.getLastVisit();
        counterpartyVisit.counterparty   = counterparty;
        counterpartyVisit.user          = App.sessionInfo.user;
    }
```

```
/*
   public void setCounterparty(Counterparty counterparty) {
      this.counterParty  = counterparty;
      counterpartyVisit               = counterParty.getLastVisit();
      counterpartyVisit.counterparty   = counterparty;
      counterpartyVisit.user          = App.sessionInfo.user;
   }
*/

   public String getCounterpartyName(){
      if(counterParty == null) return "Не выбран контрагент";
      return  counterParty.name;
   }

   public void setComment(String comment) {
      this.comment = comment;
   }

   public String getComment() {
      return comment;
   }

   public boolean canApprove() {

      if (getDocumentStatusName().equals("draft"))    return true;

      if (getDocumentStatusName().equals("error"))    return false;
      if (getDocumentStatusName().equals("uploaded")) return false;
      if (getDocumentStatusName().equals("sent"))     return false;
      if (getDocumentStatusName().equals("approved")) return false;

      return false;
   }

   public boolean deletePhotoContent(PhotoContent photoContent) {
      try {
         if (photoContentList.contains(photoContent))
            photoContentList.remove(photoContent);

         photoContentList.refreshCollection();
      } catch (SQLException e) {
         e.printStackTrace();
```

```
 return false;
        }

        return false;
    }

    public ArrayList<PhotoContent> getPhotoContentArrayList() {
        ArrayList<PhotoContent> photoContentArrayList = new ArrayList<>();

        for (PhotoContent photoContent : photoContentList)
            photoContentArrayList.add(photoContent);

        return photoContentArrayList;
    }

    public boolean deleteMerchandisingContent(MerchandisingContent
merchandisingContent) {
        if (merchandisingContentList.contains(merchandisingContent))
            return merchandisingContentList.remove(merchandisingContent);

        return false;
    }

    public boolean saveMerchandisingContent(MerchandisingContent
merchandisingContent) {
        try {
            if (merchandisingContentList.contains(merchandisingContent)) {
                boolean res =  merchandisingContentList.update(merchandisingContent) >
0;
                merchandisingContentList.refreshCollection();
                return res;
            }
            else
                return merchandisingContentList.add(merchandisingContent);

        } catch (SQLException e) {
            e.printStackTrace();
            return false;
        }
    }

    public ArrayList<MerchandisingContent> getMerchandisingContentArrayList() {
```

```
ArrayList<MerchandisingContent> orderContentArrayList = new ArrayList<>();

      for (MerchandisingContent merchandisingContent : merchandisingContentList)
         orderContentArrayList.add(merchandisingContent);

      return orderContentArrayList;
   }

   public MerchandisingContent getMerchandisingContentBySku(Sku sku) {
      for (MerchandisingContent merchandisingContent : merchandisingContentList)
{
         if (merchandisingContent.sku.equals(sku)) {
            merchandisingContent.sku = sku;
            return merchandisingContent;
         }
      }
      return null;
   }

   public MerchandisingContent getOrCreateMerchandisingContentBySku(Sku sku)
{
      for (MerchandisingContent merchandisingContent : merchandisingContentList)
{
         if (merchandisingContent.sku.equals(sku)) {
            merchandisingContent.sku = sku;
            return merchandisingContent;
         }
      }
      return new MerchandisingContent(this, sku);
   }

   public boolean canEdit() {
      if (getDocumentStatusName().equals("approved")) return true;
      if (getDocumentStatusName().equals("error"))    return true;
      if (getDocumentStatusName().equals("draft"))    return true;

      if (getDocumentStatusName().equals("uploaded")) return false;
      if (getDocumentStatusName().equals("sent"))     return false;

      return false;
   }
```

```
    public boolean canDelete() {
//        if ( someFieldNull() ) return true;

        if (getDocumentStatusName().equals("draft"))    return true;
        if (getDocumentStatusName().equals("approved")) return true;
        if (getDocumentStatusName().equals("error"))    return true;

        if (getDocumentStatusName().equals("uploaded")) return false;
        if (getDocumentStatusName().equals("sent"))     return false;

        return false;
    }


    public String getDocumentStatusName(){
        if (documentStatus == null) return  "draft";
        return  documentStatus.name;
    }

    public String getDocumentStatusAlias(){
        if (documentStatus == null) return  "Черновик";
        return  documentStatus.alias;
    }

    @Override
    public String toString() {
        return id;
    }

    @Override
    public boolean equals(Object o) {
        boolean retVal = false;
        Marketure marketureSku = (Marketure) o;

        if (marketureSku.id.equals(id)) {
            retVal = true;
        }

        return retVal;
    }

}
```

Приложение Б

Текст программы

//код модели MarketurePhotoFragment:

```
package velait.raimbek_mobile_trade.Marketure;

import android.arch.lifecycle.ViewModelProviders;
import android.os.Bundle;
import android.support.annotation.Nullable;
import android.support.v7.widget.LinearLayoutManager;
import android.support.v7.widget.RecyclerView;
import android.view.LayoutInflater;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.View;
import android.view.ViewGroup;

import java.util.ArrayList;

import velait.raimbek_mobile_trade.BaseFragment;
import velait.raimbek_mobile_trade.Catalog.CatalogViewModel;
import velait.raimbek_mobile_trade.Model.Brand;
import velait.raimbek_mobile_trade.Model.Category;
import velait.raimbek_mobile_trade.Model.Marketure;
import velait.raimbek_mobile_trade.Photo.PhotoViewModel;
import velait.raimbek_mobile_trade.R;

public class MarketurePhotoFragment extends BaseFragment {

    private View view;
    private RecyclerView recyclerView;

    private MarketurePhotoAdapter adapter;
    Marketure marketure;
    MarketureViewModel marketureViewModel;
    CatalogViewModel catalogViewModel;
    PhotoViewModel photoViewModel;

    ArrayList<Brand> brandsSelected = new ArrayList<>();
```

Продолжение приложения Б

```
        ArrayList<Category> categoryListSelected = new ArrayList<>();


    public MarketurePhotoFragment() {
    }

    public static MarketurePhotoFragment newInstance() {
        MarketurePhotoFragment fragment = new MarketurePhotoFragment();
        Bundle args = new Bundle();

        fragment.setArguments(args);
        return fragment;
    }

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        marketureViewModel =
ViewModelProviders.of(getActivity()).get(MarketureViewModel.class);
        photoViewModel =
ViewModelProviders.of(getActivity()).get(PhotoViewModel.class);
        catalogViewModel =
ViewModelProviders.of(getActivity()).get(CatalogViewModel.class);

        catalogViewModel.brandListSelectedLiveData.observe(this, brands -> {
            this.brandsSelected = brands;
        });

        marketureViewModel.marketureSelectedData.observe(this, item -> {
            marketure = item;
            if (marketure != null && marketure.photoContentList.size() > 0)
                recyclerView.setVisibility(View.VISIBLE);
            else
                recyclerView.setVisibility(View.GONE);
        });

        catalogViewModel =
ViewModelProviders.of(getActivity()).get(CatalogViewModel.class);
        catalogViewModel.filterClosed.observe(this, item -> {
            saveBrands();
        });
```

Продолжение приложения Б

```
        marketureViewModel.photoContentSelectedData.observe(this, photoContent ->
{
            if (marketureViewModel.inEditMode())
                setHasOptionsMenu(true);
            else {
                setHasOptionsMenu(false);
            }
        });
        setHasOptionsMenu(false);
    }

    @Override
    public void onCreateOptionsMenu(Menu menu, MenuInflater inflater) {
        inflater.inflate(R.menu.marketure_recyclermenu, menu);
        super.onCreateOptionsMenu(menu, inflater);
    }

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

        recyclerView = (RecyclerView) view.findViewById(R.id.recyclerView);
        recyclerView.setHasFixedSize(true);
        recyclerView.setLayoutManager(new LinearLayoutManager(getActivity()));
        adapter = new MarketurePhotoAdapter(getActivity(), this);
        recyclerView.setAdapter(adapter);
        return view;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            case R.id.action_comment:
                leaveComment();
                break;
            case R.id.action_delete:
                deleteSelectedPhotoContent();
                if (marketureViewModel.inEditMode())
                    setHasOptionsMenu(false);
                break;
```

Продолжение приложения Б

```
case R.id.comment:
                leaveComment();
                break;
            default:
                return super.onOptionsItemSelected(item);
        }
        return false;
    }

    //Delete selected rows
    public void deleteSelectedPhotoContent() {
        markетureViewModel.deletePhotoContent();
    }

    public void leaveComment() {
        MarketureCommentEditFragment markетureCommentEditFragment =
MarketureCommentEditFragment.newInstance();
        markетureCommentEditFragment.show(getFragmentManager(),
"MarketureCommentEditFragment");
        markетureCommentEditFragment.setTargetFragment(this, 1);
    }

    public void saveBrands() {
        markетureViewModel.savePhotoContentBrands(brandsSelected,
categoryListSelected);
    }

}
```