**Содержание**

[Введение 4](#_Toc516742070)

[Раздел 1 Предпроектное обследование 6](#_Toc516742071)

[1.1 Исследование предметной области 6](#_Toc516742072)

[1.2 Формирование требований к программному изделию 7](#_Toc516742073)

[Раздел 2 Техническое задание 10](#_Toc516742074)

[2.1 Постановка задачи 10](#_Toc516742075)

[2.2 Концепция информационной базы и её структура 11](#_Toc516742076)

[2.3 Неформальное описание алгоритма работы программного продукта 14](#_Toc516742077)

[2.4 Требования к контролю, обновлению и восстановлению данных, защите данных от разрушений 15](#_Toc516742078)

[2.5 Функции подсистем (модулей) 16](#_Toc516742079)

[2.6 Оценка затрат на разработку программного продукта 17](#_Toc516742080)

[Раздел 3 Эскизный проект 22](#_Toc516742081)

[Раздел 4 Разработка проектных решений 35](#_Toc516742082)

[Заключение 37](#_Toc516742083)

[Список использованных источников 38](#_Toc516742084)

[Приложение А Блок-схема алгоритма программы 39](#_Toc516742085)

[Приложение Б Документация пользователя 41](#_Toc516742086)

[Приложение В Документация программиста 44](#_Toc516742087)

[Приложение Г Исходный текст программного изделия 47](#_Toc516742088)

## Введение

Тема дипломного проекта: «Разработка приложения по учёту расходов на основе фискальных чеков».

Целью создания программного продукта является упрощение хранения, анализа и просмотра информации о фискальных чеках, которые выдают в различных розничных магазинах. Программное изделие будет производить сканирование QR-кода на фискальном чеке, производить проверку реквизитов, скачивать электронную копию чека, производить необходимые преобразования и реструктуризацию данных некоторых полей и заносить полученные данные в базу данных. После этого приложение будет выводить данные о фискальных чеках, разбитые на три категории: покупки, магазины и товары. Также для магазинов и товаров будет выводиться статистика о приобретениях: общая сумма покупок, максимальная и минимальная стоимость товаров.

На данный момент актуальность разработки особенно высока, так как всё больше людей и компаний переходит на электронный документооборот, правительство выпустило федеральный закон 54, который обязует все торговые предприятия загружать фискальные чеки на сервера Федеральной налоговой службы, а для взаимодействия с этими чеками есть только официальное приложение ФНС, которое имеет достаточно ограниченный функционал.

Разрабатываемое приложение будет использоваться любыми физическими и юридическими лицами, которым требуется обеспечить удобное хранение данных о своих приобретениях и расходах на них.

Программное изделие должно работать на любом устройстве на мобильной платформе Android версии не ниже 5.1, камерой и доступом к сети интернет.

Для успешной разработки данного программного продукта выдвигаются следующие задачи:

* исследовать предметную область;
* сформировать подробные требования к программному продукту;
* составить техническое задания для разработки;
* составить макет интерфейса приложения;
* разработать программный продукт;
* предоставить средства дистрибуции установочного пакета.

## Раздел 1 Предпроектное обследование

# 1.1 Исследование предметной области

Технологии к каждым днём проникают в нашу повседневную жизнь всё сильнее. В любой отрасли, от крупных производств до домашнего быта. И одна из отраслей, которая подвержена интенсивному технологическому росту – это сфера торговли. Компании постоянно внедряют инновационные системы, чтобы максимально оптимизировать все процессы и уменьшить расходы. С другой стороны, правительство тоже должно следить за развитием отрасли и предлагать новые решения. Одним из таких решений стал переход на электронную систему фискальных накопителей.

С 1 февраля 2017 года вступила в силу новая редакция федерального закона №54, согласно которому контрольно-кассовая техника должна отправлять электронные версии чеков оператору фискальных данных. Для некоторых ИП, работающих по упрощенной системе налогообложения, необходимость отправки электронной версии чеков наступит с 1 июля 2019 года. [1]

Но уже на сегодняшний день большое количество организаций, специализирующихся на продаже, выдают покупателем чеки, содержащие QR-штрихкод, содержащий реквизиты чека. К 1 июля 2019 все организации, реализующие какой-либо товар, будут выдавать такие кассовые чеки. Отсканировав QR-код, можно получить ссылку на сайт оператора фискальных чеков, где можно просмотреть кассовый чек или сохранить его. Предлагается на основе реквизитов, полученных сканированием фискального чека, скачивать электронную версию чека и сохранять в базу данных для дальнейшей обработки.

Все организации, занимающиеся продажей каких-либо товаров, выдают покупателям кассовые чеки. Они выступают как подтверждение покупки и обычно содержат информацию о приобретённых позициях. Часто кассовые чеки используют при точном подсчёте расходов – на какие категории товаров были потрачены деньги. Также чек выступает как одно из возможных доказательств при возврате или обмене приобретённого товара.

Проблема заключается в том, что зачастую хранить чеки неудобно. При эксплуатации любого физического носителя существуют определенные неудобства. Бумага, например, легко подвергается внешним воздействиям. Чек, напечатанный на термобумаге, становится нечитаемым в достаточно короткий срок, что делает невозможным его дальнейшее использование.

Покупатели уже сейчас могут на этапе расчёта за покупки сказать продавцу, чтобы тот вместо выдачи бумажного фискального чека отправил электронную версию на электронную почту или на мобильный телефон в СМС. Но большинство продавцов просто даже не знают о такой возможности, что затрудняет использование такого варианта. Ко всему прочему, получение чека таким способом часто является медленным, так как чек может прийти не сразу, а в течение нескольких дней.

На данный момент для мобильных устройств существует официальное приложение Федеральной налоговой службы, которое позволяет сканировать QR-код и получать электронную копию чека. Но функционал приложения ограничен только просмотром изображения чека, которое зачастую неудобно просматривать, и экспортом в форматы JSON, PNG и HTML, которые плохо подходят для анализа.

# 1.2 Формирование требований к программному изделию

Программное изделие должно представлять из себя приложение для мобильной платформы, позволяющее сохранять и просматривать информацию о совершенных покупках в оптовых и розничных магазинах, а также в различных торговых точках. Приложению требуется отображать подробную информацию о приобретениях, вести статистику денежных расходов, а также динамику цен на отдельно взятый товар. Также необходим функционал группировки торговых точек одной организации с информацией о них, включая возможность просмотра местоположения торговой точки на карте.

Получение информации о фискальных чеках должно быть реализовано путём сканирования QR-штрихкода, расположенного на бумажном чеке с помощью камеры устройства или импортом JSON-файла с данными о фискальном чеке из официального приложения Федеральной налоговой службы Российской Федерации.

Программное изделие также должно иметь понятный и удобный пользовательский интерфейс на русском языке и должно адаптироваться под различные форм-факторы и разрешения устройств.

**Требования к функциям и задачам**

Приложение должно реализовывать следующий функционал:

* получение информации о фискальном чеке с сервера оператора фискальных данных;
* добавление полученной информации о чеке в базу данных приложения;
* группировка добавленных чеков по организациям и торговым точкам;
* вывод всех приобретённых товаров;
* вывод всех организаций и торговых точек, в которых были совершены покупки;
* вывод информации и статистики по каждой организации и торговой точке, а именно: общее количество покупок, общая сумма покупок, адрес торговой точки (при наличии), название организации;
* вывод информации о каждом фискальном чеке: количество и наименование позиций, количество товара в позиции, цена товара за штуку, стоимость позиции, сумма к оплате, сумма НДС, сумма скидок (при наличии), информацию об организации, ФИО кассира (при наличии)
* вывод информации о каждом приобретённом товаре: наименование, общее купленное количество, общая стоимость, минимальная цена, максимальная цена, торговая точка.

**Требования по информационному обеспечению**

В программном изделии будет два класса хранимых данных:

1. данные о фискальных чеках, товарах, организациях и торговых точках;
2. параметры приложения;

Для первого класса данных целесообразно использовать базу данных, это даст удобное хранение всех необходимых данных в отдельных таблицах со связями между ними. Для организации базы данных будет использоваться SQLite - компактная встраиваемая СУБД. Её выбор обусловлен тем, что SQLite не использует парадигму «клиент-сервер», а работает как встраиваемая в приложение библиотека, что означает отсутствие необходимости в разработке отдельного серверного приложения. Также, эта СУБД является стандартной для платформы Android. Это позволяет уменьшить ресурсозависимость приложения и упростить разработку.

Второй класс данных целесообразно хранить как XML или JSON файл, в котором будут содержаться пары ключ-значение. Средства разработки предоставляют инструменты для работы с таким типом файлов, что позволит легко настроить взаимодействие приложения и файла с параметрами, а также даст возможность при необходимости обратиться напрямую к файлу для просмотра и редактирования.

И база данных, и параметры будет храниться на накопителе устройства, в специально выделенной для приложения папке в системном разделе, куда нет прямого доступа пользователю. Это обеспечит большую надежность от непреднамеренного повреждения информации пользователем.

## Раздел 2 Техническое задание

# 2.1 Постановка задачи

Необходимо разработать приложение, которое бы получало, хранило и обрабатывало электронные копии фискальных чеков.

В этом приложении должна быть возможность не только хранения самой копии чека, но и анализа информации, содержащейся в нём. Затем, на основе этого анализа данные должны группироваться по организациям, по торговым точкам, показываться статистика трат и динамика цен на определённый товар.

Также в приложении должен быть удобный интерфейс, который бы обеспечил быстрое взаимодействие с функционалом приложения. Приложение должно корректно отображаться на устройствах с различными размерами экранов

Программное изделие должно представлять собой мобильное приложение для платформы Android.

Приложение должно быть реализовано на языке Java версии 7. Для разметки активностей будет использован язык XML версии 1.0. Для разработки приложения будет использована интегрированная среда разработки Android Studio с SDK версии 22, что делает минимально поддерживаемой версией Android версию 5.1. Эта версия Андроид работает на 69,9% от всех мобильных устройств, что обеспечивает достаточную аудиторию для продукта. [2]

Само приложение должно быть интегрировано с официальным приложением Федеральной налоговой службы, так как ФНС не предоставляет возможности для регистрации и авторизации пользователей. Разрабатываемое приложение будет получать готовый JSON-файл копии фискального чека, который потом будет обрабатываться, анализироваться и отображаться пользователю в более удобной форме.

При этом также должна быть реализована возможность сканирования QR-кода на бумажном фискальном чеке и получения электронной копии с сервера Федеральной налоговой службы.

# 2.2 Концепция информационной базы и её структура

Для реализации программного продукта требуется реализовать необходимое информационное обеспечение, создающее условия для функционирования систем управления, обеспечение ее необходимой информацией, средствами поиска, накопления, хранения, обработки, передачи информации, организации банков данных. [3, с. 272]

Постановка задачи показывает, что в качестве СУБД для информационной базы должна выступать SQLite.

Создание базы данных сопряжено с двумя основными этапами работы: проектирование базы данных и непосредственное создание самой базы данных в определённой СУБД.

**Проектирование базы данных**

Проектирование внутренних структур данных происходит методом «сущность-связь» (ER-модель), в процессе этого будет разработана база данных для системы. На использовании разновидностей ER-модели основано большинство современных подходов к проектированию баз данных (главным образом, реляционных).

Были выделены четыре основные сущности – товар, покупка, точка продаж и магазин.

Детальное представление взаимодействия сущностей структуры можно рассмотреть на рисунке 1, где представлены ER-диаграммы данных и их связи.

Метод «сущность-связь» состоит из трёх этапов.

*Выделение сущностей:*

Товар (КодТовара, …);

Покупка (КодПокупки, …);

Точка продаж (КодТочки, …);

Магазин (КодМагазина, …).

*Выделение связей между сущностями и построение ER-диаграмм:*

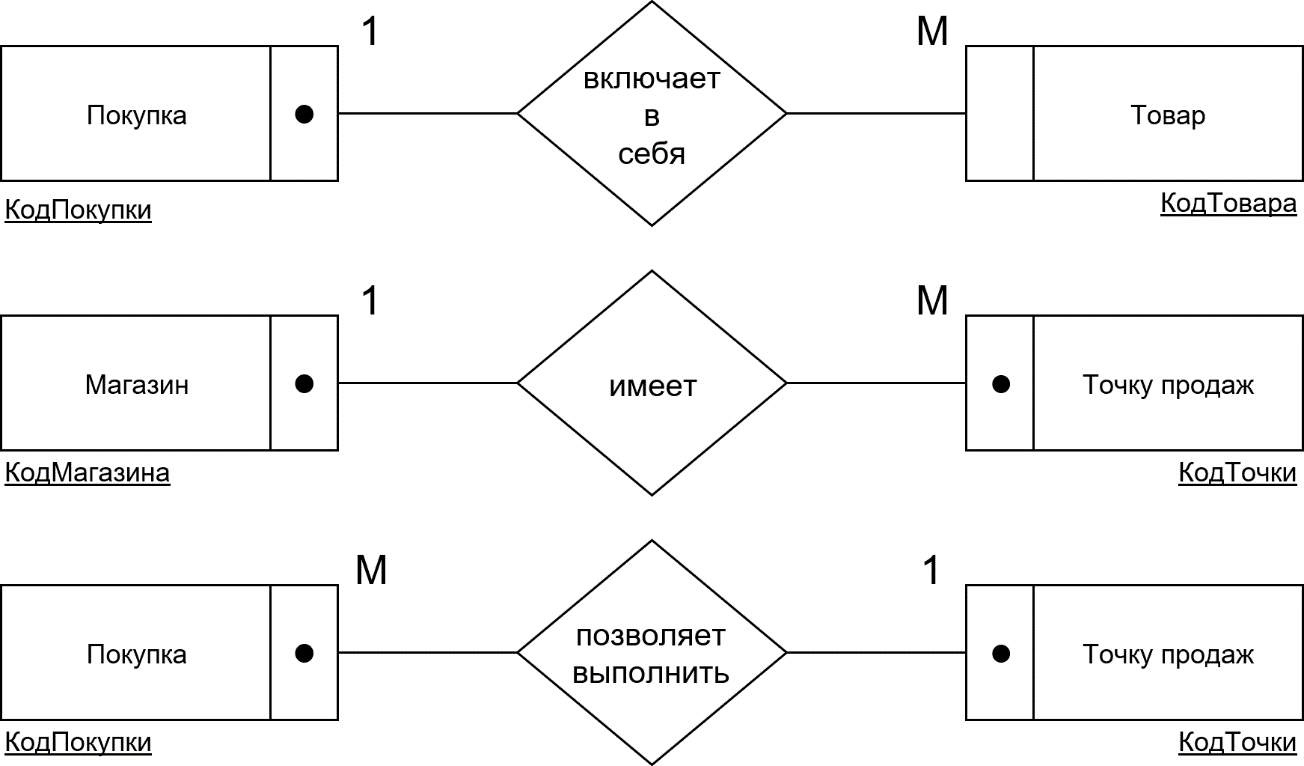


Рисунок 1 – ER-диаграмма

*Приведение предварительных отношений к усиленной З нормальной форме:*

Товар (КодТовара, Название, НДС10, НДС18, СуммаНДС10, СуммаНДС18, НомерНДС, Цена, КоличествоТовара, Сумма, НомерМагазина, КодПокупки);

Покупка (КодПокупки, НомерФискальногоДокумента, НомерФискальногоНакопителя, ПолнаяСуммаНаличными, ДатаВремя, СуммаСкидки, Метка, ПолнаяСуммаЭлектронДеньгами, ФискальнаяПодпись, КоличествоТоваров, НазваниеJSON, Оператор, НомерЗапроса, НомерСмены, РегистрационныйНомерККТ, НДС0, НДС10, НДС18, СуммаНДС10, СуммаНДС18, НомерНДС, ТипНалогообложения, ТипОперации, КодТочки, СписокТегов);

Точка продаж (КодТочки, АдресТочки, ПсевдонимТочки, КодМагазина);

Магазин (КодМагазина, НазваниеМагазина, ИНН, ПсевдонимМагазина);

**Создание базы данных**

Так как среда разработки не предлагает пользовательского интерфейса для удобного создания базы данных, было решено создать спроектированную базу данных в сервисе dbDesigner, что позволит получить готовый SQL-код для создания базы данных в приложении. Схема созданной базы данных изображена на рисунке 2.

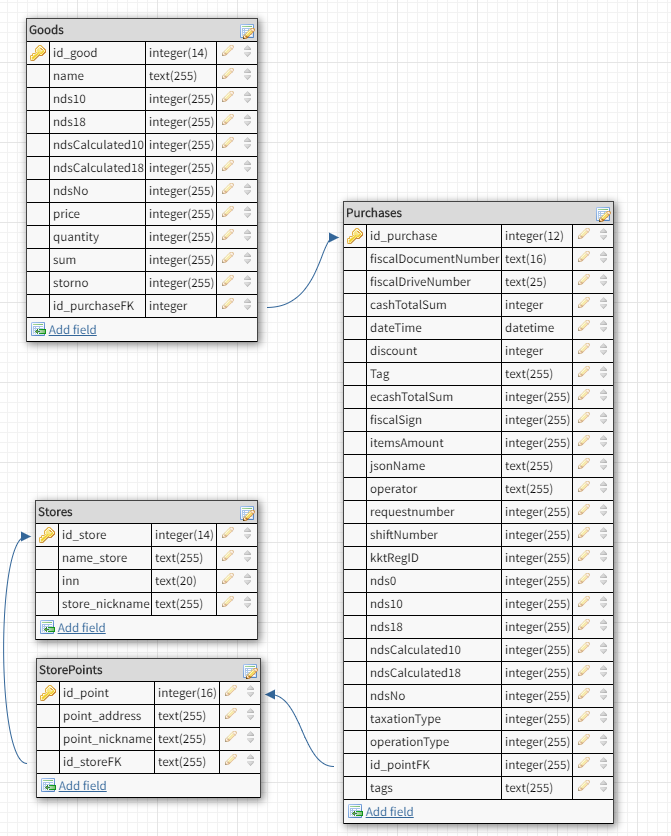


Рисунок 2 – созданные таблицы со связями

# 2.3 Неформальное описание алгоритма работы программного продукта

В программном продукте в режиме просмотра возможно выполнение следующий действий:

* просмотр списка покупок;
* просмотр списка товаров;
* просмотр списка магазинов;
* вывод детальной информации о покупках;
* вывод детальной информации о товарах;
* вывод детальной информации о магазинах;

В режиме редактирования возможно выполнение следующих действий:

* добавление фискального чека;
* изменение псевдонима организации;
* изменение псевдонима торговой точки;
* удаление фискального чека;

1. При нажатии на кнопку добавления фискального чека открывается активность со сканером QR-кода и кнопкой импорта JSON-файла. После сканирования QR-кода на кассовом чеке происходит получение подробной информации о содержимом чека, после чего вся информация добавляется в базу данных. При использовании импорта проводится проверка JSON-файла на соответствие 54-ФЗ, после успешного прохождения которой, вся информация добавляется в базу данных.
2. При выборе одной из трёх вкладок в нижнем меню приложения, происходит отображение списка одной из категорий: товары, покупки или магазины.
3. При нажатии на один из товаров в списке товаров отобразится детальная информация о нём.
4. При нажатии на одну из покупок в списке покупок отобразится детальная информация о покупке.
5. При нажатии на один из магазинов отобразится детальная информация об этом магазине.
6. При нажатии на кнопку «Переименовать» в активности просмотра информации о магазине будет открыта форма, где можно ввести желаемый псевдоним магазина или удалить текущий. После нажатия на кнопку «Сохранить» новый псевдоним будет добавлен в базу данных.
7. При нажатии на кнопку «Переименовать» в активности просмотра информации о торговой точке будет открыта форма, где можно ввести желаемый псевдоним торговой точки или удалить текущий. После нажатия на кнопку «Сохранить» новый псевдоним будет добавлен в базу данных.
8. При нажатии на кнопку «Удалить» в активности просмотра детальной информации о покупке, запись с этим фискальным чеком, а также сопутствующие записи о товарах будут удалены из базы данных.

Блок-схема алгоритма представлена в приложении А.

# 2.4 Требования к контролю, обновлению и восстановлению данных, защите данных от разрушений

Сохранность информации в системе должна обеспечиваться при следующих аварийных ситуациях:

* импульсные помехи, сбои и перерывы в электропитании;
* сбой общесистемного ПО;
* полный или частичный отказ технических средств системы, включая сбои и отказы накопителей;
* ошибки в использовании ПО пользователями.

Программа должна быть обеспечена контролем за вводом данных:

* после сканирования QR-кода должна выполняться проверка на соответствие ФЗ-54;
* перед импортом JSON-файла кассового чека выполняется проверка содержимого файла на соответствие ФЗ-54.

Обновление структуры базы данных производится с выполнением резервного копирования и корректного версионирования базы данных.

Данные приложения должны сохраняться на сервере корпорации Google, что даст возможность восстановления данных при сбросе системы или переходе на новое устройство.

# 2.5 Функции подсистем (модулей)

Программное изделие состоит из 13 модулей, которые изображены на рисунке 3.

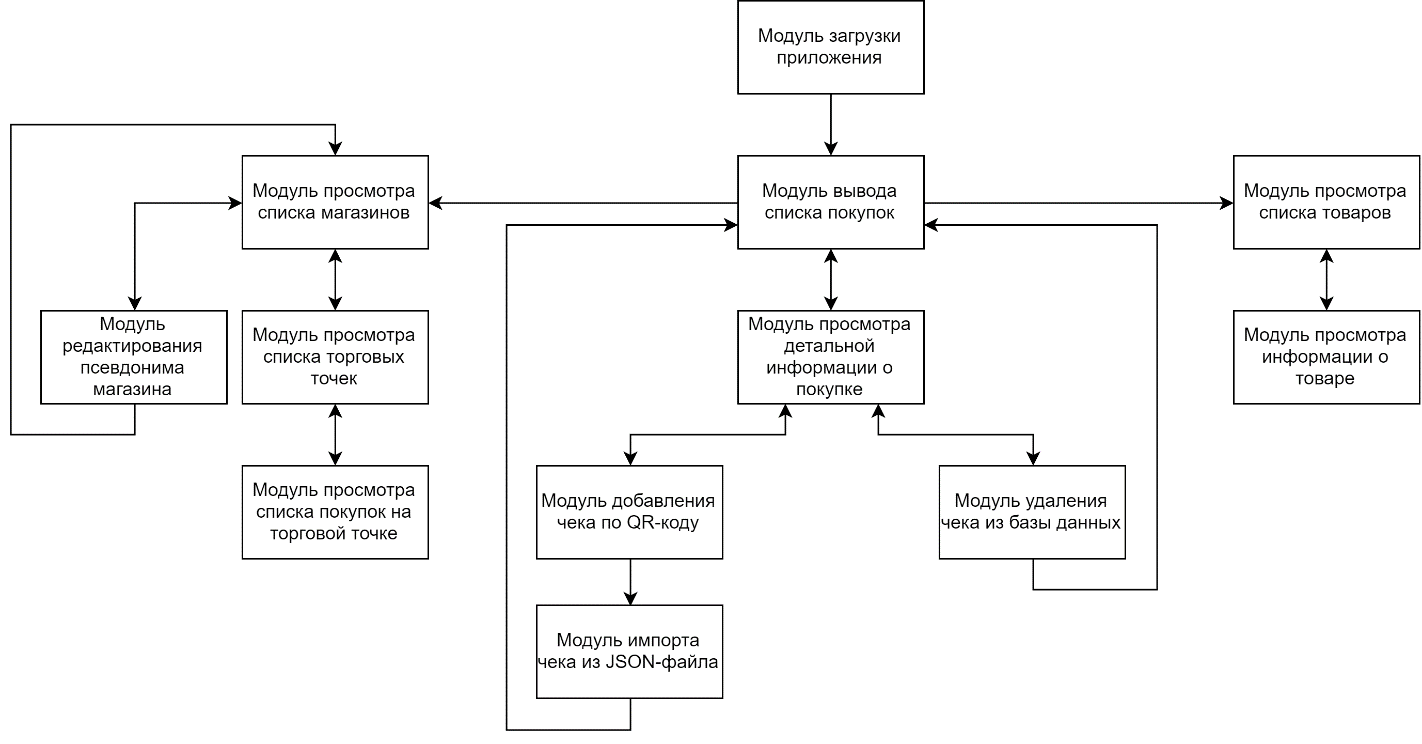


Рисунок 3 – модули программного изделия

Модули программного изделия:

1. Модуль загрузки приложения: инициализация базы данных системы, активация активности с меню.
2. Модуль вывода списка покупок: обращение к базе данных и вывод записей таблицы «Покупки».
3. Модуль просмотра списка магазинов: обращение к базе данных и вывод записей таблицы «Магазины».
4. Модуль просмотра списка товаров: обращение к базе данных и вывод записей таблицы «Товары»
5. Модуль просмотра детальной информации о покупке: обращение к базе данных и вывод атрибутов записи таблицы «Покупки».
6. Модуль просмотра информации о товаре: обращение к базе данных и вывод атрибутов записи таблицы «Товары».
7. Модуль просмотра торговых точек: обращение к базе данных и вывод записей таблицы «Точки».
8. Модуль редактирования псевдонима магазина: отображение формы с полем редактирования псевдонима магазина и изменение значения в базе данных.
9. Модуль просмотра списка покупок на торговой точке: обращение к базе данных и вывод записей таблицы «Покупки» с соответствующим значением атрибута «Торговая точка».
10. Модуль добавления чека по QR-коду: сканирования QR-кода с помощью камеры, проверка кода на валидность, получение данных о кассовом чеке с сервера Федеральной налоговой службы и добавление в базу данных.
11. Модуль импорта чека из JSON-файла: запуск проводника файлов на устройстве. Проверка JSON-файла на валидность и последующее добавление в базу данных.
12. Модуль удаления чека из базы данных: поиск нужной записи в базе данных и удаление этой записи из БД.

# 2.6 Оценка затрат на разработку программного продукта

Время рассчитывается в человеко-часах, причем Тпо (подготовка описания задачи) берется по фактически отработанному времени, а время остальных этапов определяется по расчётному условному числу команд Q.

Условное число команд Q определяется по формуле 1:

Q = q \* C; (1)

где q - коэффициент, учитывающий условное число команд в зависимости от типа задачи. Значения коэффициента q указаны ниже в таблице 1:

Таблица 1

|  |  |
| --- | --- |
| Тип задачи | Пределы изменений коэффициента |
| Задачи учета | от 1400 до 1500 |
| Задачи оперативного управления | от 1500 до 1700 |
| Задачи планирования | от 3000 до 3500 |
| Многовариантные задачи | от 4500 до 5000 |
| Комплексные задачи | от 5000 до 5500 |

Для данной задачи коэффициент q принимает значение = 1400.

С - коэффициент, учитывающий новизну и сложность программы.

Программные продукты по степени новизны могут быть отнесены к одной из 4-х групп:

группа А - разработка принципиально новых задач;

группа Б - разработка оригинальных программ;

группа В - разработка программ с использованием типовых решений.

группа Г- разовая типовая задача.

Для данной задачи степень новизны можно отнести к группе А.

По степени сложности программные продукты могут быть отнесены к одной из 3-х групп:

* алгоритмы оптимизации и моделирования систем;
* задачи учета, отчетности и статистики;
* стандартные алгоритмы.

Коэффициент C определяется из таблицы 2, на пересечении групп сложности и степени новизны.

Таблица 2 - Коэффициенты расчета трудоемкости

| Язык Программирования | Группа  Сложности | Степень новизны | | | |
| --- | --- | --- | --- | --- | --- |
| А | Б | В | Г |
| Высокого  Уровня | 1 | 1,38 | 1,26 | 1,15 | 0,69 |
| 2 | 1,30 | 1,19 | 1,08 | 0,65 |
| 3 | 1,20 | 1,15 | 1,00 | 0,60 |
| Низкого  Уровня | 1 | 1,58 | 1,45 | 1,32 | 0,79 |
| 2 | 1,49 | 1,37 | 1,24 | 0,74 |
| 3 | 1,38 | 1,26 | 1,15 | 0,69 |

Данная задача может быть отнесена к 2 группе сложности.

Для данной задачи коэффициент С по таблице принимается = 1,15. Теперь, исходя из формулы, можно определить условное число команд

Q = 1400\*1,15 =1610;

Определяем время, затраченное на каждый этап создания программного продукта:

1. Тпо (время на подготовку описания задачи), берется по факту и составляет:

Тпо = 3 чел. / час.

2. То (время на описание задачи) определяется по формуле:

То = Q \* B / (50 \* K); (2)

Где B – коэффициент учета изменений задачи, коэффициент B в зависимости от сложности задачи и числа изменений выбирается в интервале от 1,2 до 1,5. Для данной задачи B = 1,2.

K – коэффициент, учитывающий квалификацию программиста.

Выбрать значение коэффициента. К можно из таблицы 3

Таблица 3 - Значение коэффициента К.

|  |  |
| --- | --- |
| Стаж программиста | Значение коэффициента К |
| до 2-х лет | 0,8 |
| от 2 до 3 лет | 1,0 |
| от 3 до 5 лет | 1,1 - 1,2 |
| от 5 до 10 лет | 1,2 - 1,3 |
| свыше 10 лет | 1,3 - 1,5 |

В данном случае коэффициент К = 0,8.

Применяя формулу 2 подсчитываем время на описание задачи.

То = 1610 \* 1.2 / (50 \* 0,8) = 48,3 (чел/час)

3. Та (время на разработку алгоритма) рассчитываем по формуле:

Та = Q / (50 \* K); (3)

Применяя формулу 3 подсчитываем время на разработку алгоритма.

Та =1610 / (50 \* 0,8) = 40,25 (чел/час)

4. Тбс (время на разработку блок схемы) определяется аналогично Та по формуле 3 и составляет:

Тбс = 40,25 (чел/час);

5. Тн (время написания программы на языке программирования) определяется по формуле:

Тн = Q \* 1.5 / (50 \* K); (4)

Применяя формулу 4 подсчитываем время написания программы на языке программирования:

Тн =1610 \* 1,5 / (50 \* 0,8) = 60,4 (чел/час);

6. Тот (время отладки и тестирования программы) определяется по формуле:

Тот = (Q \* 4,2) / (50 \* K); (6)

Подставляем значения в формулу 6 и получаем:

Тот = (1610 \* 4,2) / (50 \* 0,8) = 169 (чел/час);

7. Тд (время на оформление документации), берется по факту и составляет (от 3-х до 5-ти дней по 8 часов):

Тд = 24 чел. / час.

Теперь, зная время, затраченное на каждом этапе, можно подсчитать общее время на создание программного продукта:

Т = Тпо + То + Та + Тбс + Тн + Тот + Тд = 344,6 (чел/час)

Расчет заработной платы исполнителя работы по созданию программного продукта:

Основная ЗП определяется по формуле: ЗПосн. = 31 ∗ Кт ∗ ТЧр ∗ 𝑡р.д. ∗ (1 + П100) (руб.),

Где:

1) 31 – месячная зарплата 1-го разряда (руб.);

2) Кт – тарифный коэффициент, соответствующий разряду тарифной сетки по которому, работает исполнитель;

3) Т – общее время на создание программного продукта (чел/час);

4) Чр – общее число рабочих дней в месяце;

5) t р. д. – продолжительность рабочего дня в часах;

6) П – процент премии если есть;

7) t р. д. = 8 часов;

8) Чр = 21 день;

9) 31 = 14000 руб.

10) Разряд исполнителя - 1, его тарифный коэффициент = 1,0.

Таким образом определяем основную заработную плату исполнителя работ по созданию программного продукта.

ЗПосн = (20000 \* 1,0 \* 344,6 / 21 \* 8) \* (1 + 10%) = 45 126,19 (руб.)

Дополнительная заработная плата берется в размере 15 % от

основной.

ЗПдоп = 6 768,92 (руб.)

Общая заработная плата будет равна сумме основной и дополнительной:

ЗПобщая = 45 126,19 + 6 768,92= 51 895,11 (руб.)

## Раздел 3 Эскизный проект

Приложение состоит из 7 экранов. Пользовательский интерфейс должен иметь удобную навигацию по основным пунктам меню. Поэтому интерфейс приложения поделен на три основных раздела: «Товары», «Покупки» и «Магазины». Переключение между разделами осуществляется с помощью нижней навигационной панели. Экран покупок представлен на рисунке 4. Список покупок является основным экраном, который отображается при запуске приложения, показан на рисунке 5. Экран со списком магазинов показан на рисунке 6. Экран с информацией о магазине показан на рисунке 7. Экран с информацией о торговой точке показан на рисунке 8. Экран с информацией о товаре показан на рисунке 9. Экран с информацией о покупке (фискальном чеке) показан на рисунке 10.

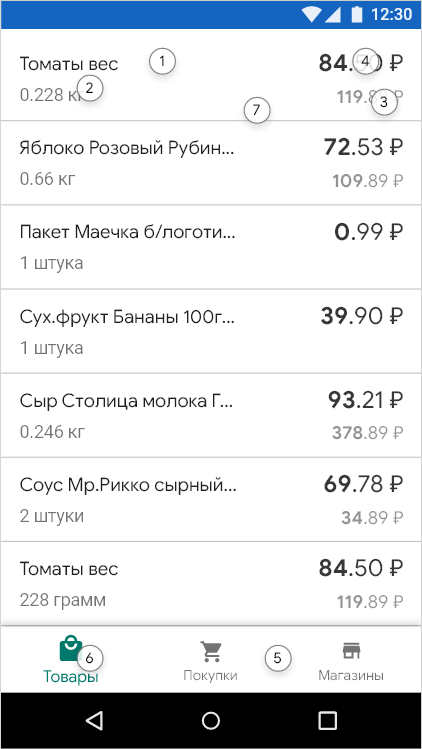


Рисунок 4 – экран списка товаров

1. название товара;
2. количество товара;
3. цена товара;
4. стоимость товара
5. навигационная панель;
6. индикатор активного раздела;
7. разделитель.

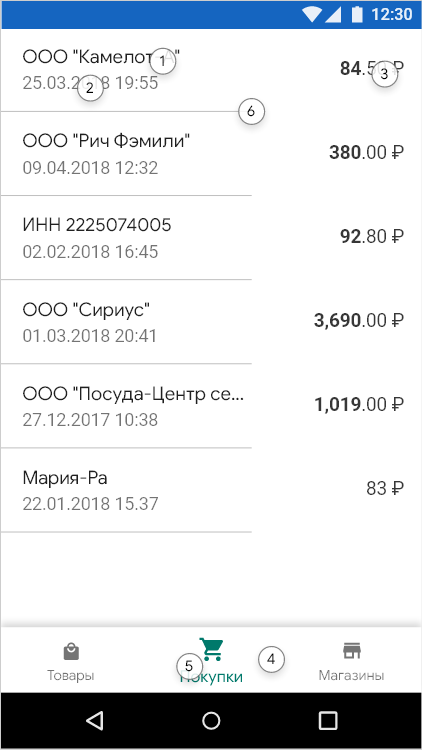


Рисунок 5 – главный экран приложения - список покупок

1. пользовательское название, юридическое название, ИНН организации;
2. дата и время покупки;
3. сумма покупки;
4. навигационная панель;
5. индикатор активного раздела;
6. разделитель.

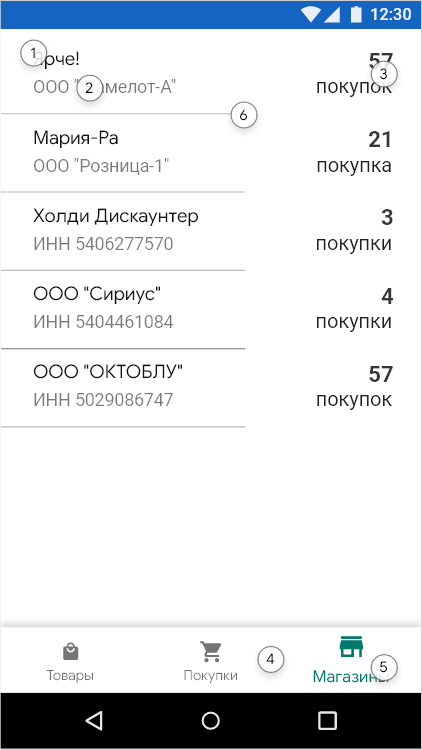


Рисунок 6 – экран списка магазинов

1. пользовательское название, юридическое название, ИНН организации;
2. юридическое название, ИНН организации;
3. количество покупок;
4. навигационная панель;
5. индикатор активного раздела;
6. разделитель.

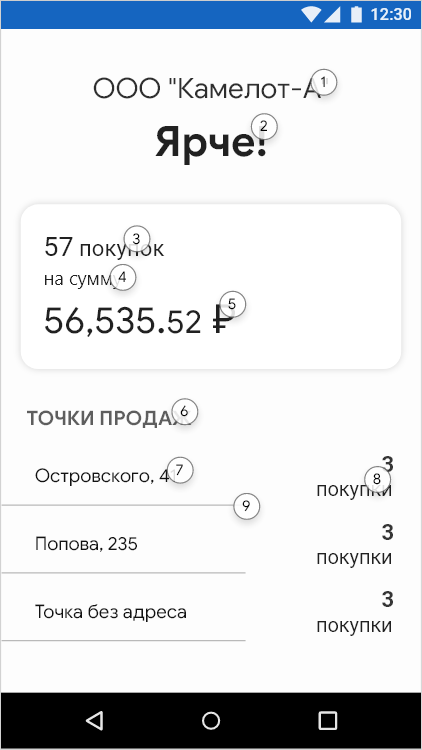


Рисунок 7 – экран информации о магазине

1. юридическое название, ИНН организации;
2. пользовательское название, юридическое название, ИНН организации;
3. общее количество покупок;
4. надпись «на сумму»;
5. общая сумма покупок;
6. надпись «Точки продаж»;
7. адрес торговой точки;
8. количество покупок;
9. разделитель.

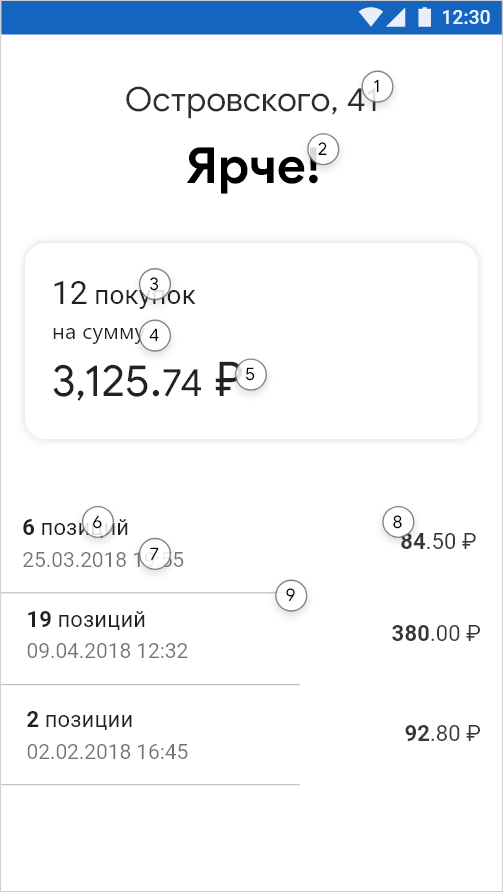


Рисунок 8 – экран информации о торговой точке

1. адрес торговой точки;
2. пользовательское название, юридическое название, ИНН организации;
3. общее количество покупок;
4. надпись «на сумму»;
5. общая сумма покупок;
6. количество позиций в чеке;
7. дата покупки;
8. сумма покупки;
9. разделитель.

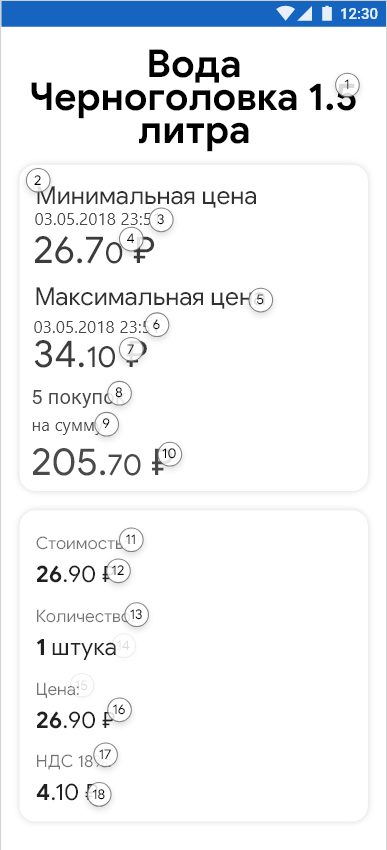


Рисунок 9 – экран информации о товаре

1. название товара;
2. надпись «Минимальная цена»;
3. дата минимальной цены товара
4. минимальная цена товара;
5. надпись «Максимальная цена»;
6. дата максимальной цены товара
7. максимальная цена товара;
8. количество приобретений товара;
9. надпись «на сумму»;
10. общая сумма приобретений товара;
11. надпись «Стоимость:»;
12. стоимость товара;
13. надпись «Количество:»;
14. количество товара;
15. надпись «Цена»;
16. цена товара;
17. надпись «НДС 10%», «НДС 18%»;
18. сумма НДС.

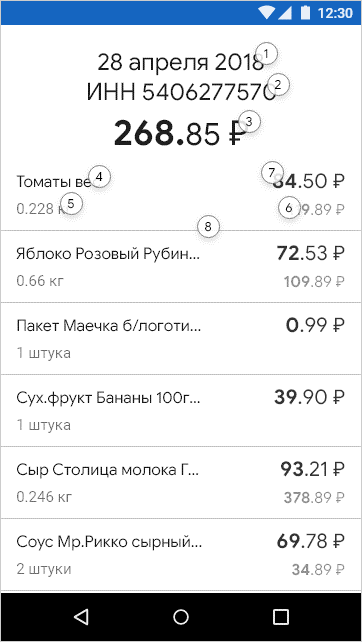


Рисунок 10 – экран просмотра покупки (фискального чека)

1. дата покупки;
2. пользовательское название, юридическое название, ИНН организации;
3. сумма покупки;
4. название товара;
5. количество товара;
6. цена товара;
7. стоимость товара;
8. разделитель.

**Контрольный пример**

При открытии приложения пользователю отображается список его покупок, который показан на рисунке 11. Пользователь может перейти на экран отображения списка товаров или списка магазинов с помощью нижнего меню. Экраны отображения списка товаров и списка магазинов показаны на рисунках 12 и 13 соответственно. При нажатии на круглую кнопку с изображением знака «плюс» пользователь попадает в сканер QR-кода, интерфейс которого представлен на рисунке 14. После сканирования и успешного добавления данных о чеке в базу данных, пользователю выводится список покупок.

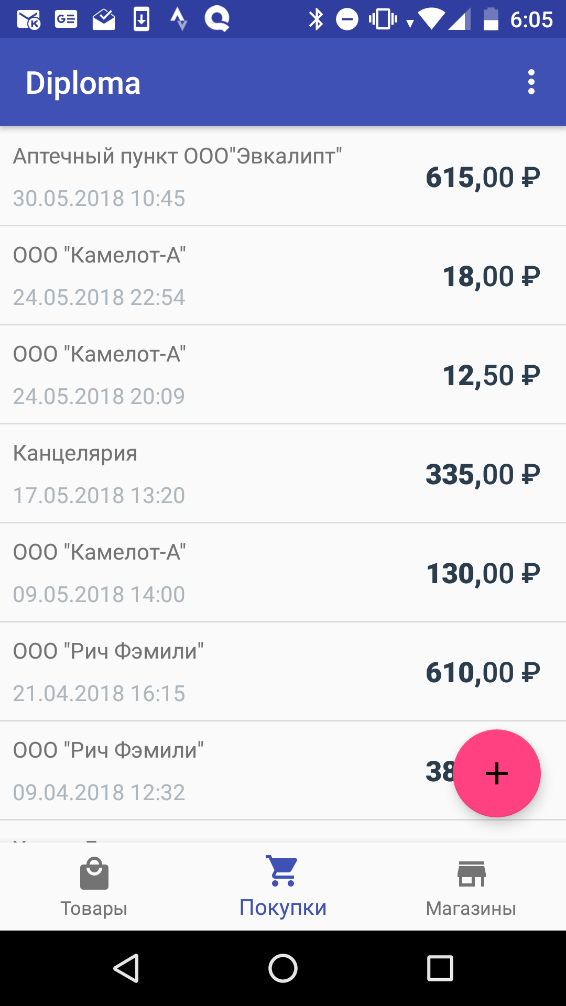


Рисунок 11 – экран списка покупок

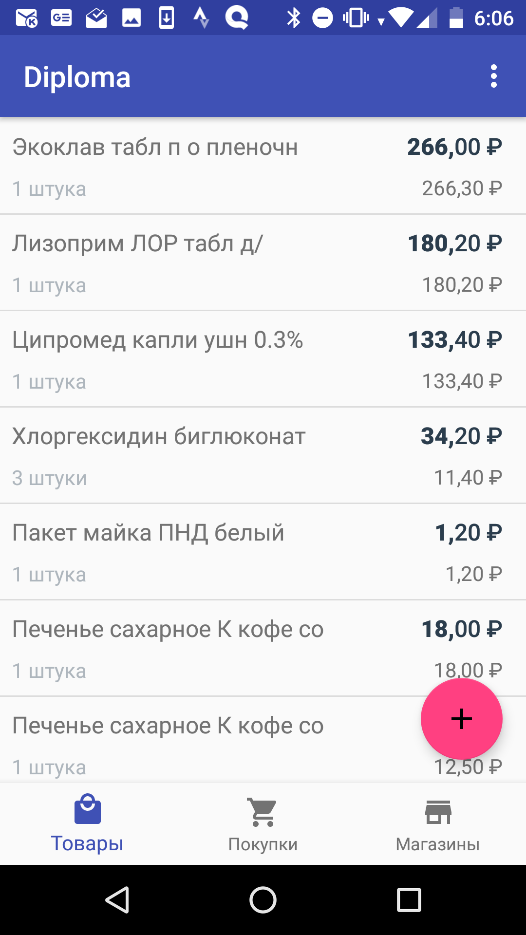


Рисунок 12 – экран списка товаров

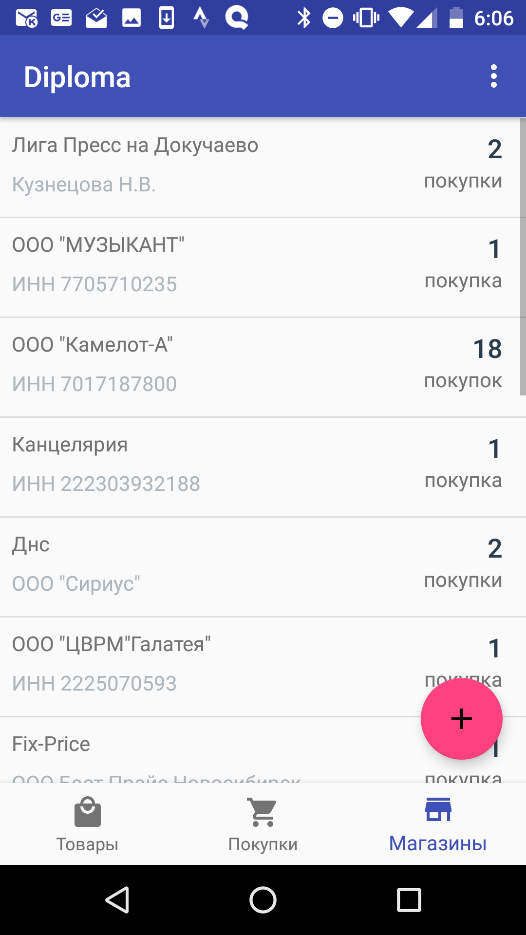


Рисунок 13 – экран списка магазинов

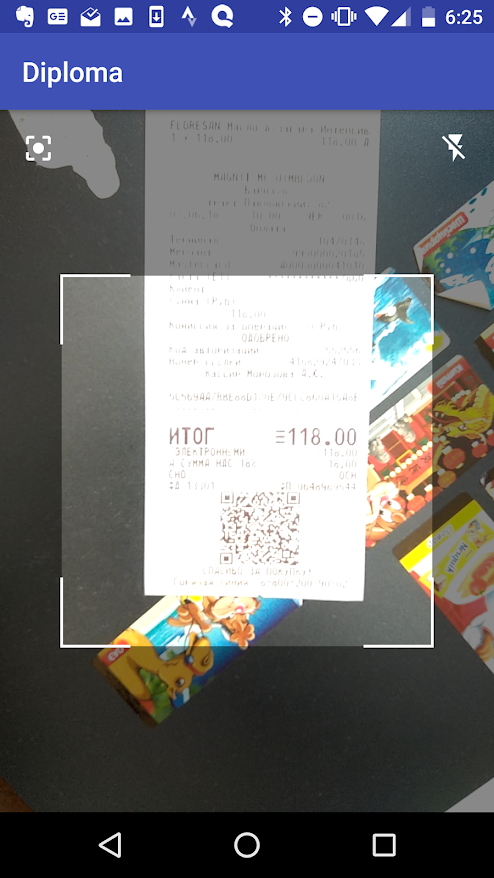


Рисунок 14 – окно сканера QR-кода

Нажав на любую из покупок, пользователь попадает на экран просмотра информации о покупке, который показан на рисунке 15. На этом экране можно нажать на любую из позиций, чтобы просмотреть детальную информацию и статистику об этом товаре, что показано на рисунке 16. Детальную информацию о товаре также можно посмотреть, нажав на один из товаров в списке товаров.

Информацию о магазине можно просмотреть, нажав на него в списке магазинов. Экран с информацией о магазине показан на рисунке 17. На этом экране присутствует список торговых точек этого магазина. Пользователь может нажать на одну из точек, после чего пользователю отобразится информация о точке и всех покупок на ней. Экран просмотра информации о точке показан на рисунке 18.

В приложении также присутствует контекстное меню. На экранах просмотра списков покупок, магазинов и товаров в меню есть опция «Справка», при нажатии на которую пользователю будет отображена инструкция по использованию программы. На экране просмотра информации о покупке в контекстное меню добавляется опция «Удалить», а на экранах просмотра информации о магазине и точке продаж добавляется опция переименовать, которая выводит диалоговое окно, где пользователь может ввести новое желаемое имя выбранного магазина или торговой точки. Диалоговое окно показано на рисунке 19.

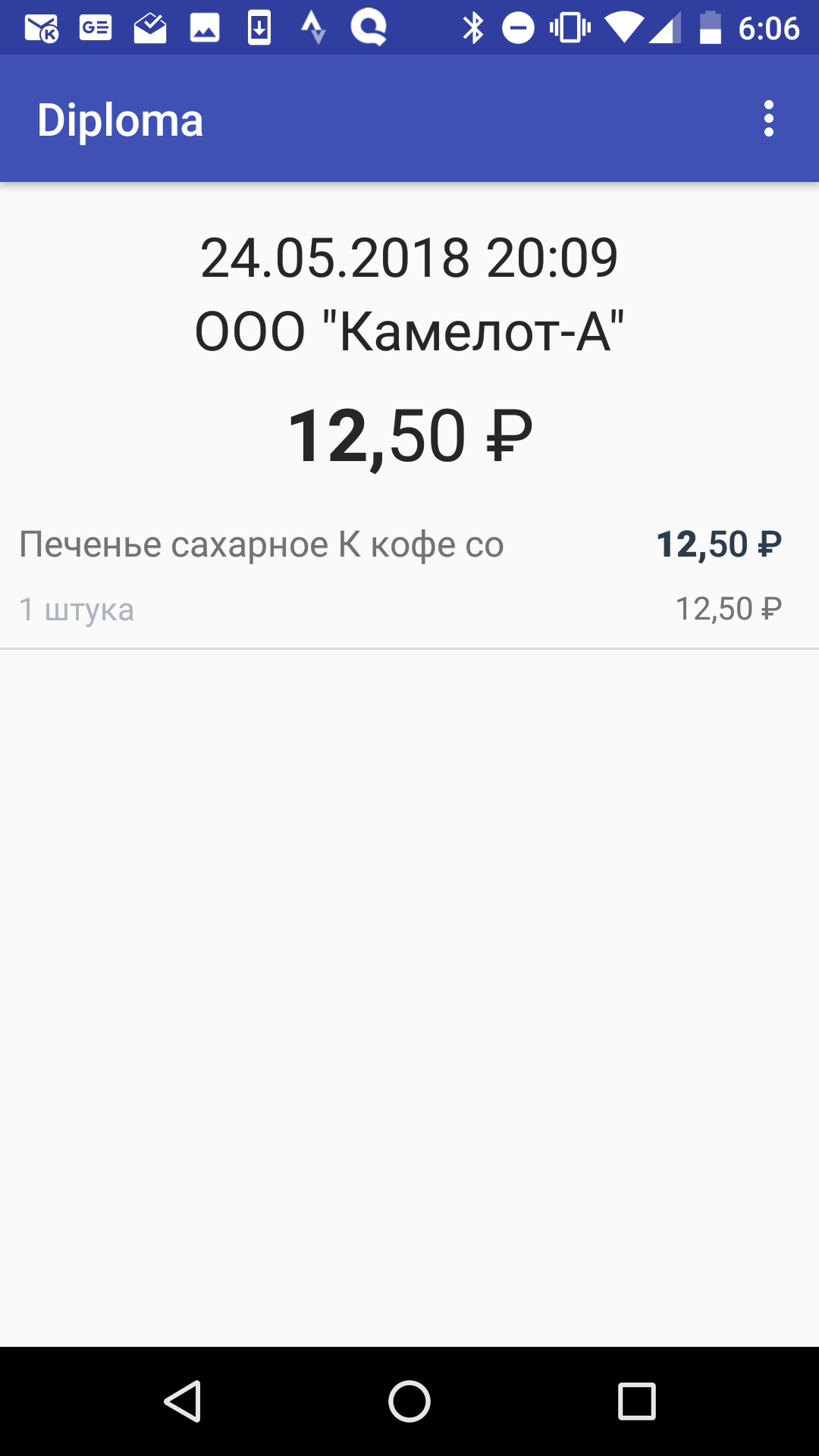


Рисунок 15 – экран с информацией о покупке

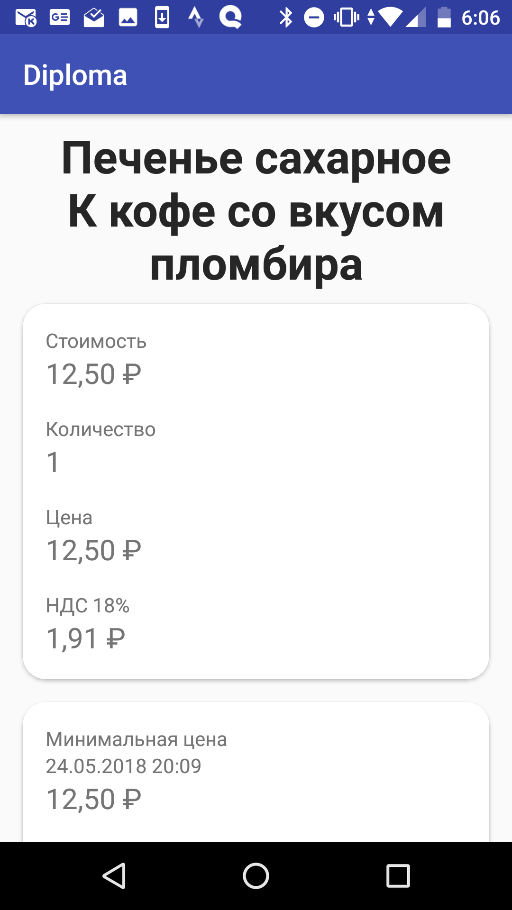


Рисунок 16 – экран с информацией о товаре

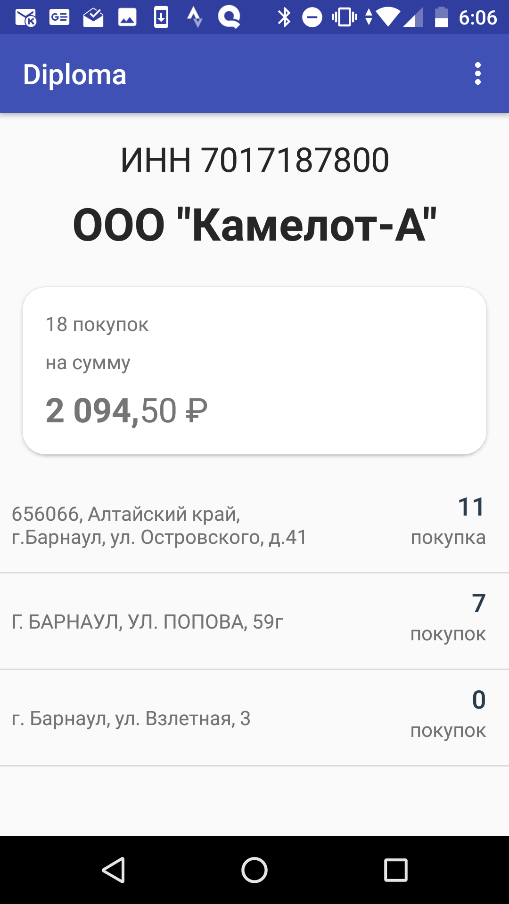


Рисунок 17 – экран с информацией о магазине

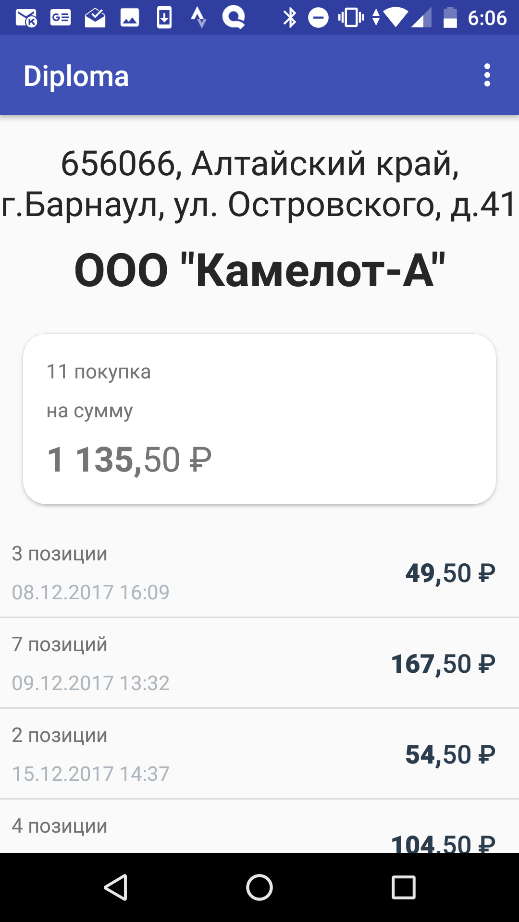


Рисунок 18 – экран с информацией о точке

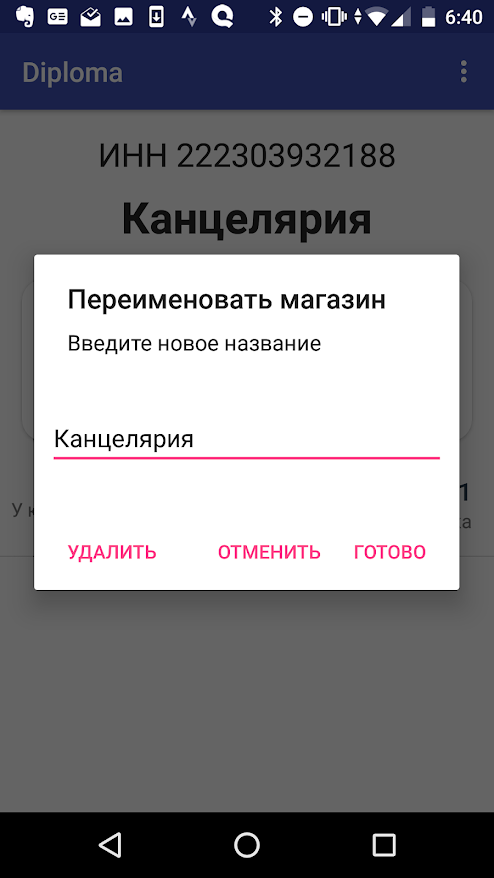


Рисунок 19 – диалоговое окно переименования магазина

## Раздел 4 Разработка проектных решений

Программное изделие было разработано при помощи языка объектно-ориентированного программирования Java 7 в интегрированной среде разработки Android Studio. Данный выбор обусловлен выбором целевой платформы – операционной системы Android. Язык Java и среда разработки Android Studio являются основными средствами разработки для данной платформы. Разметка пользовательского интерфейса была выполнена c помощью языка разметки XML, который обеспечивает удобное управление визуальными объектами без программного задания свойств. [4, с. 183]

При разработке было использовано модульное программирование. Каждый модуль представляет собой отдельный Java-класс. Некоторые из классов наследуют встроенные классы-библиотеки Java 7. Также при разработке были использованы внешние библиотеки Google Gson и Code Scanner.

Данные в приложении хранятся локально в базе данных. Для хранения используется СУБД SQLite. Для создания базы данных был использован онлайн-сервис dbDesigner.net. После создания таблиц и связей база была экспортирована в качестве констант в Android Studio. Средой разработки был автоматически создан класс, который обеспечивает создание и взаимодействие с базой данных.

В приложении доступны две точки входа – прямой запуск приложения пользователем и с помощью передачи интента из официального приложения ФНС. Во втором случае класс jsonParser принимает интент и запускает активность разбора Json-файла и добавления данных в БД, после чего происходит переход в основное окно приложения.

Для сборки проекта использовался Gradle – система автоматическое сборки программного обеспечения, которая обеспечивала совместимость компонентов и автоматическое подключение необходимых компонентов.

Целевой платформой выступал Android c версией API 26 и выше (Android 8.0+), но была обеспечена совместимость с более низкими версиями, вплоть до API 21 (Android 5.1). Выбор этих версий API позволяет использовать наибольшее количество актуальных библиотек для работы с объектами системы Android, что существенно упрощает разработку, при этом практически не теряя в охвате устройств. [5, c. 136]

Таким образом, для функционирования приложения на устройствах с версией ОС Android ниже 8.0 требуется установленный пакет Google Play Services, который доступен для скачивания в магазине приложений Google Play. Также требуется наличие камеры и доступа к сети интернет. Полные системные требования описаны в руководстве программиста (Приложение В).

В качестве справочной системы для пользователя была использована веб-страница с гиперссылками, интегрированная в приложение. Веб-страница будет отображаться в отдельной активности, в объекте WebView. Сама веб-страница при этом будет располагаться в сервисе GitHub.io, что позволит легко и быстро вносить правки при необходимости.

## Заключение

В ходе создания дипломного проекта было разработано программное изделия для широкого круга лиц.

Реализация программного изделия совпадает с требованиями, полученными в процессе разработки технического задания. Приложение соответствует требованиям целевой платформы выполняет следующие условия:

* является функционально полным;
* является устойчивым к различными факторам, таким как отсутствие интернет-соединения или ошибка в QR-коде;
* является простым и удобным в использовании;
* является безопасным и толерантным к среде выполнения, так как работает в отдельном экземпляре виртуальной машины Java.

Приложение успешно реализует свою главную функцию – внедрение электронного документооборота и отказ от хранения бумажных фискальных накопителей с ведением статистики расходов.

Тем не менее, в будущем приложение должно развиваться для большего охвата пользователей и внедрения в различные экосистемы электронного документооборота. Так, например, целесообразно сделать функционал экспорта данных об одном или нескольких чеках в другие системы, такие как 1C, Microsoft Excel и Microsoft Access. Приложение будет опубликовано в магазине Google Play для более простой дистрибуции установочного пакета и монетизации приложения. Также, при наличии ресурсов, возможно создание централизованной базы данных чеков пользователей, на основе которой будет возможно сопоставление товаров и поиск минимальной цены в магазинах.

## Список использованных источников

1. Федеральный закон от 22.05.2003 N 54-ФЗ (ред. от 03.07.2016) "О применении контрольно-кассовой техники при осуществлении наличных денежных расчетов и (или) расчетов с использованием электронных средств платежа" [Электронный ресурс]. – Режим доступа: http://www.consultant.ru/document/cons\_doc\_LAW\_42359/, свободный. – Загл. с экрана.
2. NetMarketShare [Электронный ресурс]. – Режим доступа: https://netmarketshare.com/operating-system-market-share.aspx?id=platformsMobile, свободный. – Загл. с экрана.
3. Борис, Райзберг. Современный экономический словарь / Райзберг. Борис, Лозовский. Леонид, Стародубцева. Елена. – М.: Инфра-М, 2016. – 526 c.
4. Гонсалвес, Э. Изучаем Java EE 7 / Э. Гонсалвес. – СПб.: Питер, 2018. – 640 c.
5. Филлипс, Б. Android. Программирование для профессионалов. 3-е издание / Б. Филлипс, К. Стюарт, К. Марсикано. – СПб.: Питер, 2018. – 688 c.

## Приложение А Блок-схема алгоритма программы

Блок-схема представлена на рисунках 20 и 21.

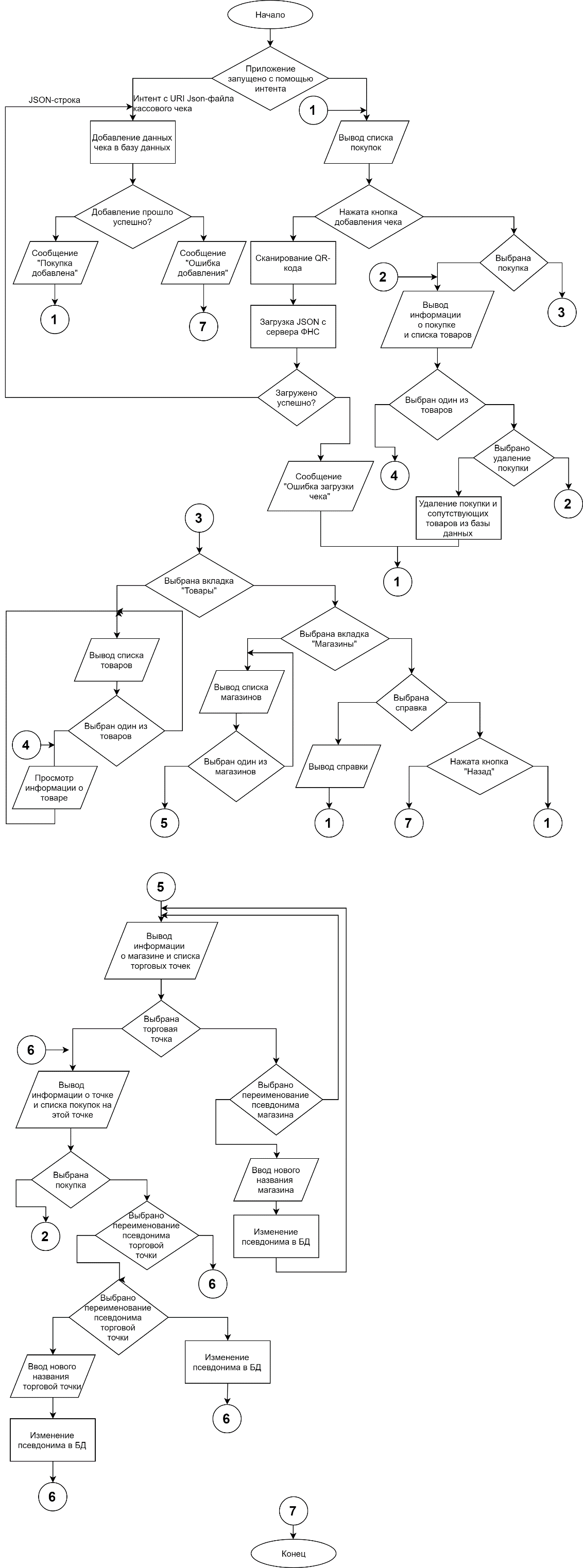


Рисунок 20 – блок-схема

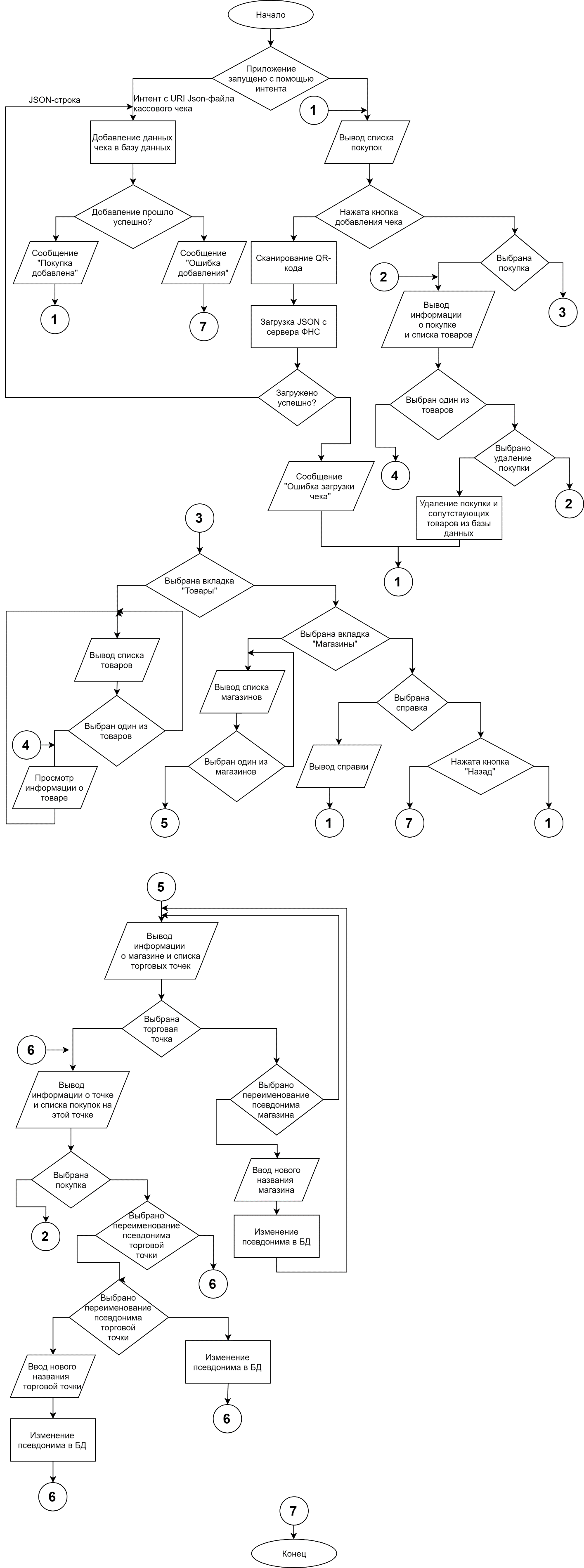


Рисунок 21 – блок-схема, продолжение

## Приложение Б Документация пользователя

**Запуск приложения**

Для запуска приложения необходимо нажать на его иконку в меню приложений или добавить новый кассовый чек путем экспорта из официального приложения ФНС.

**Добавление нового кассового чека**

Добавить новый кассовый чек можно двумя способами:

1. с помощью сканера QR-кодов;
2. экспортируя чек из официального приложения Федеральной налоговой службы.

Для добавления чека путём сканирования QR-кода, нужно нажать на кнопку добавления чека (круг со знаком «плюс») на любом из трёх главных экранов. После этого откроется окно сканера. Теперь нужно навести камеру телефона на QR-код расположенный на кассовом чеке. После этого чек будет добавлен в список покупок.

Для добавления чека путём экспорта из официального приложения ФНС нужно проделать следующие действия:

1. Открыть приложение ФНС;
2. Выбрать необходимый чек;
3. Нажать на кнопку «меню»;
4. Выбрать пункт «Отправить чек»;
5. Выбрать пункт «json»;
6. В открывшемся окне выбрать приложение «Diploma». После этого чек будет экспортирован.

**Просмотр списка покупок**

Для просмотра списка покупок необходимо нажать на пункт «Покупки» в нижней навигационной панели на главном экране приложения.

**Просмотр списка товаров**

Для просмотра списка покупок необходимо нажать на пункт «Товары» в нижней навигационной панели на главном экране приложения.

**Просмотр списка магазинов**

Для просмотра списка покупок необходимо нажать на пункт «Магазины» в нижней навигационной панели на главном экране приложения.

**Просмотр покупки (фискального чека)**

Для просмотра покупки (фискального чека) необходимо нажать на желаемую покупку (фискальный чек) в разделе «Покупки» на главном экране приложения.

**Просмотр информации о магазине**

Для просмотра информации о магазине необходимо нажать на желаемый магазин в разделе «Магазины» на главном экране приложения

**Просмотр информации о торговой точке**

Для просмотра информации о торговой точке необходимо сначала выбрать магазин, которому принадлежит торговая точка. Для этого нужно нажать на желаемый магазин в разделе «Магазины» на главном экране приложения. Теперь можно в списке торговых точек найти нужную и нажать на неё.

**Переименование магазина**

Для переименования необходимо нажать на желаемый магазин в разделе «Магазины» на главном экране приложения. Затем нужно нажать на кнопку меню «три точки» и выбрать пункт «Переименовать». В открывшемся окне следует ввести новое имя магазина и нажать «Готово». Для удаления пользовательского названия в диалоговом окне нужно нажать кнопку «Удалить».

**Переименование торговой точки**

Для переименования торговой точки необходимо сначала выбрать магазин, которому принадлежит торговая точка. Для этого нужно нажать на желаемый магазин в разделе «Магазины» на главном экране приложения. Теперь следует найти в списке торговых точек нужную торговую точку и нажать на неё. Затем нужно нажать на кнопку меню «три точки» и выбрать пункт «Переименовать». В открывшемся окне следует ввести новое имя магазина и нажать «ГОТОВО». Для удаления пользовательского названия в диалоговом окне нужно нажать кнопку «Удалить».

**Удаление покупки (фискального чека)**

Для просмотра покупки (фискального чека) необходимо нажать на желаемую покупку (фискальный чек) в разделе «Покупки» на главном экране приложения. Затем нужно нажать на кнопку меню «три точки» и выбрать пункт «Удалить», и в открывшемся диалоговом окне выбрать пункт «ОКЕЙ».

Внимание: помните, что удалённые данные невозможно восстановить.

**Завершение работы с приложением**

Для завершения работы с приложением нужно нажать на кнопку «назад» на главном экране приложения или убрать приложение из меню многозадачности.

## Приложение В Документация программиста

**Требования к устройству**

Для запуска и корректной работы приложения, устройство должно обладать следующими характеристиками:

* операционная система Android версии 5.1 или выше;
* установленный пакет Google Play Services;
* процессор с частотой не менее 800 МГц;
* не менее 32 Мбайт свободной памяти во время запуска приложения;
* не менее 128 Мбайт свободного пространства на внутреннем накопителе;
* экран с разрешением не менее 320х480 пикселей;
* сенсорная панель или другое устройство точечного ввода;
* камера с разрешением не менее 2 мегапикселей;
* соединение с сетью интернет со скоростью обмена данными не менее 512 кбит/с.

**Структура программного продукта**

Исходный код программного продукта состоит из двух основных типов файлов: Java-классов с расширением \*.java и файлов разметки языка XML с расширением \*.xml. Помимо этих типов, существуют также несколько вспомогательных типов: манифест приложения, различные графические файлы и файлы настроек сборщика Gradle.

Классы приложения располагаются в папке «main/java/xyz/tumist/diploma/».

Файлы разметки и графические файлы располагаются в папке «main/res».

Манифест приложения и файлы настроек Gradle располагаются в папке «main».

**Модули программного продукта**

Программный продукт имеет четыре основных типа модулей:

* модули работы с базой данных;
* модули получения и обработки информации;
* модули вывода информации;
* вспомогательные модули.

Модули работы с базой данных представлены тремя основными классами:

* DataContract, - класс, в котором описаны основные константы базы данных: названия таблиц, названия полей, типов данных;
* ReceiptsDBHelper – класс, который используется для создания базы данных;
* ReceiptsProvider – класс, который содержит методы для парсинга Uri-объектов при запросах к базе данных, а также полную реализацию CRUD-методов.

Модули обработки информации:

* downloadJson – класс, получающий параметры от класса QRScanner, скачивающий JSON-файл фискального чека с сервера Федеральной налоговой службы и передающий полученный файл в класс jsonParser;
* jsonParser – класс, обрабатывающий данные из JSON-файла фискального чека и добавляющий их в базу данных;
* QRScanner – класс-активность, позволяющий сканировать QR-код с фискального чека и вызвать методы класса downloadJson, куда передаются полученные параметры.

Модули вывода информации:

* MainActivity – основная активность, запускается при открытии приложения. Наполняет фрагменты с помощью вызова соответствующих классов-фрагментов.
* PointActivity – активность, которая выводит данные о торговой точке;
* StoreActivity – активность, которая выводит данные о магазине;
* ItemActivity – активность, которая выводит данные о товаре;
* PurchaseActivity – активность, которая выводит данные о покупке;
* ItemsFragment – класс-фрагмент, который выводит список товаров;
* PurchasesFragment – класс-фрагмент, который выводит список покупок;
* StoresFragment – класс-фрагмент, который выводит список магазинов.

Вспомогательные модули:

* ItemCursorAdapter – класс-адаптер, который наполняет данными элементы списка товаров;
* PointCursorAdapter – класс-адаптер, который наполняет данными элементы списка точек продаж;
* PurchaseCursorAdapter – класс-адаптер, который наполняет данными элементы списка покупок;
* PurchasePointCursorAdapter – класс-адаптер, который наполняет данными элементы списка точек продаж для покупки;
* StoreCursorAdapter – класс-адаптер, который наполняет данными элементы списка магазинов.

## Приложение Г Исходный текст программного изделия

Класс DataContract.java:

package xyz.tumist.diploma.data;

import android.content.ContentResolver;

import android.net.Uri;

import android.provider.BaseColumns;

public class DataContract {

public static final String PATH\_GOODS = "Goods";

public static final String PATH\_PURCHASES = "Purchases";

public static final String PATH\_POINTS = "Points";

public static final String PATH\_STORES = "Stores";

//Класс, определяющий содержимое таблицы с товарами

public static abstract class GoodEntry implements BaseColumns {

public static final Uri CONTENT\_URI = Uri.withAppendedPath(BASE\_CONTENT\_URI, PATH\_GOODS);

public static final String TABLE\_NAME = "Goods"; //название таблицы

public static final String COLUMN\_GOOD\_ID = "\_id";

public static final String COLUMN\_GOOD\_MODIFIERS = "good\_modifiers";

public static final String COLUMN\_GOOD\_NAME = "good\_name";

public static final String COLUMN\_GOOD\_NDS0 = "good\_nds0";

public static final String COLUMN\_GOOD\_NDS10 = "good\_nds10";

public static final String COLUMN\_GOOD\_NDS18 = "good\_nds18";

public static final String COLUMN\_GOOD\_NDSCALCULATED10 = "good\_ndsCalculated10";

public static final String COLUMN\_GOOD\_NDSCALCULATED18 = "good\_ndsCalculated18";

public static final String COLUMN\_GOOD\_NDSNO = "good\_ndsNo";

public static final String COLUMN\_GOOD\_PRICE = "good\_price";

public static final String COLUMN\_GOOD\_QUANTITY = "good\_quantity";

public static final String COLUMN\_GOOD\_SUM = "good\_sum";

public static final String COLUMN\_GOOD\_STORNO = "good\_storno";

public static final String COLUMN\_GOOD\_PURCHASE\_ID\_FK = "purchase\_id\_FK";

}

public static abstract class PurchaseEntry implements BaseColumns{

public static final Uri CONTENT\_URI = Uri.withAppendedPath(BASE\_CONTENT\_URI, PATH\_PURCHASES);

public static final String TABLE\_NAME = "Purchases"; //название таблицы

//Данные из фискального чека

public static final String COLUMN\_PURCHASE\_ID = "\_id";

public static final String COLUMN\_PURCHASE\_FISCALDOCUMENTNUMBER = "purchase\_fiscalDocumentNumber";

public static final String COLUMN\_PURCHASE\_FISCALDRIVENUMBER = "purchase\_fiscalDriveNumber";

public static final String COLUMN\_PURCHASE\_CASHTOTALSUM = "purchase\_cashTotalSum";

public static final String COLUMN\_PURCHASE\_DATETIME = "purchase\_dateTime";

public static final String COLUMN\_PURCHASE\_DISCOUNT = "purchase\_discount";

public static final String COLUMN\_PURCHASE\_DISCOUNTSUM = "purchase\_discount\_sum";

public static final String COLUMN\_PURCHASE\_ECASHTOTALSUM = "purchase\_ecashTotalSum";

public static final String COLUMN\_PURCHASE\_TOTALSUM = "purchase\_totalSum";

public static final String COLUMN\_PURCHASE\_FISCALSIGN = "purchase\_fiscalSign";

public static final String COLUMN\_PURCHASE\_ITEMSAMOUNT = "purchase\_itemsAmount";

public static final String COLUMN\_PURCHASE\_OPERATOR = "purchase\_operator";

public static final String COLUMN\_PURCHASE\_REQUESTNUMBER = "purchase\_requestNumber";

public static final String COLUMN\_PURCHASE\_SHIFTNUMBER = "purchase\_shiftNumber";

public static final String COLUMN\_PURCHASE\_KKTREGID = "purchase\_kktRegID";

public static final String COLUMN\_PURCHASE\_KKTNUMBER = "purchase\_kktNumber";

public static final String COLUMN\_PURCHASE\_NDS0 = "purchase\_nds0";

public static final String COLUMN\_PURCHASE\_NDS10 = "purchase\_nds10";

public static final String COLUMN\_PURCHASE\_NDS18 = "purchase\_nds18";

public static final String COLUMN\_PURCHASE\_NDSCALCULATED10 = "purchase\_ndsCalculated10";

public static final String COLUMN\_PURCHASE\_NDSCALCULATED18 = "purchase\_ndsCalculated18";

public static final String COLUMN\_PURCHASE\_NDSNO = "purchase\_ndsNo";

public static final String COLUMN\_PURCHASE\_TAXATIONTYPE = "purchase\_taxationType";

public static final String COLUMN\_PURCHASE\_OPERATIONTYPE = "purchase\_operationType";

public static final String COLUMN\_PURCHASE\_POINT\_ID\_FK = "point\_id\_FK";

public static final String COLUMN\_PURCHASE\_STORE\_ID\_FK = "store\_id\_FK";

//Дополнительные данные

public static final String COLUMN\_PURCHASE\_JSONNAME = "purchase\_jsonName";

public static final String COLUMN\_PURCHASE\_TAGS = "purchase\_tags";

}

public static abstract class PointEntry implements BaseColumns{

public static final Uri CONTENT\_URI = Uri.withAppendedPath(BASE\_CONTENT\_URI, PATH\_POINTS);

public static final String TABLE\_NAME = "Points"; //название таблицы

public static final String COLUMN\_POINT\_ID = "\_id";

public static final String COLUMN\_POINT\_ADDRESS = "point\_address";

public static final String COLUMN\_POINT\_NICKNAME = "point\_nickname";

public static final String COLUMN\_STORE\_ID\_FK = "store\_id\_FK";

}

public static abstract class StoreEntry implements BaseColumns {

public static final Uri CONTENT\_URI = Uri.withAppendedPath(BASE\_CONTENT\_URI, PATH\_STORES);

public static final String TABLE\_NAME = "Stores"; //название таблицы

public static final String COLUMN\_STORE\_ID = "\_id";

public static final String COLUMN\_STORE\_NAME = "store\_name";

public static final String COLUMN\_STORE\_INN = "store\_inn";

public static final String COLUMN\_STORE\_NICKNAME = "store\_nickname";

}

public static final String CONTENT\_AUTHORITY = "xyz.tumist.diploma";

public static final Uri BASE\_CONTENT\_URI = Uri.parse("content://" + CONTENT\_AUTHORITY);

//Я даже не знаю, юзается ли это

/\*\*

\* The MIME type of the {@link #CONTENT\_URI} for a list of pets.

\*/

public static final String CONTENT\_LIST\_TYPE =

ContentResolver.CURSOR\_DIR\_BASE\_TYPE + "/" + CONTENT\_AUTHORITY + "/" + PATH\_PURCHASES;

public static final String CONTENT\_ITEM\_TYPE =

ContentResolver.CURSOR\_ITEM\_BASE\_TYPE + "/" + CONTENT\_AUTHORITY + "/" + PATH\_PURCHASES;

}

Класс ReceiptsDBHelper.java:

package xyz.tumist.diploma.data;

import android.content.Context;

import android.database.DatabaseUtils;

import android.database.sqlite.SQLiteDatabase;

import android.database.sqlite.SQLiteOpenHelper;

import android.util.Log;

public class ReceiptsDBHelper extends SQLiteOpenHelper {

//Тег для логов

public static final String LOG\_TAG = ReceiptsDBHelper.class.getSimpleName();

//Название базы данных

private static final String DATABASE\_NAME = "checks.db";

//Версия базы данных. При изменении схемы БД нужно увеличивать версию БД на 1.

private static final int DATABASE\_VERSION = 1;

//Конструктор класса

public ReceiptsDBHelper(Context context){

super(context, DATABASE\_NAME, null, DATABASE\_VERSION);

}

//Используется при первом создании базы данных

@Override

public void onCreate(SQLiteDatabase db){

//Строки с SQL-командой для создания таблиц persons, debts и transactions

String SQL\_CREATE\_GOODS\_TABLE = "CREATE TABLE " + DataContract.GoodEntry.TABLE\_NAME + " (" +

DataContract.GoodEntry.COLUMN\_GOOD\_ID + " INTEGER PRIMARY KEY AUTOINCREMENT, " +

DataContract.GoodEntry.COLUMN\_GOOD\_MODIFIERS + " TEXT, " +

DataContract.GoodEntry.COLUMN\_GOOD\_NAME + " TEXT NOT NULL," +

DataContract.GoodEntry.COLUMN\_GOOD\_NDS0 + " INTEGER, " +

DataContract.GoodEntry.COLUMN\_GOOD\_QUANTITY + " INTEGER, " +

DataContract.GoodEntry.COLUMN\_GOOD\_SUM + " INTEGER NOT NULL, " +

DataContract.GoodEntry.COLUMN\_GOOD\_NDS10 + " INTEGER, " +

DataContract.GoodEntry.COLUMN\_GOOD\_NDS18 + " INTEGER, " +

DataContract.GoodEntry.COLUMN\_GOOD\_NDSCALCULATED10 + " INTEGER, " +

DataContract.GoodEntry.COLUMN\_GOOD\_NDSCALCULATED18 + " INTEGER, " +

DataContract.GoodEntry.COLUMN\_GOOD\_NDSNO + " INTEGER, " +

DataContract.GoodEntry.COLUMN\_GOOD\_PRICE + " INTEGER NOT NULL, " +

DataContract.GoodEntry.COLUMN\_GOOD\_STORNO + " TEXT," +

DataContract.GoodEntry.COLUMN\_GOOD\_PURCHASE\_ID\_FK + " INTEGER, " +

//Внешний ключ

"FOREIGN KEY(" + DataContract.GoodEntry.COLUMN\_GOOD\_PURCHASE\_ID\_FK + ") REFERENCES " +

DataContract.PurchaseEntry.TABLE\_NAME + "(" + DataContract.PurchaseEntry.COLUMN\_PURCHASE\_ID + ")" +

");";

String SQL\_CREATE\_PURCHASES\_TABLE = "CREATE TABLE " + DataContract.PurchaseEntry.TABLE\_NAME + " (" +

DataContract.PurchaseEntry.COLUMN\_PURCHASE\_ID + " INTEGER PRIMARY KEY AUTOINCREMENT, " +

DataContract.PurchaseEntry.COLUMN\_PURCHASE\_FISCALDOCUMENTNUMBER + " TEXT NOT NULL, " +

DataContract.PurchaseEntry.COLUMN\_PURCHASE\_FISCALDRIVENUMBER + " TEXT NOT NULL, " +

DataContract.PurchaseEntry.COLUMN\_PURCHASE\_CASHTOTALSUM + " INTEGER," +

DataContract.PurchaseEntry.COLUMN\_PURCHASE\_ECASHTOTALSUM + " INTEGER, " +

DataContract.PurchaseEntry.COLUMN\_PURCHASE\_TOTALSUM + " INTEGER NOT NULL, " +

DataContract.PurchaseEntry.COLUMN\_PURCHASE\_DATETIME + " INTEGER NOT NULL, " +

DataContract.PurchaseEntry.COLUMN\_PURCHASE\_DISCOUNT + " INTEGER, " +

DataContract.PurchaseEntry.COLUMN\_PURCHASE\_DISCOUNTSUM + " INTEGER, " +

DataContract.PurchaseEntry.COLUMN\_PURCHASE\_FISCALSIGN + " INTEGER, " +

DataContract.PurchaseEntry.COLUMN\_PURCHASE\_ITEMSAMOUNT + " INTEGER, " +

DataContract.PurchaseEntry.COLUMN\_PURCHASE\_OPERATOR + " TEXT, " +

DataContract.PurchaseEntry.COLUMN\_PURCHASE\_REQUESTNUMBER + " INTEGER, " +

DataContract.PurchaseEntry.COLUMN\_PURCHASE\_SHIFTNUMBER + " INTEGER, " +

DataContract.PurchaseEntry.COLUMN\_PURCHASE\_KKTREGID + " TEXT, " +

DataContract.PurchaseEntry.COLUMN\_PURCHASE\_KKTNUMBER + " TEXT, " +

DataContract.PurchaseEntry.COLUMN\_PURCHASE\_NDS0 + " INTEGER, " +

DataContract.PurchaseEntry.COLUMN\_PURCHASE\_NDS10 + " INTEGER, " +

DataContract.PurchaseEntry.COLUMN\_PURCHASE\_NDS18 + " INTEGER, " +

DataContract.PurchaseEntry.COLUMN\_PURCHASE\_NDSCALCULATED10 + " INTEGER, " +

DataContract.PurchaseEntry.COLUMN\_PURCHASE\_NDSCALCULATED18 + " INTEGER, " +

DataContract.PurchaseEntry.COLUMN\_PURCHASE\_NDSNO + " INTEGER, " +

DataContract.PurchaseEntry.COLUMN\_PURCHASE\_TAXATIONTYPE + " INTEGER, " +

DataContract.PurchaseEntry.COLUMN\_PURCHASE\_OPERATIONTYPE + " INTEGER, " +

DataContract.PurchaseEntry.COLUMN\_PURCHASE\_POINT\_ID\_FK + " INTEGER, " +

DataContract.PurchaseEntry.COLUMN\_PURCHASE\_STORE\_ID\_FK + " INTEGER, " +

//Внешний ключ к точкам и теги

"FOREIGN KEY(" + DataContract.PurchaseEntry.COLUMN\_PURCHASE\_POINT\_ID\_FK + ") REFERENCES " +

DataContract.PointEntry.TABLE\_NAME + "(" + DataContract.PointEntry.COLUMN\_POINT\_ID + ")," +

"FOREIGN KEY(" + DataContract.PurchaseEntry.COLUMN\_PURCHASE\_STORE\_ID\_FK + ") REFERENCES " +

DataContract.StoreEntry.TABLE\_NAME + "(" + DataContract.StoreEntry.COLUMN\_STORE\_ID + ")" +

");";

String SQL\_CREATE\_POINTS\_TABLE = "CREATE TABLE " + DataContract.PointEntry.TABLE\_NAME + " (" +

DataContract.PointEntry.COLUMN\_POINT\_ID + " INTEGER PRIMARY KEY AUTOINCREMENT, " +

DataContract.PointEntry.COLUMN\_POINT\_ADDRESS + " TEXT, " +

DataContract.PointEntry.COLUMN\_POINT\_NICKNAME + " TEXT, " +

DataContract.PointEntry.COLUMN\_STORE\_ID\_FK + " INTEGER, " +

"FOREIGN KEY(" + DataContract.PointEntry.COLUMN\_STORE\_ID\_FK + ") REFERENCES " +

DataContract.StoreEntry.TABLE\_NAME + "(" + DataContract.StoreEntry.COLUMN\_STORE\_ID + ")" +

");";

String SQL\_CREATE\_STORES\_TABLE = "CREATE TABLE " + DataContract.StoreEntry.TABLE\_NAME + " (" +

DataContract.StoreEntry.COLUMN\_STORE\_ID + " INTEGER PRIMARY KEY AUTOINCREMENT, " +

DataContract.StoreEntry.COLUMN\_STORE\_INN + " INTEGER NOT NULL, " +

DataContract.StoreEntry.COLUMN\_STORE\_NAME + " TEXT , " +

DataContract.StoreEntry.COLUMN\_STORE\_NICKNAME + " TEXT" +

");";

Log.v(LOG\_TAG, SQL\_CREATE\_GOODS\_TABLE);

Log.v(LOG\_TAG, SQL\_CREATE\_PURCHASES\_TABLE);

Log.v(LOG\_TAG, SQL\_CREATE\_POINTS\_TABLE);

Log.v(LOG\_TAG, SQL\_CREATE\_STORES\_TABLE);

db.execSQL(SQL\_CREATE\_STORES\_TABLE);

db.execSQL(SQL\_CREATE\_POINTS\_TABLE);

db.execSQL(SQL\_CREATE\_PURCHASES\_TABLE);

db.execSQL(SQL\_CREATE\_GOODS\_TABLE);

}

@Override

public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {

// The database is still at version 1, so there's nothing to do be done here.

}

public long getPurchasesCount(String purchaseSelection, String[] purchaseSelectionArgs) {

SQLiteDatabase db = this.getReadableDatabase();

long count = DatabaseUtils.queryNumEntries(

db,

DataContract.PurchaseEntry.TABLE\_NAME,

purchaseSelection,

purchaseSelectionArgs);

db.close();

return count;

}

public long getItemsCount(String itemSelection, String[] itemSelectionArgs) {

SQLiteDatabase db = this.getReadableDatabase();

long count = DatabaseUtils.queryNumEntries(

db,

DataContract.GoodEntry.TABLE\_NAME,

itemSelection,

itemSelectionArgs);

db.close();

return count;

}

public long getPointsCount(String pointSelection, String[] pointSelectionArgs) {

SQLiteDatabase db = this.getReadableDatabase();

long count = DatabaseUtils.queryNumEntries(

db,

DataContract.PointEntry.TABLE\_NAME,

pointSelection,

pointSelectionArgs);

db.close();

return count;

}

}

Класс MainActivity.java:

package xyz.tumist.diploma;

import android.Manifest;

import android.content.Intent;

import android.content.pm.PackageManager;

import android.os.Build;

import android.os.Bundle;

import android.support.annotation.NonNull;

import android.support.design.widget.BottomNavigationView;

import android.support.design.widget.FloatingActionButton;

import android.support.v4.app.ActivityCompat;

import android.support.v4.app.Fragment;

import android.support.v4.app.FragmentManager;

import android.support.v4.app.FragmentTransaction;

import android.support.v7.app.AppCompatActivity;

import android.util.Log;

import android.view.Menu;

import android.view.MenuItem;

import android.view.View;

import xyz.tumist.diploma.main\_page.ItemsFragment;

import xyz.tumist.diploma.main\_page.PurchasesFragment;

import xyz.tumist.diploma.main\_page.StoresFragment;

public class MainActivity extends AppCompatActivity {

private static final String LOG\_TAG = MainActivity.class.getSimpleName();

private BottomNavigationView navigation;

private Fragment fragment;

private FragmentManager fragmentManager;

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

FloatingActionButton fab = findViewById(R.id.fab);

fab.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

Intent intent = new Intent(getApplicationContext(), QRScanner.class);

startActivity(intent);

}

});

checkCameraPermissions();

checkStorageReadPermission();

checkStorageWritePermission();

navigation = (BottomNavigationView)findViewById(R.id.navigation);

// inflatemenu убивает приложение, хммм. Но я и через XML его забил.

// bottomNavigation.inflateMenu(R.menu.navigation);

fragmentManager = getSupportFragmentManager();

//Поставить по умолчанию страницу с долгами мне, чтобы сразу иинициализировать фрагмент, ну и потому что хочу этот экран стартовым

navigation.setSelectedItemId(R.id.navigation\_purchases); //выбор в bottomnavigationview

// fragment = new PurchasesFragment(); //задание фрагмента

// final FragmentTransaction transaction = fragmentManager.beginTransaction();

// transaction.replace(R.id.main\_container, fragment).commit(); //установка фрагмента

// // Begin the transaction

FragmentTransaction ft = getSupportFragmentManager().beginTransaction();

// Replace the contents of the container with the new fragment

ft.replace(R.id.main\_container, new PurchasesFragment());

// or ft.add(R.id.your\_placeholder, new FooFragment());

// Complete the changes added above

ft.commit();

navigation.setOnNavigationItemSelectedListener(new BottomNavigationView.OnNavigationItemSelectedListener() {

@Override

public boolean onNavigationItemSelected(@NonNull MenuItem item) {

int id = item.getItemId();

switch (id){

case R.id.navigation\_purchases:

fragment = new PurchasesFragment();

Log.v(LOG\_TAG, "Тыкнуты покупки в навигации");

break;

case R.id.navigation\_items:

fragment = new ItemsFragment();

break;

case R.id.navigation\_stores:

fragment = new StoresFragment();

break;

}

final FragmentTransaction transaction = fragmentManager.beginTransaction();

transaction.replace(R.id.main\_container, fragment).commit();

return true;

}

});

private void checkCameraPermissions() {

final int MY\_CAMERA\_REQUEST\_CODE = 100;

if (Build.VERSION.SDK\_INT >= 23) {

if (checkSelfPermission(Manifest.permission.CAMERA)

!= PackageManager.PERMISSION\_GRANTED) {

requestPermissions(new String[]{Manifest.permission.CAMERA},

MY\_CAMERA\_REQUEST\_CODE);

}

}

else { //permission is automatically granted on sdk<23 upon installation

Log.v(LOG\_TAG,"Permission is granted");

}

}

public boolean checkStorageReadPermission() {

if (Build.VERSION.SDK\_INT >= 23) {

if (checkSelfPermission(android.Manifest.permission.READ\_EXTERNAL\_STORAGE)

== PackageManager.PERMISSION\_GRANTED) {

Log.v(LOG\_TAG,"Permission is granted");

return true;

} else {

Log.v(LOG\_TAG,"Permission is revoked");

ActivityCompat.requestPermissions(this, new String[]{Manifest.permission.READ\_EXTERNAL\_STORAGE}, 1);

return false;

}

}

else { //permission is automatically granted on sdk<23 upon installation

Log.v(LOG\_TAG,"Permission is granted");

return true;

}

}

public boolean checkStorageWritePermission() {

if (Build.VERSION.SDK\_INT >= 23) {

if (checkSelfPermission(android.Manifest.permission.WRITE\_EXTERNAL\_STORAGE)

== PackageManager.PERMISSION\_GRANTED) {

Log.v(LOG\_TAG,"Permission is granted2");

return true;

} else {

Log.v(LOG\_TAG,"Permission is revoked2");

ActivityCompat.requestPermissions(this, new String[]{Manifest.permission.WRITE\_EXTERNAL\_STORAGE}, 2);

return false;

}

}

else { //permission is automatically granted on sdk<23 upon installation

Log.v(LOG\_TAG,"Permission is granted2");

return true;

}

}

public boolean onCreateOptionsMenu(Menu menu) {

// Inflate the menu; this adds items to the action bar if it is present.

getMenuInflater().inflate(R.menu.menu\_main, menu);

return true;

}

@Override

public boolean onOptionsItemSelected(MenuItem item) {

// Handle action bar item clicks here. The action bar will

// automatically handle clicks on the Home/Up button, so long

// as you specify a parent activity in AndroidManifest.xml.

int id = item.getItemId();

//noinspection SimplifiableIfStatement

// if (id == R.id.action\_settings) {

// return true;

// }

if (id == R.id.action\_help) {

Intent intent = new Intent(this, WebViewActivity.class);

startActivity(intent);

return true;

}

return super.onOptionsItemSelected(item);

}

}

Класс ItemsFragment.java:

public class ItemsFragment extends Fragment {

@Override

public View onCreateView(LayoutInflater inflater, ViewGroup container,

Bundle savedInstanceState) {

// Inflate the layout for this fragment

return inflater.inflate(R.layout.fragment\_list, container, false);

}

public void onViewCreated(View view, Bundle savedInstanceState) {

// Setup any handles to view objects here

// TodoDatabaseHandler is a SQLiteOpenHelper class connecting to SQLite

ReceiptsDBHelper handler = new ReceiptsDBHelper(getContext());

// Get access to the underlying writeable database

SQLiteDatabase db = handler.getWritableDatabase();

// Query for items from the database and get a cursor back

Cursor itemCursor = db.rawQuery("SELECT " +

DataContract.GoodEntry.TABLE\_NAME+"."+ DataContract.GoodEntry.COLUMN\_GOOD\_ID+", " +

DataContract.GoodEntry.COLUMN\_GOOD\_NAME + ", " +

DataContract.GoodEntry.COLUMN\_GOOD\_SUM + ", " +

DataContract.GoodEntry.COLUMN\_GOOD\_PRICE + ", " +

DataContract.GoodEntry.COLUMN\_GOOD\_QUANTITY + ", " +

DataContract.GoodEntry.COLUMN\_GOOD\_NDS10 + ", " +

DataContract.GoodEntry.COLUMN\_GOOD\_NDS18 + ", " +

DataContract.GoodEntry.COLUMN\_GOOD\_NDSNO + ", " +

DataContract.GoodEntry.COLUMN\_GOOD\_NDS0 + ", " +

DataContract.GoodEntry.COLUMN\_GOOD\_STORNO + ", " +

DataContract.GoodEntry.COLUMN\_GOOD\_PURCHASE\_ID\_FK + ", " +

DataContract.PurchaseEntry.COLUMN\_PURCHASE\_DATETIME + " " +

"FROM " + DataContract.GoodEntry.TABLE\_NAME + " INNER JOIN " + DataContract.PurchaseEntry.TABLE\_NAME +

" ON " + DataContract.GoodEntry.TABLE\_NAME + "." + DataContract.GoodEntry.COLUMN\_GOOD\_PURCHASE\_ID\_FK + " = " + DataContract.PurchaseEntry.TABLE\_NAME + "." + DataContract.PurchaseEntry.COLUMN\_PURCHASE\_ID +

" ORDER BY "+ DataContract.PurchaseEntry.COLUMN\_PURCHASE\_DATETIME + " DESC"

, null, null);

// Find ListView to populate

ListView lvItems = (ListView) view.findViewById(R.id.fragment\_listview);

// Setup cursor adapter using cursor from last step

ItemCursorAdapter itemAdapter = new ItemCursorAdapter(getContext(), itemCursor);

// Attach cursor adapter to the ListView

lvItems.setAdapter(itemAdapter);

lvItems.setOnItemClickListener(new AdapterView.OnItemClickListener() {

@Override

public void onItemClick(AdapterView<?> adapterView, View view, int position, long id) {

Intent intent = new Intent(getContext(), ItemActivity.class);

Uri currentDebtUri = ContentUris.withAppendedId(DataContract.GoodEntry.CONTENT\_URI, id);

// Set the URI on the data field of the intent

intent.setData(currentDebtUri);

//intent.putExtra("id", debtID);

startActivityForResult(intent, 10002);

}

});

}

@Override

public void onActivityResult(int requestCode, int resultCode, Intent data)

{

super.onActivityResult(requestCode, resultCode, data);

if (requestCode == 10002)

{

// recreate your fragment here

FragmentTransaction ft = getFragmentManager().beginTransaction();

ft.detach(this).attach(this).commit();

}

}

}

Класс PurchasesFragment.java:

public class PurchasesFragment extends Fragment {

@Override

public View onCreateView(LayoutInflater inflater, ViewGroup container,

Bundle savedInstanceState) {

// Inflate the layout for this fragment

return inflater.inflate(R.layout.fragment\_list, container, false);

}

public void onViewCreated(View view, Bundle savedInstanceState) {

// Setup any handles to view objects here

// TodoDatabaseHandler is a SQLiteOpenHelper class connecting to SQLite

ReceiptsDBHelper handler = new ReceiptsDBHelper(getContext());

// Get access to the underlying writeable database

SQLiteDatabase db = handler.getWritableDatabase();

// Query for items from the database and get a cursor back

Cursor purchaseCursor = db.rawQuery("SELECT \* FROM " + DataContract.PurchaseEntry.TABLE\_NAME + " ORDER BY "+ DataContract.PurchaseEntry.COLUMN\_PURCHASE\_DATETIME + " DESC"

,null);

// Find ListView to populate

ListView lvPurchases = (ListView) view.findViewById(R.id.fragment\_listview);

// Setup cursor adapter using cursor from last step

PurchaseCursorAdapter purchaseAdapter = new PurchaseCursorAdapter(getContext(), purchaseCursor);

// Attach cursor adapter to the ListView

lvPurchases.setAdapter(purchaseAdapter);

lvPurchases.setOnItemClickListener(new AdapterView.OnItemClickListener() {

@Override

public void onItemClick(AdapterView<?> adapterView, View view, int position, long id) {

Intent intent = new Intent(getContext(), PurchaseActivity.class);

Uri currentPurchaseUri = ContentUris.withAppendedId(DataContract.PurchaseEntry.CONTENT\_URI, id);

// Set the URI on the data field of the intent

intent.setData(currentPurchaseUri);

//intent.putExtra("id", debtID);

startActivityForResult(intent, 10001);

}

});

}

@Override

public void onActivityResult(int requestCode, int resultCode, Intent data)

{

super.onActivityResult(requestCode, resultCode, data);

if (requestCode == 10001)

{

// recreate your fragment here

FragmentTransaction ft = getFragmentManager().beginTransaction();

ft.detach(this).attach(this).commit();

}

}

}

Класс downloadJson.java:

public class downloadJson extends AsyncTask<String, String, String> {

Context mContext;

String mFN, mFD, mFPD;

public downloadJson (Context context, String FN, String FD, String FPD){

mContext = context.getApplicationContext();

mFN = FN;

mFD = FD;

mFPD = FPD;

}

HttpURLConnection urlConnection;

@Override

protected void onPreExecute() {

super.onPreExecute();

// ProgressDialog progDailog = new ProgressDialog(mContext);

// progDailog.setMessage("Загрузка чека");

// progDailog.setIndeterminate(false);

// progDailog.setProgressStyle(ProgressDialog.STYLE\_SPINNER);

// progDailog.setCancelable(true);

// progDailog.show();

}

@Override

protected String doInBackground(String... args) {

StringBuilder result = new StringBuilder();

try {

String fullURL = "http://proverkacheka.nalog.ru:8888/v1/inns/\*/kkts/\*/fss/" + mFN + "/tickets/" + mFD + "?fiscalSign=" + mFPD + "&sendToEmail=no";

URL url = new URL(fullURL);

urlConnection = (HttpURLConnection) url.openConnection();

urlConnection.setRequestProperty ("Authorization", "Basic Kzc5OTk0NzYyNjc2OjcyNDEzMQ==");

urlConnection.setRequestProperty ("Device-Id", "0008cccc-eeeeeeee-vvvvvvvv-gggggggg");

urlConnection.setRequestProperty ("Device-OS", "Android 5.1");

urlConnection.setRequestMethod("GET");

InputStream in = new BufferedInputStream(urlConnection.getInputStream());

BufferedReader reader = new BufferedReader(new InputStreamReader(in));

String line;

while ((line = reader.readLine()) != null) {

result.append(line);

}

}catch( Exception e) {

e.printStackTrace();

}

finally {

urlConnection.disconnect();

}

return result.toString();

}

@Override

protected void onPostExecute(String result) {

try {

Log.v("downloadJSON", result.toString());

StringBuilder sb = new StringBuilder(result.toString());

sb.delete(0, 23);

sb.delete(sb.length() - 2, sb.length());

String trimmedResult = sb.toString();

String timeHuman = trimmedResult.substring(trimmedResult.indexOf("Time\":\"") + 7, trimmedResult.indexOf("Time\":\"") + 26);

Log.v("downloadJSON", trimmedResult);

Long epochSeconds = 0L;

try {

epochSeconds = new SimpleDateFormat("yyyy-MM-dd'T'HH:mm:ss").parse(timeHuman).getTime() / 1000L + 25200L;

} catch (ParseException e) {

e.printStackTrace();

}

String finalResult = trimmedResult.replaceAll(timeHuman, String.valueOf(epochSeconds));

Intent i = new Intent(mContext, jsonParser.class);

i.putExtra("jsonString", "true");

i.putExtra("result", finalResult);

mContext.startActivity(i);

}

catch (Exception e)

{

Toast toast = Toast.makeText(mContext.getApplicationContext(), "Ошибка получения чека с сервера", Toast.LENGTH\_SHORT);

toast.show();

}

}

}

Класс jsonParser.java;

public class jsonParser extends AppCompatActivity {

Purchase purchase;

Gson gson = new Gson();

public static final String LOG\_TAG = jsonParser.class.getSimpleName();

Uri receivedUri;

public Long StoreID;

public Long PointID;

public Long PurchaseID;

private ReceiptsDBHelper mDbHelper;

public String receiptJSON;

//public Intent intent;

@Override

protected void onCreate(Bundle savedInstanceState) {

mDbHelper = new ReceiptsDBHelper(getApplicationContext());

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_json\_parser);

// Get intent, action and MIME type

Intent intent = getIntent();

String action = intent.getAction();

String type = intent.getType();

//Проверям, есть ли права на хранилище

isStoragePermissionGranted();

// String jsonString = intent.getStringExtra("jsonString");

// Log.v("Тип ", type);

// Log.v(LOG\_TAG, intent.getStringExtra("result"));

if (Intent.ACTION\_SEND.equals(action) && type != null) {

//получаем сслыку на файл в URI

receivedUri = (Uri) intent.getParcelableExtra(Intent.EXTRA\_STREAM);

try {

//Открываем буферное чтение

BufferedReader bufferedReader = new BufferedReader(new FileReader(receivedUri.getPath()));

//Переводим JSON-файл в GSON-объект, ориентируясь на класс

purchase = gson.fromJson(bufferedReader, Purchase.class);

Log.v(LOG\_TAG, "Активирован считывающий Штайнер");

Log.v(LOG\_TAG, purchase.operator);

Log.v(LOG\_TAG, purchase.items[0].name);

Log.v(LOG\_TAG, String.valueOf(purchase.nds0));

//Вызываем методы по порядку

storeCheck();

pointCheck();

purchaseCheck();

//itemsAdd();

//jsonReader.close();

} catch (IOException e) {

Toast.makeText(getApplicationContext(), "Невозможно открыть файл", Toast.LENGTH\_SHORT).show();

}

}

else try {

if (intent.getStringExtra("jsonString").equals("true"))

{

purchase = gson.fromJson(intent.getStringExtra("result"), Purchase.class);

//Вызываем методы по порядку

storeCheck();

pointCheck();

purchaseCheck();

}

} catch (Exception e)

{

}

}

public boolean isStoragePermissionGranted() {

if (Build.VERSION.SDK\_INT >= 23) {

if (checkSelfPermission(android.Manifest.permission.READ\_EXTERNAL\_STORAGE)

== PackageManager.PERMISSION\_GRANTED) {

Log.v(LOG\_TAG,"Permission is granted");

return true;

} else {

Log.v(LOG\_TAG,"Permission is revoked");

ActivityCompat.requestPermissions(this, new String[]{Manifest.permission.READ\_EXTERNAL\_STORAGE}, 1);

return false;

}

}

else { //permission is automatically granted on sdk<23 upon installation

Log.v(LOG\_TAG,"Permission is granted");

return true;

}

}

public void storeCheck() {

//Проверка на наличие в БД магазина из GSON-объекта.

//Если магазина нет, то он добавляется в БД

Cursor cursor;

SQLiteDatabase database = mDbHelper.getReadableDatabase();

String[] projection = {

DataContract.StoreEntry.COLUMN\_STORE\_ID,

DataContract.StoreEntry.COLUMN\_STORE\_NAME

};

String selection = DataContract.StoreEntry.COLUMN\_STORE\_INN + " LIKE ?";

String[] selectionArgs = {String.valueOf(purchase.userInn)};

cursor = database.query(

DataContract.StoreEntry.TABLE\_NAME, // Таблица, к которой обращён запрос

projection, // Колонки, которые требуется вернуть

selection, // Колонки для WHERE

selectionArgs, // Значения для WHERE

null, // Не группировать ряды

null, // Не фильтровать сгруппированные ряды

null); // Порядок сортировки

if (cursor.getCount() == 0) {

SQLiteDatabase dbWrite = mDbHelper.getWritableDatabase();

ContentValues storeValues = new ContentValues();

storeValues.put(DataContract.StoreEntry.COLUMN\_STORE\_INN, purchase.userInn);

if (purchase.user != null) {

storeValues.put(DataContract.StoreEntry.COLUMN\_STORE\_NAME, purchase.user);

}

Uri insertedDebtUri = getContentResolver().insert(DataContract.StoreEntry.CONTENT\_URI, storeValues);

StoreID = ContentUris.parseId(insertedDebtUri);

} else {

cursor.moveToFirst();

int storeIdColumnIndex = cursor.getColumnIndex(DataContract.StoreEntry.COLUMN\_STORE\_ID);

StoreID = cursor.getLong(storeIdColumnIndex);

if (cursor.getString(cursor.getColumnIndex(DataContract.StoreEntry.COLUMN\_STORE\_NAME)) == null) {

if (purchase.user != null){

ContentValues storeValues = new ContentValues();

storeValues.put(DataContract.StoreEntry.COLUMN\_STORE\_NAME, purchase.user);

int numOfUpdatedRows = getContentResolver().update(DataContract.StoreEntry.CONTENT\_URI, storeValues,

DataContract.StoreEntry.COLUMN\_STORE\_ID + " LIKE " + StoreID, null);

}

}

}

cursor.close();

}

public void pointCheck() {

//Проверка на наличие в БД точки продаж из GSON-объекта.

//Если точки нет, то она добавляется в БД

if (purchase.retailPlaceAddress == null){

//Если нет адреса точки, то она добавляется без адреса

long noAddressPointCount = mDbHelper.getPointsCount(

DataContract.PointEntry.COLUMN\_POINT\_ADDRESS + " IS NULL AND " + DataContract.PointEntry.COLUMN\_STORE\_ID\_FK + " LIKE " + StoreID,

null);

if (noAddressPointCount != 0) return;

}

Cursor cursor;

SQLiteDatabase database = mDbHelper.getReadableDatabase();

String[] projection = {

DataContract.PointEntry.COLUMN\_POINT\_ID,

DataContract.PointEntry.COLUMN\_POINT\_ADDRESS,

DataContract.PointEntry.COLUMN\_STORE\_ID\_FK

};

String selection = DataContract.PointEntry.COLUMN\_POINT\_ADDRESS + " LIKE ?" + " AND " + DataContract.PointEntry.COLUMN\_STORE\_ID\_FK + " LIKE ?";

String[] selectionArgs = {String.valueOf(purchase.retailPlaceAddress), String.valueOf(StoreID)};

cursor = database.query(

DataContract.PointEntry.TABLE\_NAME, // Таблица, к которой обращён запрос

projection, // Колонки, которые требуется вернуть

selection, // Колонки для WHERE

selectionArgs, // Значения для WHERE

null, // Не группировать ряды

null, // Не фильтровать сгруппированные ряды

null); // Порядок сортировки

if (cursor.getCount() == 0){

SQLiteDatabase dbWrite = mDbHelper.getWritableDatabase();

ContentValues pointValues = new ContentValues();

//pointValues.put(DataContract.StoreEntry.COLUMN\_STORE\_INN, purchase.userInn);

if (purchase.retailPlaceAddress != null)

{

pointValues.put(DataContract.PointEntry.COLUMN\_POINT\_ADDRESS, purchase.retailPlaceAddress);

}

pointValues.put(DataContract.PointEntry.COLUMN\_STORE\_ID\_FK, StoreID);

Uri insertedDebtUri = getContentResolver().insert(DataContract.PointEntry.CONTENT\_URI, pointValues);

PointID = ContentUris.parseId(insertedDebtUri);

}

else {

cursor.moveToNext();

int pointIdColumnIndex = cursor.getColumnIndex(DataContract.PointEntry.COLUMN\_POINT\_ID);

PointID = cursor.getLong(pointIdColumnIndex);

}

cursor.close();

}

public void purchaseCheck() {

Cursor cursor;

SQLiteDatabase database = mDbHelper.getReadableDatabase();

String[] projection = {

DataContract.PurchaseEntry.COLUMN\_PURCHASE\_FISCALDOCUMENTNUMBER,

DataContract.PurchaseEntry.COLUMN\_PURCHASE\_FISCALDRIVENUMBER

};

String selection = DataContract.PurchaseEntry.COLUMN\_PURCHASE\_FISCALDOCUMENTNUMBER + " LIKE ?" + " AND " + DataContract.PurchaseEntry.COLUMN\_PURCHASE\_FISCALDRIVENUMBER + " LIKE ?";

String[] selectionArgs = {String.valueOf(purchase.fiscalDocumentNumber), String.valueOf(purchase.fiscalDriveNumber)};

cursor = database.query(

DataContract.PurchaseEntry.TABLE\_NAME, // Таблица, к которой обращён запрос

projection, // Колонки, которые требуется вернуть

selection, // Колонки для WHERE

selectionArgs, // Значения для WHERE

null, // Не группировать ряды

null, // Не фильтровать сгруппированные ряды

null); // Порядок сортировки

if (cursor.getCount() == 0){

SQLiteDatabase dbWrite = mDbHelper.getWritableDatabase();

ContentValues purchaseValues = new ContentValues();

purchaseValues.put(DataContract.PurchaseEntry.COLUMN\_PURCHASE\_TAXATIONTYPE, purchase.taxationType);

purchaseValues.put(DataContract.PurchaseEntry.COLUMN\_PURCHASE\_SHIFTNUMBER, purchase.shiftNumber);

purchaseValues.put(DataContract.PurchaseEntry.COLUMN\_PURCHASE\_REQUESTNUMBER,purchase.requestNumber);

purchaseValues.put(DataContract.PurchaseEntry.COLUMN\_PURCHASE\_DISCOUNTSUM, purchase.discountSum);

purchaseValues.put(DataContract.PurchaseEntry.COLUMN\_PURCHASE\_POINT\_ID\_FK, PointID);

purchaseValues.put(DataContract.PurchaseEntry.COLUMN\_PURCHASE\_STORE\_ID\_FK, StoreID);

purchaseValues.put(DataContract.PurchaseEntry.COLUMN\_PURCHASE\_OPERATOR, purchase.operator);

purchaseValues.put(DataContract.PurchaseEntry.COLUMN\_PURCHASE\_OPERATIONTYPE, purchase.operationType);

purchaseValues.put(DataContract.PurchaseEntry.COLUMN\_PURCHASE\_NDSNO, purchase.ndsNo);

purchaseValues.put(DataContract.PurchaseEntry.COLUMN\_PURCHASE\_NDSCALCULATED18, purchase.ndsCalculated18);

purchaseValues.put(DataContract.PurchaseEntry.COLUMN\_PURCHASE\_NDSCALCULATED10, purchase.ndsCalculated10);

purchaseValues.put(DataContract.PurchaseEntry.COLUMN\_PURCHASE\_NDS18, purchase.nds18);

purchaseValues.put(DataContract.PurchaseEntry.COLUMN\_PURCHASE\_NDS10, purchase.nds10);

purchaseValues.put(DataContract.PurchaseEntry.COLUMN\_PURCHASE\_NDS0, purchase.nds0);

purchaseValues.put(DataContract.PurchaseEntry.COLUMN\_PURCHASE\_KKTREGID, purchase.kktRegId);

purchaseValues.put(DataContract.PurchaseEntry.COLUMN\_PURCHASE\_FISCALSIGN, purchase.fiscalSign);

purchaseValues.put(DataContract.PurchaseEntry.COLUMN\_PURCHASE\_KKTNUMBER, purchase.kktNumber);

purchaseValues.put(DataContract.PurchaseEntry.COLUMN\_PURCHASE\_FISCALDRIVENUMBER, purchase.fiscalDriveNumber);

purchaseValues.put(DataContract.PurchaseEntry.COLUMN\_PURCHASE\_FISCALDOCUMENTNUMBER, purchase.fiscalDocumentNumber);

purchaseValues.put(DataContract.PurchaseEntry.COLUMN\_PURCHASE\_ECASHTOTALSUM, purchase.ecashTotalSum);

purchaseValues.put(DataContract.PurchaseEntry.COLUMN\_PURCHASE\_CASHTOTALSUM, purchase.cashTotalSum);

purchaseValues.put(DataContract.PurchaseEntry.COLUMN\_PURCHASE\_TOTALSUM, purchase.totalSum);

purchaseValues.put(DataContract.PurchaseEntry.COLUMN\_PURCHASE\_DISCOUNT, purchase.discount);

purchaseValues.put(DataContract.PurchaseEntry.COLUMN\_PURCHASE\_DATETIME, purchase.dateTime);

Uri insertedPurchaseUri = getContentResolver().insert(DataContract.PurchaseEntry.CONTENT\_URI, purchaseValues);

PurchaseID = ContentUris.parseId(insertedPurchaseUri);

cursor.close();

itemsAdd();

}

else {

// cursor.moveToNext();

// int purchaseIdColumnIndex = cursor.getColumnIndex(DataContract.PurchaseEntry.COLUMN\_PURCHASE\_ID);

// PurchaseID = cursor.getLong(purchaseIdColumnIndex);

Toast toast = Toast.makeText(getApplicationContext(), "Такой чек уже добавлен", Toast.LENGTH\_SHORT);

toast.show();

cursor.close();

kill\_activity();

}

}

public void itemsAdd() {

if (purchase.items.length == 0){

return;

}

//Cursor cursor;

for (int i = 0; i < purchase.items.length; i++ ){

ContentValues itemValues = new ContentValues();

itemValues.put(DataContract.GoodEntry.COLUMN\_GOOD\_NAME, purchase.items[i].name);

itemValues.put(DataContract.GoodEntry.COLUMN\_GOOD\_NDS10, purchase.items[i].nds10);

itemValues.put(DataContract.GoodEntry.COLUMN\_GOOD\_NDS18, purchase.items[i].nds18);

itemValues.put(DataContract.GoodEntry.COLUMN\_GOOD\_NDSCALCULATED10, purchase.items[i].ndsCalculated10);

itemValues.put(DataContract.GoodEntry.COLUMN\_GOOD\_NDSCALCULATED18, purchase.items[i].ndsCalculated18);

itemValues.put(DataContract.GoodEntry.COLUMN\_GOOD\_NDSNO, purchase.items[i].ndsNo);

//itemValues.put(DataContract.GoodEntry.COLUMN\_GOOD\_MODIFIERS, purchase.items[i].modifiers);

itemValues.put(DataContract.GoodEntry.COLUMN\_GOOD\_NDS0, purchase.items[i].nds0);

itemValues.put(DataContract.GoodEntry.COLUMN\_GOOD\_PRICE, purchase.items[i].price);

itemValues.put(DataContract.GoodEntry.COLUMN\_GOOD\_QUANTITY, purchase.items[i].quantity);

itemValues.put(DataContract.GoodEntry.COLUMN\_GOOD\_PURCHASE\_ID\_FK, PurchaseID);

// if (purchase.items[i].storno == "true"){

// itemValues.put(DataContract.GoodEntry.COLUMN\_GOOD\_STORNO, 1);

// }

// else

// {

// itemValues.put(DataContract.GoodEntry.COLUMN\_GOOD\_STORNO, 0);

// }

itemValues.put(DataContract.GoodEntry.COLUMN\_GOOD\_SUM, purchase.items[i].sum);

Uri insertedItemUri = getContentResolver().insert(DataContract.GoodEntry.CONTENT\_URI, itemValues);

}

Toast toast = Toast.makeText(getApplicationContext(), "Покупка добавлена", Toast.LENGTH\_SHORT);

toast.show();

Intent intent = new Intent(this, MainActivity.class);

startActivity(intent);

kill\_activity();

}

void kill\_activity()

{

finish();

}

/\*\* И в классе покупки, и в классе товара убрал несколько полей,

\* потому что они почти не используются, и где-то идут пустыми массивами, а где-то значением false.

\* Лень пока делать обработку этого, всё равно абсолютно не нужно.

\*/

public class Purchase {

public Long cashTotalSum;

public Long dateTime;

public Long discount;

public Long discountSum;

public Long ecashTotalSum;

public String fiscalDocumentNumber;

public String fiscalDriveNumber;

public String fiscalSign;

public String kktNumber;

public String kktRegId;

public String addressToCheckFiscalSign;

public String senderAddress;

public String rawData;

public Long nds0;

public Long nds10;

public Long nds18;

public Long ndsCalculated10;

public Long ndsCalculated18;

public Long ndsNo;

public int operationType;

public String operator;

public int requestNumber;

public String retailPlaceAddress;

public int shiftNumber;

//public Item[] stornoItems;

public int taxationType;

public Long totalSum;

public String user;

public String userInn;

public Item[] items;

}

static class Item {

//public String modifiers;

public String name;

public Long nds0;

public Long nds10;

public Long nds18;

public Long ndsCalculated10;

public Long ndsCalculated18;

public Long ndsNo;

public Long price;

public double quantity;

public Long sum;

//public String storno;

}

}

Класс QRScanner:

package xyz.tumist.diploma;

import android.app.Activity;

import android.support.annotation.NonNull;

import android.support.v7.app.AppCompatActivity;

import android.os.Bundle;

import android.util.Log;

import android.view.View;

import android.widget.Toast;

import com.budiyev.android.codescanner.CodeScanner;

import com.budiyev.android.codescanner.CodeScannerView;

import com.budiyev.android.codescanner.DecodeCallback;

import com.google.zxing.Result;

public class QRScanner extends AppCompatActivity {

private CodeScanner mCodeScanner;

final String LOG\_TAG = PurchaseActivity.class.getSimpleName();

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_qrscanner);

CodeScannerView scannerView = findViewById(R.id.scanner\_view);

final Activity activity = this;

mCodeScanner = new CodeScanner(this, scannerView);

mCodeScanner.setDecodeCallback(new DecodeCallback() {

@Override

public void onDecoded(@NonNull final Result result) {

activity.runOnUiThread(new Runnable() {

@Override

public void run() {

String scanResult = result.getText();

String FN = scanResult.substring(scanResult.indexOf("fn=") + 3, scanResult.indexOf("&i="));

String FD = scanResult.substring(scanResult.indexOf("i=") + 2, scanResult.indexOf("&fp="));

String FPD = scanResult.substring(scanResult.indexOf("&fp=") + 4, scanResult.indexOf("&n="));

Log.v("QRScanner", "FN=" + FN);

Log.v("QRScanner", "FD=" + FD);

Log.v("QRScanner", "FPD=" + FPD);

//Toast.makeText(activity, result.getText(), Toast.LENGTH\_SHORT).show();

downloadJson DJ = new downloadJson(getApplicationContext(), FN, FD, FPD);

DJ.execute();

finish();

}

});

}

});

scannerView.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

mCodeScanner.startPreview();

}

});

}

@Override

protected void onResume() {

super.onResume();

mCodeScanner.startPreview();

}

@Override

protected void onPause() {

mCodeScanner.releaseResources();

super.onPause();

}

}

Класс ItemActivity.java:

public class ItemActivity extends AppCompatActivity {

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_item);

ReceiptsDBHelper handler = new ReceiptsDBHelper(this);

SQLiteDatabase db = handler.getReadableDatabase();

Locale myLocale = new Locale("ru","RU");

NumberFormat n = NumberFormat.getCurrencyInstance(myLocale);

TextView tvItemName = (TextView) findViewById(R.id.item\_name);

TextView tvItemSum = (TextView) findViewById(R.id.item\_sum);

TextView tvItemPrice = (TextView) findViewById(R.id.item\_price);

TextView tvItemQuantity = (TextView) findViewById(R.id.item\_quantity);

TextView tvItemNds10Label = (TextView) findViewById(R.id.item\_nds10\_label);

TextView tvItemNds18Label = (TextView) findViewById(R.id.item\_nds18\_label);

TextView tvItemNds10 = (TextView) findViewById(R.id.item\_nds10);

TextView tvItemNds18 = (TextView) findViewById(R.id.item\_nds18);

TextView tvItemMinPrice = (TextView) findViewById(R.id.item\_card\_view\_min\_price);

TextView tvItemMinPriceDate = (TextView) findViewById(R.id.item\_card\_view\_min\_price\_date);

TextView tvItemMaxPriceDate = (TextView) findViewById(R.id.item\_card\_view\_max\_price\_date);

TextView tvItemMaxPrice = (TextView) findViewById(R.id.item\_card\_view\_max\_price);

TextView tvItemPurchasesSum = (TextView) findViewById(R.id.item\_card\_view\_purchases\_sum);

TextView tvItemPurchasesQuantity = (TextView) findViewById(R.id.item\_card\_view\_purchases\_quantity);

TextView tvItemPriceNoDifference = (TextView) findViewById(R.id.item\_card\_view\_price\_no\_difference);

LinearLayout llStats = (LinearLayout) findViewById(R.id.layout\_stats);

Intent intent = getIntent();

Uri currentItemUri = intent.getData();

String[] itemProjection = {

DataContract.GoodEntry.COLUMN\_GOOD\_NAME,

DataContract.GoodEntry.COLUMN\_GOOD\_SUM,

DataContract.GoodEntry.COLUMN\_GOOD\_QUANTITY,

DataContract.GoodEntry.COLUMN\_GOOD\_PRICE,

DataContract.GoodEntry.COLUMN\_GOOD\_NDS18,

DataContract.GoodEntry.COLUMN\_GOOD\_NDS10

};

Cursor itemCursor = getContentResolver().query(currentItemUri, null, null, null, null);

itemCursor.moveToNext();

Log.v("itemID", String.valueOf(currentItemUri));

String itemName = itemCursor.getString(itemCursor.getColumnIndex(DataContract.GoodEntry.COLUMN\_GOOD\_NAME));

String itemArraySelection = DataContract.GoodEntry.COLUMN\_GOOD\_NAME + " LIKE '" + itemName + "'";

Cursor itemArray = getContentResolver().query(DataContract.GoodEntry.CONTENT\_URI, null, itemArraySelection, null, null);

if (itemArray.getCount() > 1) {

if (itemArray.getCount() % 10 == 1) tvItemPurchasesQuantity.setText(itemArray.getCount() + " покупка");

else if (itemArray.getCount() % 10 == 2) tvItemPurchasesQuantity.setText(itemArray.getCount() + " покупки");

else if (itemArray.getCount() % 10 == 3) tvItemPurchasesQuantity.setText(itemArray.getCount() + " покупки");

else if (itemArray.getCount() % 10 == 4) tvItemPurchasesQuantity.setText(itemArray.getCount() + " покупки");

else tvItemPurchasesQuantity.setText(itemArray.getCount() + " покупок");

Cursor sumCursor = db.rawQuery("SELECT SUM(" + DataContract.GoodEntry.COLUMN\_GOOD\_SUM + ") as Total FROM " + DataContract.GoodEntry.TABLE\_NAME + " WHERE " + DataContract.GoodEntry.COLUMN\_GOOD\_NAME + " = '" + itemName + "'", null);

if (sumCursor.moveToFirst()) {

long total = sumCursor.getLong(sumCursor.getColumnIndex("Total"));// get final total

Log.v("TotalSum", String.valueOf(total));

tvItemPurchasesSum.setText(n.format(total / 100.00));

}

long itemMinPrice;

long itemMaxPrice;

Cursor itemMaxPriceCursor = db.query(DataContract.GoodEntry.TABLE\_NAME, null, DataContract.GoodEntry.COLUMN\_GOOD\_PRICE + " =(SELECT MAX(" + DataContract.GoodEntry.COLUMN\_GOOD\_PRICE + ") FROM " + DataContract.GoodEntry.TABLE\_NAME+ " WHERE " + DataContract.GoodEntry.COLUMN\_GOOD\_NAME + " = '" + itemName + "')", null, null, null, null);

itemMaxPriceCursor.moveToFirst();

itemMaxPrice = itemMaxPriceCursor.getLong(itemMaxPriceCursor.getColumnIndex(DataContract.GoodEntry.COLUMN\_GOOD\_PRICE));

Cursor itemMinPriceCursor = db.query(DataContract.GoodEntry.TABLE\_NAME, null, DataContract.GoodEntry.COLUMN\_GOOD\_PRICE + " =(SELECT MIN(" + DataContract.GoodEntry.COLUMN\_GOOD\_PRICE + ") FROM " + DataContract.GoodEntry.TABLE\_NAME+ " WHERE " + DataContract.GoodEntry.COLUMN\_GOOD\_NAME + " = '" + itemName + "')", null, null, null, null);

itemMinPriceCursor.moveToFirst();

itemMinPrice = itemMinPriceCursor.getLong(itemMinPriceCursor.getColumnIndex(DataContract.GoodEntry.COLUMN\_GOOD\_PRICE));

if (itemMaxPrice != itemMinPrice){

SimpleDateFormat sdf = new SimpleDateFormat("dd.MM.yyyy HH:mm"); // the format of your date

sdf.setTimeZone(TimeZone.getTimeZone("Etc/UTC"));

String purchaseItemMaxSelection = DataContract.PurchaseEntry.COLUMN\_PURCHASE\_ID + " = " + itemMaxPriceCursor.getString(itemMaxPriceCursor.getColumnIndex(DataContract.GoodEntry.COLUMN\_GOOD\_PURCHASE\_ID\_FK));

Cursor purchaseItemMax = getContentResolver().query(DataContract.PurchaseEntry.CONTENT\_URI,null, purchaseItemMaxSelection, null, null);

purchaseItemMax.moveToFirst();

tvItemMaxPrice.setText(n.format(itemMaxPrice/100.00));

tvItemMaxPriceDate.setText(sdf.format(purchaseItemMax.getLong(purchaseItemMax.getColumnIndexOrThrow(DataContract.PurchaseEntry.COLUMN\_PURCHASE\_DATETIME))\*1000L));

purchaseItemMax.close();

itemMaxPriceCursor.close();

String purchaseItemMinSelection = DataContract.PurchaseEntry.COLUMN\_PURCHASE\_ID + " = " + itemMinPriceCursor.getString(itemMinPriceCursor.getColumnIndex(DataContract.GoodEntry.COLUMN\_GOOD\_PURCHASE\_ID\_FK));

Cursor purchaseItemMin = getContentResolver().query(DataContract.PurchaseEntry.CONTENT\_URI,null, purchaseItemMinSelection, null, null);

purchaseItemMin.moveToFirst();

tvItemMinPrice.setText(n.format(itemMinPriceCursor.getLong(itemMinPriceCursor.getColumnIndex(DataContract.GoodEntry.COLUMN\_GOOD\_PRICE))/100.00));

tvItemMinPriceDate.setText(sdf.format(purchaseItemMin.getLong(purchaseItemMin.getColumnIndexOrThrow(DataContract.PurchaseEntry.COLUMN\_PURCHASE\_DATETIME))\*1000L));

purchaseItemMin.close();

itemMinPriceCursor.close();

tvItemPriceNoDifference.setVisibility(GONE);

}

else

{

llStats.setVisibility(GONE);

}

}

else

{

tvItemPurchasesQuantity.setText("1 покупка");

tvItemPurchasesSum.setText(n.format(itemCursor.getLong(itemCursor.getColumnIndex(DataContract.GoodEntry.COLUMN\_GOOD\_SUM))/100.00));

llStats.setVisibility(GONE);

tvItemPriceNoDifference.setVisibility(GONE);

}

tvItemName.setText(itemCursor.getString(itemCursor.getColumnIndex(DataContract.GoodEntry.COLUMN\_GOOD\_NAME)));

tvItemSum.setText(n.format(itemCursor.getLong(itemCursor.getColumnIndex(DataContract.GoodEntry.COLUMN\_GOOD\_SUM)) / 100.0));

tvItemPrice.setText(n.format(itemCursor.getLong(itemCursor.getColumnIndex(DataContract.GoodEntry.COLUMN\_GOOD\_PRICE))/100.0));

tvItemQuantity.setText(itemCursor.getString(itemCursor.getColumnIndex(DataContract.GoodEntry.COLUMN\_GOOD\_QUANTITY)));

if (itemCursor.getString(itemCursor.getColumnIndex(DataContract.GoodEntry.COLUMN\_GOOD\_NDS10)) != null){

tvItemNds10.setText(n.format(itemCursor.getLong(itemCursor.getColumnIndex(DataContract.GoodEntry.COLUMN\_GOOD\_NDS10))/100.0));

} else {

tvItemNds10.setVisibility(GONE);

tvItemNds10Label.setVisibility(GONE);

}

if (itemCursor.getString(itemCursor.getColumnIndex(DataContract.GoodEntry.COLUMN\_GOOD\_NDS18)) != null){

tvItemNds18.setText(n.format(itemCursor.getLong(itemCursor.getColumnIndex(DataContract.GoodEntry.COLUMN\_GOOD\_NDS18))/100.0));

} else {

tvItemNds18.setVisibility(GONE);

tvItemNds18Label.setVisibility(GONE);

}

}

}

Класс PointActivity.java:

public class PointActivity extends AppCompatActivity {

private long pointID;

private long storeID;

Cursor pointCursor;

Cursor storeCursor;

Uri currentPointUri;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_point);

ReceiptsDBHelper handler = new ReceiptsDBHelper(this);

Locale myLocale = new Locale("ru","RU");

TextView tvPointPrimaryName = (TextView) findViewById(R.id.point\_name);

TextView tvPointAddress = (TextView) findViewById(R.id.point\_address);

TextView tvPointTotalSum = (TextView) findViewById(R.id.point\_card\_view\_purchases\_sum);

TextView tvPointPurchasesAmount = (TextView) findViewById(R.id.point\_card\_view\_purchases\_amount);

Intent intent = getIntent();

currentPointUri = intent.getData();

pointCursor = getContentResolver().query(currentPointUri, null, null, null, null);

pointCursor.moveToFirst();

pointID = pointCursor.getLong(pointCursor.getColumnIndex(DataContract.PointEntry.COLUMN\_POINT\_ID));

storeID = pointCursor.getLong(pointCursor.getColumnIndex(DataContract.PointEntry.COLUMN\_STORE\_ID\_FK));

String storeSelection = DataContract.StoreEntry.COLUMN\_STORE\_ID + " LIKE " + storeID;

storeCursor = getContentResolver().query(DataContract.StoreEntry.CONTENT\_URI, null, storeSelection, null, null);

storeCursor.moveToFirst();

String storeName = null;

String storeSecondName = null;

int purchasesCount = 0;

if (storeCursor.getString(storeCursor.getColumnIndex(DataContract.StoreEntry.COLUMN\_STORE\_NICKNAME)) != null){

tvPointPrimaryName.setText(storeCursor.getString(storeCursor.getColumnIndex(DataContract.StoreEntry.COLUMN\_STORE\_NICKNAME)));

}

else if (storeCursor.getString(storeCursor.getColumnIndex(DataContract.StoreEntry.COLUMN\_STORE\_NAME)) != null){

tvPointPrimaryName.setText(storeCursor.getString(storeCursor.getColumnIndex(DataContract.StoreEntry.COLUMN\_STORE\_NAME)));

}

else {

tvPointPrimaryName.setText("ИНН " + storeCursor.getString(storeCursor.getColumnIndex(DataContract.StoreEntry.COLUMN\_STORE\_INN)));

}

if (pointCursor.getString(pointCursor.getColumnIndex(DataContract.PointEntry.COLUMN\_POINT\_NICKNAME)) != null){

tvPointAddress.setText(pointCursor.getString(pointCursor.getColumnIndex(DataContract.PointEntry.COLUMN\_POINT\_NICKNAME)));

}

else if (pointCursor.getString(pointCursor.getColumnIndex(DataContract.PointEntry.COLUMN\_POINT\_ADDRESS)) != null){

tvPointAddress.setText(pointCursor.getString(pointCursor.getColumnIndex(DataContract.PointEntry.COLUMN\_POINT\_ADDRESS)));

}

else {

tvPointAddress.setText("Точка без адреса");

}

long purchasesPointCount = handler.getPurchasesCount(

DataContract.PurchaseEntry.COLUMN\_PURCHASE\_POINT\_ID\_FK + " LIKE " + pointID,

null);

if (purchasesPointCount % 10 == 1) tvPointPurchasesAmount.setText(String.valueOf(purchasesPointCount) + " покупка");

else if (purchasesPointCount % 10 == 2) tvPointPurchasesAmount.setText(String.valueOf(purchasesPointCount) + " покупки");

else if (purchasesPointCount % 10 == 3) tvPointPurchasesAmount.setText(String.valueOf(purchasesPointCount) + " покупки");

else if (purchasesPointCount % 10 == 4) tvPointPurchasesAmount.setText(String.valueOf(purchasesPointCount) + " покупки");

else tvPointPurchasesAmount.setText(String.valueOf(purchasesPointCount) + " покупок");

SQLiteDatabase db = handler.getReadableDatabase();

Cursor purchaseSumCursor = db.rawQuery("select sum(" + DataContract.PurchaseEntry.COLUMN\_PURCHASE\_TOTALSUM +

") from " + DataContract.PurchaseEntry.TABLE\_NAME +

" where " + DataContract.PurchaseEntry.COLUMN\_PURCHASE\_POINT\_ID\_FK + " = '" + pointID + "' ;", null);

long amount;

if(purchaseSumCursor.moveToFirst())

amount = purchaseSumCursor.getLong(0);

else

amount = -1;

purchaseSumCursor.close();

NumberFormat n = NumberFormat.getCurrencyInstance(myLocale);

String formatedTotalSum = n.format(amount / 100.0);

/\*\* Делаем целую часть суммы жирной \*\*/

SpannableStringBuilder str = new SpannableStringBuilder(formatedTotalSum);

str.setSpan(new android.text.style.StyleSpan(android.graphics.Typeface.BOLD), 0, formatedTotalSum.length()-4, Spannable.SPAN\_EXCLUSIVE\_EXCLUSIVE);

tvPointTotalSum.setText(str);

Cursor purchasePointCursor = db.rawQuery("SELECT \* FROM " + DataContract.PurchaseEntry.TABLE\_NAME + " WHERE " + DataContract.PurchaseEntry.COLUMN\_PURCHASE\_POINT\_ID\_FK + " LIKE " + pointID, null);

// Find ListView to populate

ListView lvPoint = (ListView) findViewById(R.id.point\_listview);

// Setup cursor adapter using cursor from last step

PurchasePointCursorAdapter purchasePointAdapter = new PurchasePointCursorAdapter(this, purchasePointCursor);

// Attach cursor adapter to the ListView

lvPoint.setAdapter(purchasePointAdapter);

lvPoint.setOnItemClickListener(new AdapterView.OnItemClickListener() {

@Override

public void onItemClick(AdapterView<?> adapterView, View view, int position, long id) {

Intent intent = new Intent(getApplicationContext(), PurchaseActivity.class);

Uri currentPurchaseUri = ContentUris.withAppendedId(DataContract.PurchaseEntry.CONTENT\_URI, id);

// Set the URI on the data field of the intent

intent.setData(currentPurchaseUri);

//intent.putExtra("id", debtID);

startActivity(intent);

}

});

}

public boolean onCreateOptionsMenu(Menu menu) {

// Inflate the menu; this adds items to the action bar if it is present.

getMenuInflater().inflate(R.menu.menu\_store, menu);

return true;

}

@Override

public boolean onOptionsItemSelected(MenuItem item) {

// Handle action bar item clicks here. The action bar will

// automatically handle clicks on the Home/Up button, so long

// as you specify a parent activity in AndroidManifest.xml.

int id = item.getItemId();

if (id == R.id.action\_rename) {

showStoreRenameDialog();

return true;

}

if (id == R.id.action\_help) {

Intent intent = new Intent(this, WebViewActivity.class);

startActivity(intent);

return true;

}

return super.onOptionsItemSelected(item);

}

public void showStoreRenameDialog() {

AlertDialog.Builder dialogBuilder = new AlertDialog.Builder(this);

LayoutInflater inflater = this.getLayoutInflater();

final View dialogView = inflater.inflate(R.layout.rename\_dialog, null);

dialogBuilder.setView(dialogView);

final EditText edt = (EditText) dialogView.findViewById(R.id.editText);

final TextView tvPointAddress = (TextView) findViewById(R.id.point\_address);

dialogBuilder.setTitle("Переименовать точку");

dialogBuilder.setMessage("Введите новое название");

dialogBuilder.setPositiveButton("Готово", new DialogInterface.OnClickListener() {

public void onClick(DialogInterface dialog, int whichButton) {

ContentValues pointValues = new ContentValues();

pointValues.put(DataContract.PointEntry.COLUMN\_POINT\_NICKNAME, edt.getText().toString());

int numOfUpdatedRows = getContentResolver().update(DataContract.PointEntry.CONTENT\_URI, pointValues,

DataContract.PointEntry.COLUMN\_POINT\_ID + " LIKE " + pointID, null);

Log.v("Количество обновленных", String.valueOf(numOfUpdatedRows) );

tvPointAddress.setText(edt.getText().toString());

}

});

dialogBuilder.setNegativeButton("Отменить", new DialogInterface.OnClickListener() {

public void onClick(DialogInterface dialog, int whichButton) {

//pass

}

});

pointCursor = getContentResolver().query(currentPointUri, null, null, null, null);

pointCursor.moveToFirst();

if (pointCursor.getString(pointCursor.getColumnIndex(DataContract.PointEntry.COLUMN\_POINT\_NICKNAME)) != null) {

dialogBuilder.setNeutralButton("Удалить", new DialogInterface.OnClickListener() {

public void onClick(DialogInterface dialog, int whichButton) {

ContentValues pointValues = new ContentValues();

pointValues.putNull(DataContract.PointEntry.COLUMN\_POINT\_NICKNAME);

int numOfUpdatedRows = getContentResolver().update(DataContract.PointEntry.CONTENT\_URI, pointValues,

DataContract.PointEntry.COLUMN\_POINT\_ID + " LIKE " + pointID, null);

Log.v("Количество обновленных", String.valueOf(numOfUpdatedRows));

if (pointCursor.getString(pointCursor.getColumnIndex(DataContract.PointEntry.COLUMN\_POINT\_ADDRESS)) != null) {

tvPointAddress.setText(pointCursor.getString(pointCursor.getColumnIndex(DataContract.PointEntry.COLUMN\_POINT\_ADDRESS)));

} else {

tvPointAddress.setText("Точка без адреса");

}

}

});

}

AlertDialog b = dialogBuilder.create();

b.show();

}

}

Класс PurchaseActivity.java:

public class PurchaseActivity extends AppCompatActivity {

private long purchaseID;

private long storeID;

private long pointID;

final String LOG\_TAG = PurchaseActivity.class.getSimpleName();

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_purchase);

Locale myLocale = new Locale("ru","RU");

TextView tvPurchaseDate = (TextView) findViewById(R.id.purchase\_date);

TextView tvPurchaseStoreName = (TextView) findViewById(R.id.purchase\_store\_name);

TextView tvPurchaseTotalSum = (TextView) findViewById(R.id.purchase\_total\_sum);

Intent intent = getIntent();

Uri currentPurchaseUri = intent.getData();

Cursor purchaseCursor = getContentResolver().query(currentPurchaseUri, null, null, null, null);

purchaseCursor.moveToNext();

//присваиваем глобальной переменной ID покупки, так как это понадобится для удаления.

purchaseID = purchaseCursor.getLong(purchaseCursor.getColumnIndex(DataContract.PurchaseEntry.COLUMN\_PURCHASE\_ID));

storeID = purchaseCursor.getLong(purchaseCursor.getColumnIndex(DataContract.PurchaseEntry.COLUMN\_PURCHASE\_STORE\_ID\_FK));

pointID = purchaseCursor.getLong(purchaseCursor.getColumnIndex(DataContract.PurchaseEntry.COLUMN\_PURCHASE\_POINT\_ID\_FK));

Log.v(LOG\_TAG, "Проверим выдачу ID покупки, да? ID покупки равен " + purchaseID);

NumberFormat n = NumberFormat.getCurrencyInstance(myLocale);

String formatedTotalSum = n.format(purchaseCursor.getLong(purchaseCursor.getColumnIndex(DataContract.PurchaseEntry.COLUMN\_PURCHASE\_TOTALSUM)) / 100.0);

/\*\* Делаем целую часть суммы жирной \*\*/

SpannableStringBuilder str = new SpannableStringBuilder(formatedTotalSum);

str.setSpan(new android.text.style.StyleSpan(android.graphics.Typeface.BOLD), 0, formatedTotalSum.length()-4, Spannable.SPAN\_EXCLUSIVE\_EXCLUSIVE);

tvPurchaseTotalSum.setText(str);

SimpleDateFormat sdf = new SimpleDateFormat("dd.MM.yyyy HH:mm"); // the format of your date

sdf.setTimeZone(TimeZone.getTimeZone("Etc/UTC"));

String formattedDate = sdf.format(purchaseCursor.getLong(purchaseCursor.getColumnIndexOrThrow(DataContract.PurchaseEntry.COLUMN\_PURCHASE\_DATETIME))\*1000L);

tvPurchaseDate.setText(formattedDate);

String[] storeProjection = {

DataContract.StoreEntry.COLUMN\_STORE\_INN,

DataContract.StoreEntry.COLUMN\_STORE\_NICKNAME,

DataContract.StoreEntry.COLUMN\_STORE\_NAME

};

String storeSelection = DataContract.StoreEntry.COLUMN\_STORE\_ID + " LIKE ?";

String[] storeSelectionArgs = {purchaseCursor.getString(purchaseCursor.getColumnIndexOrThrow(DataContract.PurchaseEntry.COLUMN\_PURCHASE\_STORE\_ID\_FK))};

Cursor storeCursor = getContentResolver().query(DataContract.StoreEntry.CONTENT\_URI,

storeProjection,

storeSelection,

storeSelectionArgs,

null

);

storeCursor.moveToNext();

String storeName = null;

int purchasesCount = 0;

if (storeCursor.getString(storeCursor.getColumnIndex(DataContract.StoreEntry.COLUMN\_STORE\_NICKNAME)) != null){

storeName = storeCursor.getString(storeCursor.getColumnIndex(DataContract.StoreEntry.COLUMN\_STORE\_NICKNAME));

}

else if (storeCursor.getString(storeCursor.getColumnIndex(DataContract.StoreEntry.COLUMN\_STORE\_NAME)) != null){

storeName = storeCursor.getString(storeCursor.getColumnIndex(DataContract.StoreEntry.COLUMN\_STORE\_NAME));

}

else storeName = "ИНН " + storeCursor.getString(storeCursor.getColumnIndex(DataContract.StoreEntry.COLUMN\_STORE\_INN));

tvPurchaseStoreName.setText(storeName);

ReceiptsDBHelper handler = new ReceiptsDBHelper(this);

// Get access to the underlying writeable database

SQLiteDatabase db = handler.getWritableDatabase();

// Query for items from the database and get a cursor back

String[] itemProjection = {

DataContract.GoodEntry.COLUMN\_GOOD\_ID,

DataContract.GoodEntry.COLUMN\_GOOD\_NAME,

DataContract.GoodEntry.COLUMN\_GOOD\_QUANTITY,

DataContract.GoodEntry.COLUMN\_GOOD\_SUM,

DataContract.GoodEntry.COLUMN\_GOOD\_PRICE

};

String itemSelection = DataContract.GoodEntry.COLUMN\_GOOD\_PURCHASE\_ID\_FK + " LIKE ?";

String[] itemSelectionArgs = {purchaseCursor.getString(purchaseCursor.getColumnIndexOrThrow(DataContract.PurchaseEntry.COLUMN\_PURCHASE\_ID))};

Cursor itemCursor = db.query(DataContract.GoodEntry.TABLE\_NAME ,

itemProjection,

itemSelection,

itemSelectionArgs,

null,

null,

null

);

Log.v(LOG\_TAG, "Количество товаров в покупке" + String.valueOf(itemCursor.getCount()));

//db.rawQuery("SELECT \* FROM " + DataContract.GoodEntry.TABLE\_NAME , null);

// Find ListView to populate

ListView lvItems = (ListView) findViewById(R.id.fragment\_listview);

Log.v(LOG\_TAG, "ListView is set");

// Setup cursor adapter using cursor from last step

ItemCursorAdapter itemAdapter = new ItemCursorAdapter(this, itemCursor);

// Attach cursor adapter to the ListView

lvItems.setAdapter(itemAdapter);

itemAdapter.changeCursor(itemCursor);

lvItems.setOnItemClickListener(new AdapterView.OnItemClickListener() {

@Override

public void onItemClick(AdapterView<?> adapterView, View view, int position, long id) {

Intent intent = new Intent(getApplicationContext(), ItemActivity.class);

Uri currentDebtUri = ContentUris.withAppendedId(DataContract.GoodEntry.CONTENT\_URI, id);

// Set the URI on the data field of the intent

intent.setData(currentDebtUri);

//intent.putExtra("id", debtID);

startActivity(intent);

}

});

}

public boolean onCreateOptionsMenu(Menu menu) {

// Inflate the menu; this adds items to the action bar if it is present.

getMenuInflater().inflate(R.menu.menu\_purchase, menu);

return true;

}

@Override

public boolean onOptionsItemSelected(MenuItem item) {

// Handle action bar item clicks here. The action bar will

// automatically handle clicks on the Home/Up button, so long

// as you specify a parent activity in AndroidManifest.xml.

int id = item.getItemId();

if (id == R.id.action\_delete) {

new AlertDialog.Builder(this)

.setTitle("Удаление")

.setMessage("Вы действительно хотите удалить эту покупку?")

//.setIcon(android.R.drawable.ic\_dialog\_alert)

.setPositiveButton(android.R.string.yes, new DialogInterface.OnClickListener() {

public void onClick(DialogInterface dialog, int whichButton) {

deletePurchase();

Toast toast = Toast.makeText(getApplicationContext(), "Покупка удалена", Toast.LENGTH\_SHORT);

toast.show();

finish();

}})

.setNegativeButton(android.R.string.no, null).show();

return true;

}

if (id == R.id.action\_help) {

Intent intent = new Intent(this, WebViewActivity.class);

startActivity(intent);

return true;

}

return super.onOptionsItemSelected(item);

}

private void deletePurchase() {

//Создаём хендлер для вызова метода счёта покупок

ReceiptsDBHelper handler = new ReceiptsDBHelper(this);

int itemRowsDeleted = getContentResolver().delete(

DataContract.GoodEntry.CONTENT\_URI,

DataContract.GoodEntry.COLUMN\_GOOD\_PURCHASE\_ID\_FK + " LIKE " + purchaseID,

null);

Log.v(LOG\_TAG, "Количество удалённых товаров " + itemRowsDeleted);

if (itemRowsDeleted > 0) {

int purchaseRowsDeleted = getContentResolver().delete(

DataContract.PurchaseEntry.CONTENT\_URI,

DataContract.PurchaseEntry.COLUMN\_PURCHASE\_ID + " LIKE " + purchaseID,

null);

Log.v(LOG\_TAG, "Количество удалённых покупок " + purchaseRowsDeleted);

}

long purchasesPointCount = handler.getPurchasesCount(

DataContract.PurchaseEntry.COLUMN\_PURCHASE\_POINT\_ID\_FK + " LIKE " + storeID,

null);

if (purchasesPointCount == 0) {

int pointRowsDeleted = getContentResolver().delete(

DataContract.PointEntry.CONTENT\_URI,

DataContract.PointEntry.COLUMN\_POINT\_ID + " LIKE " + pointID,

null);

Log.v(LOG\_TAG, "Количество удалённых точек продаж " + pointRowsDeleted);

long purchasesStoreCount = handler.getPurchasesCount(

DataContract.PurchaseEntry.COLUMN\_PURCHASE\_STORE\_ID\_FK + " LIKE " + storeID,

null);

if (purchasesStoreCount == 0) {

int storeRowsDeleted = getContentResolver().delete(

DataContract.StoreEntry.CONTENT\_URI,

DataContract.StoreEntry.COLUMN\_STORE\_ID + " LIKE " + storeID,

null);

Log.v(LOG\_TAG, "Количество удалённых магазинов " + storeRowsDeleted);

}

}

}

@Override

protected void onDestroy(){

super.onDestroy();

setResult(Activity.RESULT\_OK);

}

}

Класс StoreActivity.java:

public class StoreActivity extends AppCompatActivity {

private long storeID;

Cursor storeCursor;

Uri currentStoreUri;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_store);

ReceiptsDBHelper handler = new ReceiptsDBHelper(this);

Locale myLocale = new Locale("ru","RU");

TextView tvStorePrimaryName = (TextView) findViewById(R.id.store\_primary\_name);

TextView tvStoreSecondName = (TextView) findViewById(R.id.store\_second\_name);

TextView tvStoreTotalSum = (TextView) findViewById(R.id.item\_card\_view\_purchases\_sum);

TextView tvStorePurchasesAmount = (TextView) findViewById(R.id.item\_card\_view\_purchases\_quantity);

Intent intent = getIntent();

currentStoreUri = intent.getData();

storeCursor = getContentResolver().query(currentStoreUri, null, null, null, null);

storeCursor.moveToNext();

storeID = storeCursor.getLong(storeCursor.getColumnIndex(DataContract.StoreEntry.COLUMN\_STORE\_ID));

String storeName = null;

String storeSecondName = null;

int purchasesCount = 0;

if (storeCursor.getString(storeCursor.getColumnIndex(DataContract.StoreEntry.COLUMN\_STORE\_NICKNAME)) != null){

tvStorePrimaryName.setText(storeCursor.getString(storeCursor.getColumnIndex(DataContract.StoreEntry.COLUMN\_STORE\_NICKNAME)));

if (storeCursor.getString(storeCursor.getColumnIndex(DataContract.StoreEntry.COLUMN\_STORE\_NAME)) != null) {

tvStoreSecondName.setText(storeCursor.getString(storeCursor.getColumnIndex(DataContract.StoreEntry.COLUMN\_STORE\_NAME)));

} else tvStoreSecondName.setText("ИНН " + storeCursor.getString(storeCursor.getColumnIndex(DataContract.StoreEntry.COLUMN\_STORE\_INN)));

}

else if (storeCursor.getString(storeCursor.getColumnIndex(DataContract.StoreEntry.COLUMN\_STORE\_NAME)) != null){

tvStorePrimaryName.setText(storeCursor.getString(storeCursor.getColumnIndex(DataContract.StoreEntry.COLUMN\_STORE\_NAME)));

tvStoreSecondName.setText("ИНН " + storeCursor.getString(storeCursor.getColumnIndex(DataContract.StoreEntry.COLUMN\_STORE\_INN)));

}

else {

tvStorePrimaryName.setText("ИНН " + storeCursor.getString(storeCursor.getColumnIndex(DataContract.StoreEntry.COLUMN\_STORE\_INN)));

tvStoreSecondName.setVisibility(View.GONE);

}

long purchasesStoreCount = handler.getPurchasesCount(

DataContract.PurchaseEntry.COLUMN\_PURCHASE\_STORE\_ID\_FK + " LIKE " + storeID,

null);

if (purchasesStoreCount % 10 == 1) tvStorePurchasesAmount.setText(String.valueOf(purchasesStoreCount) + " покупка");

else if (purchasesStoreCount % 10 == 2) tvStorePurchasesAmount.setText(String.valueOf(purchasesStoreCount) + " покупки");

else if (purchasesStoreCount % 10 == 3) tvStorePurchasesAmount.setText(String.valueOf(purchasesStoreCount) + " покупки");

else if (purchasesStoreCount % 10 == 4) tvStorePurchasesAmount.setText(String.valueOf(purchasesStoreCount) + " покупки");

else tvStorePurchasesAmount.setText(String.valueOf(purchasesStoreCount) + " покупок");

SQLiteDatabase db = handler.getReadableDatabase();

Cursor purchaseSumCursor = db.rawQuery("select sum(" + DataContract.PurchaseEntry.COLUMN\_PURCHASE\_TOTALSUM +

") from " + DataContract.PurchaseEntry.TABLE\_NAME +

" where " + DataContract.PurchaseEntry.COLUMN\_PURCHASE\_STORE\_ID\_FK + " = '" + storeID + "' ;", null);

long amount;

if(purchaseSumCursor.moveToFirst())

amount = purchaseSumCursor.getLong(0);

else

amount = -1;

purchaseSumCursor.close();

NumberFormat n = NumberFormat.getCurrencyInstance(myLocale);

String formatedTotalSum = n.format(amount / 100.0);

/\*\* Делаем целую часть суммы жирной \*\*/

SpannableStringBuilder str = new SpannableStringBuilder(formatedTotalSum);

str.setSpan(new android.text.style.StyleSpan(android.graphics.Typeface.BOLD), 0, formatedTotalSum.length()-4, Spannable.SPAN\_EXCLUSIVE\_EXCLUSIVE);

tvStoreTotalSum.setText(str);

Cursor purchaseCursor = db.rawQuery("SELECT \* FROM " + DataContract.PointEntry.TABLE\_NAME + " WHERE " + DataContract.PointEntry.COLUMN\_STORE\_ID\_FK + " LIKE " + storeID, null);

// Find ListView to populate

ListView lvPoint = (ListView) findViewById(R.id.store\_listview);

// Setup cursor adapter using cursor from last step

PointCursorAdapter pointAdapter = new PointCursorAdapter(this, purchaseCursor);

// Attach cursor adapter to the ListView

lvPoint.setAdapter(pointAdapter);

lvPoint.setOnItemClickListener(new AdapterView.OnItemClickListener() {

@Override

public void onItemClick(AdapterView<?> adapterView, View view, int position, long id) {

Intent intent = new Intent(getApplicationContext(), PointActivity.class);

Uri currentPointUri = ContentUris.withAppendedId(DataContract.PointEntry.CONTENT\_URI, id);

// Set the URI on the data field of the intent

intent.setData(currentPointUri);

//intent.putExtra("id", debtID);

startActivity(intent);

}

});

}

public boolean onCreateOptionsMenu(Menu menu) {

// Inflate the menu; this adds items to the action bar if it is present.

getMenuInflater().inflate(R.menu.menu\_store, menu);

return true;

}

@Override

public boolean onOptionsItemSelected(MenuItem item) {

// Handle action bar item clicks here. The action bar will

// automatically handle clicks on the Home/Up button, so long

// as you specify a parent activity in AndroidManifest.xml.

int id = item.getItemId();

if (id == R.id.action\_rename) {

showStoreRenameDialog();

return true;

}

if (id == R.id.action\_help) {

Intent intent = new Intent(this, WebViewActivity.class);

startActivity(intent);

return true;

}

return super.onOptionsItemSelected(item);

}

public void showStoreRenameDialog() {

AlertDialog.Builder dialogBuilder = new AlertDialog.Builder(this);

LayoutInflater inflater = this.getLayoutInflater();

final View dialogView = inflater.inflate(R.layout.rename\_dialog, null);

dialogBuilder.setView(dialogView);

final EditText edt = (EditText) dialogView.findViewById(R.id.editText);

final TextView tvStorePrimaryName = (TextView) findViewById(R.id.store\_primary\_name);

final TextView tvStoreSecondName = (TextView) findViewById(R.id.store\_second\_name);

dialogBuilder.setTitle("Переименовать магазин");

dialogBuilder.setMessage("Введите новое название");

dialogBuilder.setPositiveButton("Готово", new DialogInterface.OnClickListener() {

public void onClick(DialogInterface dialog, int whichButton) {

ContentValues storeValues = new ContentValues();

storeValues.put(DataContract.StoreEntry.COLUMN\_STORE\_NICKNAME, edt.getText().toString());

int numOfUpdatedRows = getContentResolver().update(DataContract.StoreEntry.CONTENT\_URI, storeValues,

DataContract.StoreEntry.COLUMN\_STORE\_ID + " LIKE " + storeID, null);

Log.v("Количество обновленных", String.valueOf(numOfUpdatedRows) );

storeCursor = getContentResolver().query(currentStoreUri, null, null, null, null);

storeCursor.moveToNext();

//TODO: Это повторяющийся код. Вынести в отдельный метод и поработать с излишним вызовом курсоров

if (storeCursor.getString(storeCursor.getColumnIndex(DataContract.StoreEntry.COLUMN\_STORE\_NICKNAME)) != null){

tvStorePrimaryName.setText(storeCursor.getString(storeCursor.getColumnIndex(DataContract.StoreEntry.COLUMN\_STORE\_NICKNAME)));

if (storeCursor.getString(storeCursor.getColumnIndex(DataContract.StoreEntry.COLUMN\_STORE\_NAME)) != null) {

tvStoreSecondName.setText(storeCursor.getString(storeCursor.getColumnIndex(DataContract.StoreEntry.COLUMN\_STORE\_NAME)));

} else tvStoreSecondName.setText("ИНН " + storeCursor.getString(storeCursor.getColumnIndex(DataContract.StoreEntry.COLUMN\_STORE\_INN)));

}

else if (storeCursor.getString(storeCursor.getColumnIndex(DataContract.StoreEntry.COLUMN\_STORE\_NAME)) != null){

tvStorePrimaryName.setText(storeCursor.getString(storeCursor.getColumnIndex(DataContract.StoreEntry.COLUMN\_STORE\_NAME)));

tvStoreSecondName.setText("ИНН " + storeCursor.getString(storeCursor.getColumnIndex(DataContract.StoreEntry.COLUMN\_STORE\_INN)));

}

else {

tvStorePrimaryName.setText("ИНН " + storeCursor.getString(storeCursor.getColumnIndex(DataContract.StoreEntry.COLUMN\_STORE\_INN)));

tvStoreSecondName.setVisibility(View.GONE);

}

tvStorePrimaryName.setPaintFlags(0);

tvStoreSecondName.setVisibility(View.VISIBLE);

}

});

dialogBuilder.setNegativeButton("Отменить", new DialogInterface.OnClickListener() {

public void onClick(DialogInterface dialog, int whichButton) {

//pass

}

});

storeCursor = getContentResolver().query(currentStoreUri, null, null, null, null);

storeCursor.moveToNext();

if (storeCursor.getString(storeCursor.getColumnIndex(DataContract.StoreEntry.COLUMN\_STORE\_NICKNAME)) != null) {

edt.setText(storeCursor.getString(storeCursor.getColumnIndex(DataContract.StoreEntry.COLUMN\_STORE\_NICKNAME)));

dialogBuilder.setNeutralButton("Удалить", new DialogInterface.OnClickListener() {

public void onClick(DialogInterface dialog, int whichButton) {

ContentValues pointValues = new ContentValues();

pointValues.putNull(DataContract.StoreEntry.COLUMN\_STORE\_NICKNAME);

int numOfUpdatedRows = getContentResolver().update(DataContract.StoreEntry.CONTENT\_URI, pointValues,

DataContract.StoreEntry.COLUMN\_STORE\_ID + " LIKE " + storeID, null);

Log.v("Количество обновленных", String.valueOf(numOfUpdatedRows));

if (storeCursor.getString(storeCursor.getColumnIndex(DataContract.StoreEntry.COLUMN\_STORE\_NAME)) != null){

tvStorePrimaryName.setText(storeCursor.getString(storeCursor.getColumnIndex(DataContract.StoreEntry.COLUMN\_STORE\_NAME)));

tvStoreSecondName.setText("ИНН " + storeCursor.getString(storeCursor.getColumnIndex(DataContract.StoreEntry.COLUMN\_STORE\_INN)));

}

else {

tvStorePrimaryName.setText("ИНН " + storeCursor.getString(storeCursor.getColumnIndex(DataContract.StoreEntry.COLUMN\_STORE\_INN)));

tvStoreSecondName.setVisibility(View.GONE);

}

}

});

}

AlertDialog b = dialogBuilder.create();

b.show();

}

}

Класс WebViewActivity.java:

public class WebViewActivity extends AppCompatActivity {

private WebView webview;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_web\_view);

webview =(WebView)findViewById(R.id.webView);

webview.setWebViewClient(new WebViewClient());

webview.getSettings().setJavaScriptEnabled(true);

webview.getSettings().setDomStorageEnabled(true);

webview.setOverScrollMode(WebView.OVER\_SCROLL\_NEVER);

webview.loadUrl("https://tumist76.github.io/ReceiptManager/");

}

}