Работа с библиотекой интеграции для смарт-терминала “Кассатка”

1. Введение **4**

2. Добавление библиотеки к проекту **4**

3. Работа с весами **5**

3.1 Описание API 5

3.1.1 Открытие карты товара 5

3.1.2 Закрытие карты товара 5

3.1.3 Отправка веса продукта 5

3.1.4 Отправка производителя весов 5

3.1.5 Отправка модели весов 5

3.1.6 Широковещательный приемник данных весов. 5

3.2 Описание работы 6

3.2.1 Описание 6

3.2.2 Пример 7

4. Вызов окон смарт-терминала **8**

4.1 Описание методов класса NavigationApi 8

4.1.1 Создание цели, вызывающей активность учета 8

4.1.2 Создание цели, вызывающей активность быстрой продажи 8

4.1.3 Создание цели, вызывающей активность конфигурации товара 8

4.1.4 Создание цели, вызывающей активность помощи 8

4.1 5 Создание цели, вызывающей активность ОФД 9

4.1.6 Создание цели, вызывающей активность закрытия смены 9

4.1.7 Создание цели, вызывающей активность изъятия или внесения денежных средств в кассу 9

4.1.8 Создание цели, вызывающей активность списка транзакций 9

4.1.9 Создание цели, вызывающей активность транзакции по чеку 9

4.1.10 Создание цели, вызывающей активность смены 9

4.1.11 Создание цели, вызывающей активность создания нового товара 9

4.1.12 Создание цели, вызывающей активность продажи 9

4.1.13 Создание цели, вызывающей активность перехода к оплате 10

4.1.14 Создание цели, вызывающей активность оплаты наличностью 10

4.1.15 Создание цели, вызывающей активность скидок 10

4.1.16 Создание цели, вызывающей активность отправки чека 10

4.1.17 Создание цели, вызывающей активность чека коррекции. 10

4.1.18 Создание цели, вызывающей активность добавления клиента 10

4.1.19 Вызов нового окна 10

4.2 Пример вызова нового окна 10

5. Получение данных из БД смарт-терминала **10**

5.1 Описание доступных таблиц БД смарт-терминала 11
5.1.1 Таблица категорий товара 11
5.1.2 Таблица чеков 11
5.1.3 Таблица позиций чеков 11
5.1.4 Таблица скидок 13
5.1.5 Таблица записей в журнале 13
5.1.6 Таблица карт лояльности 14
5.1.7 Таблица товаров 14
5.1.8 Таблица RabbitMessage 14
5.1.9 Таблица склада 15
5.1.10 Таблица пользователя 15
5.2 Описание методов класса QueryBuilder 15
5.2.1 Конструктор по умолчанию 15
5.2.2 Конструктор с параметрами 16
5.2.3 Выбор колонок таблицы 16
5.2.4 Добавление условия WHERE 16
5.2.5 Добавление полей к условию WHERE 16
5.2.6 Добавление параметра эквивалентности полям в условии WHERE 16
5.2.7 Добавление параметра больше или равно полям в условии WHERE 16
5.2.8 Добавление параметра меньше или равно полям в условии WHERE 16
5.2.9 Добавление параметра меньше полям в условии WHERE 17
5.2.10 Добавление параметра больше полям в условии WHERE 17
5.2.11 Добавление AND в условие WHERE 17
5.2.12 Добавление OR в условие WHERE 17
5.2.13 Добавление параметра LIKE полям в условии WHERE 17
5.2.14 Добавление параметра IN полям в условии WHERE 17
5.2.15 Добавление условия GROUP BY 17
5.2.16 Добавление полей к условию GROUP BY 17
5.2.17 Добавление условия ORDER BY 18
5.2.18 Добавление полей к условию ORDER BY 18
5.2.19 Добавление условия ASC к ORDER BY 18
5.2.20 Добавление условия DESC к ORDER BY 18
5.2.22 Получение строки запроса 18
5.2.23 Выполнение сформированного запроса 18
5.3 Пример формирования запросов 18

6. Добавление дополнительных полей в чек 19
6.0 Список публичных полей класса 19
6.1 Описание методов класса CheckExtraParamsBuilder 20
6.1.1 Конструктор класса 20
6.1.2 Добавление телефон оператора перевода 20
6.1.3 Добавление операции платежного агента 20

6.1.4 Добавление телефона платежного агента 20
6.1.5 Добавление телефона оператора по приему платежей 20
6.1.6 Добавление наименование оператора перевода 20
6.1.7 Добавление ИНН оператора перевода 21
6.1.8 Добавление телефона поставщика 21
6.1.9 Добавление дополнительного реквизита чека. 21
6.1.10 Добавление телефона или электронного адреса покупателя 21
6.1.11 Добавление адреса электронной почты отправителя чека 21
6.1.12 Добавление адреса оператора перевода 21
6.1.13 Добавление реквизитов пользователя 21
6.1.14 Добавление нескольких дополнительных полей 21

7. Обработка событий ККТ. 22
7.1 Получение событий об изменении чека. 22
7.2 Получение событий об изменении статуса смены. 23
7.3 Получение событий о работе с картой товара. 23

8. Команды для работы с ККТ. 24
8.1 Команды для работы с чеком. 24
8.2 Команды для работы с внутреннем принтером. 25
8.3 Команды для работы с со скидками. 27
8.4 Команды для работы с отчетами и сменой. 28

9. Работа с системами лояльности 29
9.1 Применение системы лояльности 29
9.2 Событие продажи 29
9.3 Отправка событий о бонусах 30
9.4 Получение событий о бонусах 31

1. Введение

Данный документ описывает правила взаимодействия сторонних приложений с ПО “Кассатка”, устанавливаемых на кассы “Кассатка”. Описание возможностей интеграции дано с целью создания удобного комплекса приложений для работы с данными кассовыми аппаратами.

В данной документации описаны возможности взаимодействия сторонних пользовательских приложений с основным приложением кассы с помощью разработанного и реализованного комплекса программных средств разработки (SDK).

2. Добавление библиотеки к проекту

В build.gradle проекта необходимо добавить ссылку на репозиторий jitpack:

```
allprojects {
    repositories {
        maven { url "https://jitpack.io" }
    }
}
```

в модуле `build.gradle` необходимо прописать зависимость и указать точную версию:

```
implementation 'com.github.kassatka-online:android-api:' +
apiLatestVersion
```

где, `apiLatestVersion` определена в секции ext того же build.gradle файла.

Также необходимо указать `minSdkVersion` проекта:

```
defaultConfig {
      minSdkVersion 22
      ...
}
```

Для корректной сборки проекта должны быть добавлены следующие строки в build.gradle файл модуля:

```
android {
  compileOptions {
    sourceCompatibility JavaVersion.VERSION_1_8
    targetCompatibility JavaVersion.VERSION_1_8
  }}
```

3. Работа с весами

3.1 Описание API

3.1.1 *Открытие карты товара*

`ru.kassa.action.cart.config.open` *- действие генерируемое ПО “Кассатка” при открытии карты товара при продаже. Данные не передаются.*

3.1.2 Закрытие карты товара

`ru.kassa.action.cart.config.open` *- действие генерируемое ПО “Кассатка” при закрытии карты товара при продаже. Данные не передаются.*

3.1.3 Отправка веса продукта

`ru.kassa.action.cart.config.SEND_WEIGHT` - действие, генерируемое сторонним приложением для передачи параметра веса. При генерации необходимо отправить `long` экстра-параметр с именем `PRODUCT_WEIGHT`, обозначающий вес взвешенного продукта

3.1.4 Отправка производителя весов

`ru.kassa.action.cart.config.SEND_VENDOR_NAME` - действие, генерируемое сторонним приложением для передачи названия производителя весов. При генерации необходимо отправить `String` экстра-параметр с именем *`VENDOR_NAME`*, обозначающий производителя весов.

3.1.5 Отправка модели весов

`ru.kassa.action.cart.config.SEND_MODEL_NAME` - действие генерируемое сторонним приложением для передачи названия производителя весов. При генерации необходимо отправить String экстра-параметр с именем `MODEL_NAME`, обозначающий модель весов.

3.1.6 Широковещательный приемник данных весов.

`ScalesDataBroadcastReceiver` - класс унаследованный от `BroadcastReceiver`, реализованный в подключаемой библиотеке и осуществляющий непосредственное взаимодействие с ККТ для передачи данных о весах и взвешенном товаре.

Необходимо зарегистрировать данный широковещательный приемник в манифесте вашего приложения следующим образом:

```
<receiver
   android:name="ru.kassa"
   android:exported="true">
   <intent-filter>
       <action android:name="ru.kassa.action.cart.config.SEND_WEIGHT" />
       <action
android:name="ru.kassa.action.cart.config.SEND_VENDOR_NAME" />
       <action
android:name="ru.kassa.action.cart.config.SEND_MODEL_NAME" />
   </intent-filter>
</receiver>
```

3.2 Описание работы

3.2.1 Описание

При вызове (открытии) окна с настройками количества товара в момент продажи генерируется событие `ru.kassa.action.cart.config.open (START_GET_WEIGHT)` и отсылается широковещательным запросом. При закрытии генерируется аналогичное событие `ru.kassa.action.cart.config.close (STOP_GET_WEIGHT)`.

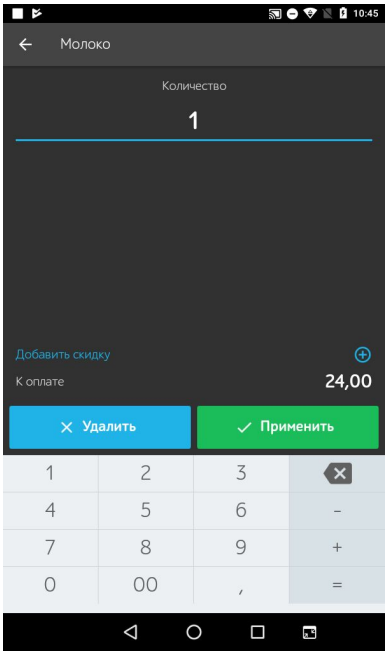

Рис. 1: Экран настроек количества товара.

Также необходимо зарегистрировать `ScalesDataBroadcastReceiver` реализованный в данном SDK, отвечающий на следующие действия (actions):

```
ru.kassa.action.cart.config.SEND_WEIGHT
ru.kassa.action.cart.config.SEND_VENDOR_NAME
ru.kassa.action.cart.config.SEND_MODEL_NAME
```

Для того чтобы реализовать корректную передачу веса в смарт-терминал достаточно реализовать BroadcastReceiver, который будет перехватывать действия `START_GET_WEIGHT` и `STOP_GET_WEIGHT` и генерировать действия для `ScalesDataBroadcastReceiver` после взвешивания товара.

3.2.2 Пример

```
public class KassatkaScalesBroadcastReceiver extends BroadcastReceiver {

public static final String START_GET_WEIGHT =
"ru.kassa.action.cart.config.open";
public static final String STOP_GET_WEIGHT =
"ru.kassa.action.cart.config.close";
private static final String SEND_WEIGHT =
"ru.kassa.action.cart.config.SEND_WEIGHT";
private static final String SEND_VENDOR_NAME =
"ru.kassa.action.cart.config.SEND_VENDOR_NAME";
private static final String SEND_MODEL_NAME =
"ru.kassa.action.cart.config.SEND_MODEL_NAME";
 @Override
 public void onReceive(Context context, Intent intent) {
   final String action = intent.getAction();
   switch (action) {
     case START_GET_WEIGHT:
       // Необходимо взвесить товар и получить
данные о производителе и модели весов.
       sendBrodcastAction(SEND_WEIGHT, "PRODUCT_WEIGHT", productWeight,
context);
       sendBrodcastAction(SEND_VENDOR_NAME, "VENDOR_NAME", vendorName,
context);
       sendBrodcastAction(SEND_MODEL_NAME, "MODEL_NAME", "THIS is MODEL
NAME", context);
       break;
     case STOP_GET_WEIGHT:
        // Можете выключить ваши весы.
     default:
       break;
```

```
    }
  }

  private void sendBrodcastAction(String Action, String extra, String
value, Context context) {
    Intent intent = new Intent(Action);
    intent.putExtra(extra, value);
    context.sendBroadcast(intent);
  }

  private void sendBrodcastAction(String action, String extra, Long
value, Context context) {
    Intent intent = new Intent(action);
    intent.putExtra(extra, value);
    context.sendBroadcast(intent);
  }
```

4. Вызов окон смарт-терминала

Класс `NavigationApi` предоставляет возможности запуска различных активностей ПО “Кассатка”.

4.1 Описание методов класса NavigationApi

4.1.1 Создание цели, вызывающей активность учета

`Intent` **`createAccountingActivityIntent()`** - метод возвращающий объект `Intent`, указывающий на активность учета ПО “Кассатка”.

4.1.2 Создание цели, вызывающей активность быстрой продажи

`Intent` **`createSaleQuickActivityIntent()`** - метод возвращающий объект `Intent`, указывающий на активность быстрой продажи ПО “Кассатка”.

4.1.3 Создание цели, вызывающей активность конфигурации товара

`Intent` **`createCartConfigActivityIntent()`** - метод возвращающий объект `Intent`, указывающий на активность конфигурации товара ПО “Кассатка”.

4.1.4 Создание цели, вызывающей активность помощи

`Intent` **`createHelpActivityIntent()`** - метод возвращающий объект `Intent`, указывающий на активность помощи ПО “Кассатка”. Перенаправление можно использовать для отправления логов ККТ.

4.1 5 Создание цели, вызывающей активность ОФД

`Intent createOfdInfoActivityIntent()` - метод возвращающий объект `Intent`, указывающий на активность ОФД ПО “Кассатка”.

4.1.6 Создание цели, вызывающей активность закрытия смены

`Intent createManualShifActivityIntent()` - метод возвращающий объект `Intent`, указывающий на активность закрытия смены ПО “Кассатка”.

4.1.7 Создание цели, вызывающей активность изъятия или внесения денежных средств в кассу

`Intent createBalanceActivityIntent(boolean isPayout)` - метод возвращающий объект `Intent`, указывающий на активность изъятия или внесения денежных средств в кассу ККТ. Параметр `isPayout` указывает на изъятие денежных средств из кассы, если он равен `true`, и на внесение - в другом случае.

4.1.8 Создание цели, вызывающей активность списка транзакций

`Intent createTransactionsActivityIntent()` - метод возвращающий объект `Intent`, указывающий на активность списка транзакций ККТ.

4.1.9 Создание цели, вызывающей активность транзакции по чеку

`Intent createTransactionsCardActivityIntent(long checkId)` - метод возвращающий объект `Intent`, указывающий на активность транзакции чека, который ищется по `id` чека - переданному параметру.

4.1.10 Создание цели, вызывающей активность смены

`Intent createShiftActivityIntent()` - метод возвращающий объект `Intent`, указывающий на активность информации о текущей смене.

4.1.11 Создание цели, вызывающей активность создания нового товара

`Intent createCreateProductActivityIntent()` - метод возвращающий объект `Intent`, указывающий на активность создания нового товара.

4.1.12 Создание цели, вызывающей активность продажи

`Intent createSaleActivityIntent()` - метод возвращающий объект `Intent`, указывающий на активность продажи.

4.1.13 Создание цели, вызывающей активность перехода к оплате

`Intent createCartActivityIntent()` - метод возвращающий объект `Intent`, указывающий на активность перехода к оплате.

4.1.14 Создание цели, вызывающей активность оплаты наличностью

`Intent createPaymentCashActivityIntent()` - метод возвращающий объект `Intent`, указывающий на активность оплаты наличностью.

4.1.15 Создание цели, вызывающей активность скидок

`Intent createDiscountActivityIntent()` - метод возвращающий объект `Intent`, указывающий на активность скидок.

4.1.16 Создание цели, вызывающей активность отправки чека

`Intent createSendCheckActivityIntent()` - метод возвращающий объект `Intent`, указывающий на активность отправки чека.

4.1.17 Создание цели, вызывающей активность чека коррекции.

`Intent createCorrectionActivityIntent()` - метод возвращающий объект `Intent`, указывающий на активность чека коррекции.

4.1.18 Создание цели, вызывающей активность добавления клиента

`Intent createAddClientActivityIntent()` - метод возвращающий объект `Intent`, указывающий на активность добавления клиента.

4.1.19 Вызов нового окна

`void startActivity(Context context, Intent intent)` - метод запускающий активность из контекста `context` по заданной цели `intent`.

4.2 Пример вызова нового окна

```
NavigationApi.startActivity(
getApplicationContext(),
NavigationApi.createBalanceActivityIntent(true));
```

5. Получение данных из БД смарт-терминала

Для получения данных из БД смарт-терминала был реализован класс `QueryBuilder`, который позволяет сформировать и отправить запрос к БД и вернуть объект `Cursor` через интерфейс `QueryBuilderListener`.

5.1 Описание доступных таблиц БД смарт-терминала

5.1.1 Таблица категорий товара

Для реализации доступа к колонкам таблицы необходимо воспользоваться публичным перечислением (enum) CategoryModelField, содержащем следующие колонки таблицы:

- _id
- EXT_ID
- NAME
- COLOR
- ORDER_BY
- SHOW_ON_TABLET
- PRIORITY
- ALL

5.1.2 Таблица чеков

Для реализации доступа к колонкам таблицы необходимо воспользоваться публичным перечислением (enum) CheckModelField, содержащем следующие колонки таблицы:

- _ORG_NAME,
- EGAIS_CHECK_URL,
- EGAIS_SIGN,
- FISCAL_PRINT_TEXT,
- FISCAL_QR,
- ALL

5.1.3 Таблица позиций чеков

Для реализации доступа к колонкам таблицы необходимо воспользоваться публичным перечислением (enum) CheckPositionModelField, содержащем следующие колонки таблицы:

- _id,
- CHECK_ID,
- PRODUCT_EXT_ID,
- PRODUCT_NAME,
- CATEGORY_EXT_ID,
- PRODUCT_PRICE,
- COUNT,
- POSITION_PRICE_WITHOUT_DISCOUNT,
- POSITION_PRICE,
- VAT_TYPE_id
- CHECK_ID,

- DATE,
- CHECK_PRINT_NUM,
- FISCAL_EXT_ID,
- TERMINAL_EXT_ID,
- CLIENT_PHONE,
- CLIENT_EMAIL,
- RELATION_CHECK_ID
- RELATION_FISCAL_EXT_ID,
- RELATION_TERMINAL_EXT_ID,
- RELATION_FISCAL_DATE,
- ORGANISATION_NAME,
- ORGANISATION_ADDRESS,
- ORGANISATION_INN,
- STORE_ADDRESS,
- STATE,
- IS_ACCEPTED,
- DOC_TYPE,
- DOC_TYPE_TITLE,
- WORKMAN_FIO,
- WORKMAN_POSITION,
- WORKMAN_INN,
- TOTAL_PRICE,
- TOTAL_PRICE_WITHOUT_DISCOUNT,
- TOTAL_DISCOUNT,
- TOTAL_VAT,
- IS_SIGNATURE,
- IS_CLIENT_SIGNATURE,
- IS_DEMO,
- PAY_TYPE,
- IS_OPEN_CASH_DRAWER,
- CHANGE,
- MONEY_FROM_BUYER,
- COMMENT,
- USER_EXT_ID,
- CASH_REMAINS_MANUAL,
- BANK_REMAINS_MANUAL,
- TO_STORE_ID,
- TO_STORE_NAME,
- CORRECTION_DOC_TYPE,
- CORRECTION_TYPE,
- BONUSES,
- CARD_NUM,
- TAX_MODE,
- KASSA_NUM,
- EGAIS_INN,
- EGAIS_KPP,

- EGAIS,
- VAT_SUM,
- DISCOUNT_EXT_ID,
- DISCOUNT_NAME,
- DISCOUNT_TYPE,
- AGENT_TYPE,
- PAY_ATTRIBUTES_TYPE,
- GOOD_ATTRIBUTES_TYPE,
- DISCOUNT_VALUE,
- EGAIS_CODE,
- BAR_CODE,
- PRODUCT_TYPE,
- ALCOHOL_BY_VOLUME,
- ALCOHOL_PRODUCT_KIND_CODE,
- TARE_VOLUME,
- ALL

5.1.4 Таблица скидок

Для реализации доступа к колонкам таблицы необходимо воспользоваться публичным перечислением (enum) DiscountModelField, содержащем следующие колонки таблицы:

- _id,
- EXT_ID,
- NAME,
- TYPE,
- UNIT,
- UNIT_TITLE,
- VALUE,
- IN_ALL_CATEGORIES,
- DOW_ID,
- TIME_INTERVAL_ID,
- CATEGORIES_IDS

5.1.5 Таблица записей в журнале

Для реализации доступа к колонкам таблицы необходимо воспользоваться публичным перечислением (enum) LogEntryModelField, содержащем следующие колонки таблицы:

- _id,
- DEVICE_TIMESTAMP,
- DATA,
- ALL

5.1.6 Таблица карт лояльности

Для реализации доступа к колонкам таблицы необходимо воспользоваться публичным перечислением (enum) LoyaltySaleModelField, содержащем следующие колонки таблицы:

- _id,
- DATA,
- CARD_NUM,
- ALL

5.1.7 Таблица товаров

Для реализации доступа к колонкам таблицы необходимо воспользоваться публичным перечислением (enum) ProductModelField, содержащем следующие колонки таблицы:

- _id,
- EXT_ID,
- BARCODE,
- NAME,
- GROUP,
- COLOR,
- PRIORITY,
- FREE_PRICE,
- FREE_SALE,
- VAT_TYPE,
- PRICE,
- UNIT,
- AGENT_TYPE,
- GOOD_ATTRIBUTES_TYPE,
- IS_FAVOURITE,
- IS_REMAINDERS_CONTROL,
- CURRENT_REMAINDERS,
- IS_HIDE_ON_SALE,
- ALCOHOL_BY_VOLUME,
- ALCOHOL_PRODUCT_KIND_CODE,
- TARE_VOLUME,
- TAX_MODE,
- PRODUCT_TYPE,
- ALL

5.1.8 Таблица RabbitMessage

Для реализации доступа к колонкам таблицы необходимо воспользоваться публичным перечислением (enum) RabbitMessageModelField, содержащем следующие колонки таблицы:

- _id,
- ROUTING_KEY,
- MESSAGE,
- ALL

5.1.9 Таблица склада

Для реализации доступа к колонкам таблицы необходимо воспользоваться публичным перечислением (enum) StoreModelField, содержащем следующие колонки таблицы:

- _id,
- EXT_ID,
- NAME,
- ADDRESS,
- ALL

5.1.10 Таблица пользователя

Для реализации доступа к колонкам таблицы необходимо воспользоваться публичным перечислением (enum) UserModelField, содержащем следующие колонки таблицы:

- _id,
- NAME,
- EXT_ID,
- ACCOUNT_ID,
- PIN,
- TABLET_HIDE_XREPORT,
- ROLE,
- INN,
- ALL

5.2 Описание методов класса QueryBuilder

5.2.1 Конструктор по умолчанию

QueryBuilder() - конструктор, начинающий формирование SELECT оператора, для таблицы PRODUCT_MODEL.

5.2.2 Конструктор с параметрами

`QueryBuilder(QueryBuilderListener queryBuilderListener, String table)` - конструктор, начинающий формирование `SELECT` оператора для таблицы `table`, который переопределяет параметр queryBuilderListener содержащийся в классе `QueryBuilder`.

5.2.3 Выбор колонок таблицы

`QueryBuilder<T> withFields(ArrayList<T> checkModelFields)` - метод возвращающий тот же объект, но с заполненными наименованиями колонок таблицы, которые необходимы для запроса `SELECT`. `checkModelFields` - массив перечислений указанных в разделе 4.1.

5.2.4 Добавление условия WHERE

`QueryBuilder<T> where()` - метод возвращающий тот же объект, но с дополнительной заготовкой для добавления условий фильтрации запроса к БД ККТ.

5.2.5 Добавление полей к условию WHERE

`QueryBuilder<T> field(T checkModelField)` - метод возвращающий тот же объект, но с указанием поля, которое добавляется в условие `WHERE` запроса к БД смарт-терминала.

5.2.6 Добавление параметра эквивалентности полям в условии WHERE

`QueryBuilder<T> equal(String value)` - метод возвращающий тот же объект, но добавляющий к WHERE условие эквивалентности `value` последнему полю. Если такого поля нет, или последней операцией было добавлено не оно будет вызвано `RuntimeException`.

5.2.7 Добавление параметра больше или равно полям в условии WHERE

`QueryBuilder<T> moreEqual(String value)` - метод возвращающий тот же объект, но добавляющий к `WHERE` условие ‘последнее поле больше или равно `value`’. Если такого поля нет, или последней операцией было добавлено не оно будет вызвано `RuntimeException`.

5.2.8 Добавление параметра меньше или равно полям в условии WHERE

`QueryBuilder<T> lessEqual(String value)` - метод возвращающий тот же объект, но добавляющий к `WHERE` условие ‘последнее поле меньше или равно value’.

Если такого поля нет, или последней операцией было добавлено не оно будет вызвано `RuntimeException`.

5.2.9 Добавление параметра меньше полям в условии WHERE

`QueryBuilder<T> less(String value)` - метод возвращающий тот же объект, но добавляющий к `WHERE` условие 'последнее поле меньше `value`'. Если такого поля нет, или последней операцией было добавлено не оно будет вызвано `RuntimeException`.

5.2.10 Добавление параметра больше полям в условии WHERE

`QueryBuilder<T> more(String value)` - метод возвращающий тот же объект, но добавляющий к WHERE условие 'последнее поле больше `value`'. Если такого поля нет, или последней операцией было добавлено не оно будет вызвано `RuntimeException`.

5.2.11 Добавление AND в условие WHERE

`QueryBuilder<T> and()` - метод возвращающий тот же объект, но добавляющий к `WHERE` условие `AND`.

5.2.12 Добавление OR в условие WHERE

`QueryBuilder<T> or()` - метод возвращающий тот же объект, но добавляющий к `WHERE` условие `OR`.

5.2.13 Добавление параметра LIKE полям в условии WHERE

`QueryBuilder<T> like(String value)` - метод возвращающий тот же объект, но добавляющий к `WHERE` условие: 'последнее поле `like value`'. Если такого поля нет, или последней операцией было добавлено не оно будет вызвано `RuntimeException`.

5.2.14 Добавление параметра IN полям в условии WHERE

`QueryBuilder<T> like(ArrayList<String> values)` - метод возвращающий тот же объект, но добавляющий к `WHERE` условие: 'последнее поле `in values`'. Если такого поля нет, или последней операцией было добавлено не оно будет вызвано `RuntimeException`.

5.2.15 Добавление условия GROUP BY

`QueryBuilder<T> groupBy()` - метод возвращающий тот же объект, но с дополнительной заготовкой для добавления условий фильтрации запроса к БД ККТ.

5.2.16 Добавление полей к условию GROUP BY

`QueryBuilder<T> groupField(T checkModelField)` - метод возвращающий тот

же объект, но с указанием поля, которое добавляется в условие `GROUP BY` запроса к БД смарт-терминала.

5.2.17 Добавление условия ORDER BY

`QueryBuilder<T> orderBy()` - метод возвращающий тот же объект, но с дополнительной заготовкой для добавления условий фильтрации запроса к БД ККТ.

5.2.18 Добавление полей к условию ORDER BY

`QueryBuilder<T> groupField(T checkModelField)` - метод возвращающий тот же объект, но с указанием поля, которое добавляется в условие `ORDER BY` запроса к БД смарт-терминала.

5.2.19 Добавление условия ASC к ORDER BY

`QueryBuilder<T> asc()` - метод возвращающий тот же объект, но с указанием метода `ASC` сортировки строк при выдаче ответа на формируемый запрос к БД терминала

5.2.20 Добавление условия DESC к ORDER BY

`QueryBuilder<T> desc()` - метод возвращающий тот же объект, но с указанием метода `DESC` сортировки строк при выдаче ответа на формируемый запрос к БД терминала

5.2.22 Получение строки запроса

`String getQuery()` - метод возвращающий строку запроса к БД смарт-терминала

5.2.23 Выполнение сформированного запроса

`void executeSelectQuery(Context context)` - метод отправляющий через `ContentResolver` сформированный запрос к БД ККТ и возвращающий результат в метод объекта интерфейса `QueryBuilderListener`, переданный в конструкторе.

5.3 Пример формирования запросов

```
public class BuilderFragment extends Fragment implements
QueryBuilder.QueryBuilderListener {
public View onCreateView(LayoutInflater inflater, ViewGroup container,
Bundle savenInstanceState) {
...
new QueryBuilder<ProductModelField>(this, ""PRODUCT_MODEL")
.withFields(
new ArrayList<ProductMOdelField>() {
{
```

```
add(ProductModelField.NAME);
add(ProductModelField.BARCODE);
add(ProductModelField.PRICE);
}
})
.where()
.field(ProductModelField.NAME).equal("М о л о к о")
.and()
.field(ProductModelField.BARCODE).like("%112")
.groupBy().groupField(ProductModelField.BARCODE)
.orderBy()
.orderField(ProductModelField.PRICE)
.asc().executeSelectQuery(getApplicationContext());
...
}

public void displaySelectedFields(Cursor cursor) {
if (cursor == null) return;
if (cursor.getCount() == 0) return;
cursor.moveToFirst();
productList = new ArrayList<>();
do {
...
} while(cursor.moveToNext());
...
}
}
```

6. Добавление дополнительных полей в чек

Для добавления дополнительных полей реализован класс `CheckExtraParamsBuilder`. Данный класс отправляет широковещательные запросы к приложению “Кассатка”, которое добавляет дополнительные поля в чек.

6.0 Список публичных полей класса

- `String ADD_PHONE_OF_TRANSFER_OPERATOR_ACTION`
- `String ADD_OPERATION_OF_PAYMENT_AGENT_ACTION`
- `String ADD_PHONE_OF_PAYMENT_AGENT_ACTION`
- `String ADD_PHONE_OF_GET_PAYMENT_AGENT_ACTION`
- `String ADD_NAME_OF_TRANSFER_OPERATOR_ACTION`
- `String ADD_INN_OF_TRANSFER_OPERATOR_ACTION`
- `String ADD_PHONE_OF_PROVIDER_ACTION`
- `String ADD_ADDITIONAL_DOCUMENT_REQUISITE_ACTION`

- `String ADD_PHONE_OR_EMAIL_OF_CUSTOMER_ACTION`
- `String ADD_EMAIL_OF_DEVICE_USER_ACTION`
- `String ADD_ADDRESS_OF_TRANSFER_OPERATOR`
- `String ADD_USER_REQUISITE`

6.1 Описание методов класса CheckExtraParamsBuilder

6.1.1 Конструктор класса

`CheckExtraParamsBuilder(Context context)` - конструктор, запоминающий контекст, для дальнейшей отправки широковещательных сообщений.

6.1.2 Добавление телефон оператора перевода

`void addPhoneOfTransferOperator(String value)` - метод, вызываемый для отправки широковещательного сообщения приложению “Кассатка”, в котором передается телефон оператора перевода.

6.1.3 Добавление операции платежного агента

`void addOperationOfTransferAgent(String value)` - метод, вызываемый для отправки широковещательного сообщения приложению “Кассатка”, в котором передается операция платежного агента.

6.1.4 Добавление телефона платежного агента

`void addPhoneOfPaymentAgent(String value)` - метод, вызываемый для отправки широковещательного сообщения приложению “Кассатка”, в котором передается телефон платежного агента.

6.1.5 Добавление телефона оператора по приему платежей

`void addPhoneOfGetPaymentOperator(String value)` - метод, вызываемый для отправки широковещательного сообщения приложению “Кассатка”, в котором передается телефон оператора по приему платежей.

6.1.6 Добавление наименование оператора перевода

`void addNameOfTransferOperator(String value)` - метод, вызываемый для отправки широковещательного сообщения приложению “Кассатка”, в котором передается наименование оператора перевода.

6.1.7 Добавление ИНН оператора перевода

`void addInnOfTransferOperator(String value)` - метод, вызываемый для отправки широковещательного сообщения приложению “Кассатка”, в котором передается ИНН оператора перевода.

6.1.8 Добавление телефона поставщика

`void addPhoneOfProvider(String value)` - метод, вызываемый для отправки широковещательного сообщения приложению “Кассатка”, в котором передается телефон поставщика.

6.1.9 Добавление дополнительного реквизита чека.

`void addAdditionalDocumentRequisite(String value)` - метод, вызываемый для отправки широковещательного сообщения приложению “Кассатка”, в котором передается дополнительный реквизит чека.

6.1.10 Добавление телефона или электронного адреса покупателя

`void addPhoneOrEmailOfCustomer(String value)` - метод, вызываемый для отправки широковещательного сообщения приложению “Кассатка”, в котором передается телефон или электронный адрес покупателя.

6.1.11 Добавление адреса электронной почты отправителя чека

`void addEmailOfDeviceUser(String value)` - метод, вызываемый для отправки широковещательного сообщения приложению “Кассатка”, в котором передается адрес электронной почты отправителя чека.

6.1.12 Добавление адреса оператора перевода

`void addAddressOfTransferOperator(String value)` - метод, вызываемый для отправки широковещательного сообщения приложению “Кассатка”, в котором передается адрес оператора перевода.

6.1.13 Добавление реквизитов пользователя

`void addUserRequisite(String value, String title)` - метод, вызываемый для отправки широковещательного сообщения приложению “Кассатка”, в котором передается `value` - значение реквизитов пользователя и `title` - заголовок реквизитов пользователя.

6.1.14 Добавление нескольких дополнительных полей

`void addExtraParamsList(List<Pair<String, String> > extraParams)` - метод вызываемый для последовательной отправки нескольких широковещательных сообщений приложению “Кассатка”, в который передается список пар. Левое значение

пары - одно из публичных полей класса `CheckExtraParamsBuilder` описанных в разделе 5.0, второе - передаваемое значение.

7. Обработка событий ККТ.

Широковещательные сообщения не требуют ответа. Достаточно зарегистрировать приемник сообщений (`BroadcastReceiver`) для получения событий ККТ.

Для получения событий объявите в манифесте свой `BroadcastReceiver`.

```
<receiver
        Название вашего класса
        android:name=".OpenCheckBroadcastReceiver"
        android:enabled="true"
        android:exported="true">
        <intent-filter>
            Укажите события на который хотите
подписаться
            <action android:name="ru.kassa.action.open.doc.sale " />
            <category android:name="android.intent.category.DEFAULT" />
        </intent-filter>
    </receiver>
```

7.1 Получение событий об изменении чека.

7.1.1 Открыт документ продажи

При создании чека продажи отсылается интент `ru.kassa.action.open.doc.sale` содержащий в extra `"CheckId"` тип `long(uint 32)`

7.1.2 Открыт документ возврата

При создании чека возврата отсылается интент `ru.kassa.action.open.doc.return` содержащий в extra `"CheckId"` тип `long(uint 32)`

7.1.3 В чек добавлена позиция

При добавлении позиции в чек отсылается интент `ru.kassa.action.position.add` содержащий в extra `"UUID"` (universally unique identifier).

Пример обработки события:

```
public class PositionBroadcastReceiver extends BroadcastReceiver {
     @Override
     public void onReceive(Context context, Intent intent){
             if(intent.getAction().equals(
ru.kassa.action.position.add){
                 UUID uuid= intent.getExtras().get("UUID ")
             }
     }
}
```

7.1.4 Обновление позиции чека.

При обновлении позиции в чеке посылается интент `ru.kassa.action.position.update` содержащий в extra `"UUID"` и `"POSITION"` - id номенклатуры позиции для получении из базы.

7.1.5 Сохранения чека в базу.

При сохранении чека посылается интент `ru.kassa.action.check.save` содержащий в extra `"CheckId"` тип `long(uint 32)`.

7.1.6 Ошибка при печати чека

Если чек не удалось напечатать отправляется сообщение `ru.kassa.action.check.print.error` содержащие в extra `"CheckId"` и `"ErrorMessage"` сообщение об ошибке тип `String`.

7.1.7 Закрытие чека(Успешная печать)

Если чек успешно напечатан отправляется сообщение `ru.kassa.action.check.close` содержащие в extra `"CheckId"` и `"DOC_TYPE"` (значение `"SALE"` или `"RETURN"`).

7.1.8 Событие обновления номенклатуры

Если номенклатура синхронизирована с ЛК, то отправляется `ru.kassa.action.inventory.update`

7.2 Получение событий об изменении статуса смены.

7.2.1 Сообщение при закрытии смены

Если напечатан чек о закрытии смены отправляется сообщение `ru.kassa.action.check.shift.close.`

7.3 Получение событий о работе с картой товара.

7.3.1 Сообщение о открытии карты

Если открыта карта с товаром отправляется сообщение `ru.kassa.action.cart.config.open`

7.3.2 Сообщение о закрытии карты

Если карта с товаром закрыта отправляется сообщение `ru.kassa.action.cart.config.close`

8. Команды для работы с ККТ.

Для событий, которые требуют ответа от торгового приложения нужно использовать команды описанные ниже. Для реализации команды достаточно вставить в свой код приведенные примеры реализации. Команды отличаются друг от друга только параметрами передаваемые в класс `Extra`, который необходим для того, чтобы сформировать `JSON` для отправки на КАССАТКУ.

Для работы с данным API необходимо подключить библиотеку для работы с `Gson`. Для этого в `build.gradle` модуля добавьте следующую зависимость:

```
implementation 'com.google.code.gson:gson:2.8.5'
```

8.1 Команды для работы с чеком.

Для выполнения всех команд, связанных с работой чека необходимо проинициализировать класс `CheckCommand`:

```
CheckCommand command = new CheckCommand ();
```

Для каждой команды имеется свой набор параметров передаваемых в класс `Extra`.

8.1.1 Команда создания чека.

Пример кода для создания чека.

```
CheckCommand command = new CheckCommand ();
Extra extra = new Extra();
```

В "ACTION" передается названия события.

```
extra.putExtra("ACTION", "CREATE_CHECK");
```

Указываем тип документа "SALE" или "RETURN".

```
extra.putExtra("DOC_TYPE", "SALE/RETURN");
```

При создании чека можно указать реквизиты организации

```
extra.putExtra("DOC_TYPE", "SALE/RETURN");
```

Необязательные поля для заголовка чека, если не указаны, то беруться данные полученные при регистрации ККТ.

```
extra.putExtra("ORGANISATION_ADDRESS", "Адрес
 организации");
extra.putExtra("ORGANISATION_NAME", "Имя организации");
extra.putExtra("ORGANISATION_INN", "ИНН организации");
```

Для добавления `extra` к Intent нужно выполнить, следующие команды.

```
command.openCheckCommand(extra);
command.startCommand(context, callBack);
```

В `callBack` возвращается `boolean`.

8.1.2 Команда для добавления позиций в чек.

```
Extra extra = new Extra();
extra.putExtra("ACTION", "ADD_ITEM");
```

`product` - модель товара содержит `id`, наименование, цену, тип налогообложения.

```
command.addItemsCommand(product, extra).startCommand(context, callBack);
```

В `callBack` возвращается `UUID` продукта и `boolean`.

8.1.3 Команда для удаления позиции из чека.

```
Extra extra = new Extra();
extra.putExtra("ACTION", "REMOVE_ITEM");
```

`product` - модель товара, содержащий `UUID` товара, который нужно удалить из чека.

```
command.removeItemsCommand(product, extra).startCommand(context,
callBack);
```

8.1.4 Закрытие чека(Печать чека).

```
Extra extra = new Extra();
extra.putExtra("ACTION", "CLOSE_CHECK");
extra.putExtra("CASH",(double) 1000);// Стоимость всех
товаров в чеке.
command.closeCheckCommand(extra).startCommand(context, callBack);
```

8.1.5 Указать систему налогообложения.

```
Extra extra = new Extra();
extra.putExtra("ACTION", "CLOSE_CHECK");
extra.putExtra("TAX_MODE", (byte) 1);
command.closeCheckCommand(extra).startCommand(context, callBack);
```

8.1.6 Закрытие чека(Печать чека).

```
Extra extra = new Extra();
extra.putExtra("ACTION", "CLOSE_CHECK");
extra.putExtra("CASH",(double) 1000);
command.closeCheckCommand(extra).startCommand(context, callBack);
```

8.2 Команды для работы с внутреннем принтером.

8.2.1 Печать строк

Для печати строки на внутреннем принтере, нужно реализовать в вашем классе следующую команду.

```
PrintCommand printCommand = new PrintCommand();
Extra extra = new Extra();
extra.putExtra("ACTION", "PRINT_STRING");
extra.putExtra("STRING", "Some text...");
printCommand.printEmtyString(extra);
printCommand.startCommand(context, new CallBackPrint() {
@Override
public void result(Boolean isPrint) {
}
});
```

В `callBack` возвращается ответ напечатан текст или нет.

Пример использования Для печати рекламного текста, после печати документа продажи или возврата.

Приложения должно уметь обрабатывать событие закрытия чека, смотреть выше.

8.2.2 Печать пустых строк

```
PrintCommand printCommand = new PrintCommand();
Extra extra = new Extra();
extra.putExtra("ACTION", "PRINT_EMPTY");
extra.putExtra("COUNT", 5); Количество строк
printCommand.printEmtyString(extra);
printCommand.startCommand(MainActivity.this, new CallBackPrint() {
@Override
public void result(Boolean isPrint) {
}
});
```

В callBack возвращается ответ напечатан текст или нет.

8.2.3 Печать EAN13 баркода

```
PrintCommand printCommand = new PrintCommand();
Extra extra = new Extra();
extra.putExtra("ACTION", "EAN13");
extra.putExtra("BARCODE", "2727587587578"); должен содержать
12-13 символов
printCommand.printEmtyString(extra);
printCommand.startCommand(MainActivity.this, new CallBackPrint() {
@Override
```

```
public void result(Boolean isPrint) {
}
});
```

В `callBack` возвращается ответ напечатан баркод нет.

8.3 Команды для работы с со скидками.

Для работы со скидками приложение должно в манифесте содержать активити подписанное на событие "ru.kassa.add.discount". По нажатию на кнопку добавить скидку будет отправляться intent "ru.kassa.add.discount" для запуска одного из приложения работающего со скидками. Пользователь сам выберет одно из предложенных приложений.

8.3.1 Начисление скидки на весь чек

```
Extra extra = new Extra();
extra.putExtra("ACTION", "DISCOUNT_CHECK");
extra.putExtra("NAME", "Скидка в др"); Имя скидки
extra.putExtra("Value", "10");
extra.putExtra("DISCOUNT_TYPE", "PERCENT/MONEY");// Тип скидки: процент от суммы или фиксированная сумма.
```

```
DiscountCommand command = new DiscountCommand();
command.setDiscountAllCheck(extra).startCommand(context, callBack);
```

В `callBack` возвращается:

- `true` - скидка установлена;
- `false` - скидка не установлена;
- `Error` - чек закрыт.

8.3.2 Начисление скидки на определенную позицию

Класс `Position` содержит в себе `UUID` (universally unique identifier) позиции `id` товара.

```
Position position = new Position(intent);
Extra extra = new Extra();
extra.putExtra("ACTION", "DISCOUNT_POSITION");
extra.putExtra("NAME", "Скидка на колбасу");
extra.putExtra("Value", "10");
extra.putExtra("DISCOUNT_TYPE", "PERCENT/MONEY");
DiscountCommand command = new DiscountCommand();
command.setDiscountPosition(position.getUuid(), extra);
command.startCommand(context, new DiscountCallBack() {
    @Override
    public void OnSuccess(Boolean isSetDiscount) {
    }
     @Override
```

```
      public void OnError(String msg) {
      }
    });
}
```

8.4 Команды для работы с отчетами и сменой.

8.4.1 Получение статуса и номер смены

```
ShiftCommand checkCommand = new ShiftCommand();
Extra extra = new Extra();
extra.putExtra("ACTION", "SHIFT_STATUS");
checkCommand.getStatus(extra);
checkCommand.startCommand(MainActivity.this, new ShiftCallBack() {
       @Override
       public void OnSuccess(Long shiftNum, Boolean shiftIsOpen) {
       }
       @Override
       public void OnError(String message) {
        }
      });
});
```

В `callBack` возвращается:

- `Long` – номер кассовой смены.
- `true` – смена открыта.
- `false` – смена закрыта.
- `Error` - не удалось определить состояние текущей кассовой смены

8.4.2 Печать X отчета

```
ShiftCommand checkCommand = new ShiftCommand();
Extra extra = new Extra();
extra.putExtra("ACTION", "X_REPORT");
checkCommand.PrintXReport(extra);
checkCommand.startCommand(MainActivity.this, new CallBackPrint() {
       @Override
        public void result(Boolean isPrint) {
        }
});
```

В `callBack` возвращается `boolean` - удалось напечатать отчет или нет.

9. Работа с системами лояльности

Для того чтобы посмотреть доступные смарт-терминалу “Кассатка ”системы лояльности можно использовать `enum LoayltySystem` из библиотеки интеграции.

9.1 Применение системы лояльности

Применить систему лояльности можно с помощью объекта класса `LoyaltyApplier`. Для этого можно воспользоваться следующим методом:

`void applyLoyalty(Context context, T card, LoyaltySystem loyaltySystem)` - метод отправляющий событие применения системы лояльности для карты card и системы лояльности `loyaltySystem` приложению Кассатка.

Для использования этого метода необходимо заполнить класс `BaseLoyaltyCard`, содержащий следующие поля в формате используемой системы лояльности:

- `String number` - номер карты в виде строки;
- `String client` - информация о клиенте;
- `String phone` - телефон клиента;
- `double balance` - баланс карты;
- `double isActive` - информация об активности карты;
- `double cardBonus` - бонусы, зачисленные на карту;
- `double cardDiscount` - скидка, применяемая к карте;
- `double discountName` - название скидки, применяемой к карте.

Пример:

```
void setLoyaltyCard(BaseLoyaltyCard card, Context context) {
      LoyaltyApplier<BaseLoyaltyCard> applier = new LoyaltyApplier<>();
      applier.applyLoyalty(context, card, LoyaltySystem.FOUR_CLIENT);
}
```

9.2 Событие продажи

Для получения информации о продаже в библиотеке интеграции реализован класс `SaleActionReceiver`.

Для регистрации данного ресивера необходимо создать объект с помощью следующего конструктора:

```
SaleActionReceiver(Listener listener)
```

, где `listener` - объект интерфейса для обработки непосредственно в пользовательском приложении, содержащий следующий метод:

`void saleActionReceived(String cardNumber, SaleModel saleModel)` - метод реализуемый пользователем для применения события продажи описанного в `saleModel` к карте с номером `cardNumber`.

Далее `SaleActionReceiver` регистрируется таким же образом как BroadcastReceiver.

`SaleModel` содержит следущие поля в формате используемой системы лояльности:

- `String store` - наименование магазина;
- `double saleSum` - сумма чека;
- `double moneySum` - сумма внесенная деньгами;
- `double bonusSum` - сумма внесенная бонусами;
- `Date datetime` - дата и время;
- `DeviceStatus additionalInfo` - дополнительная информация об устройстве.

9.3 Отправка событий о бонусах

Для отправки событий о действиях с бонусами реализован класс `LoyaltyBonusesActions`, в котором каждая функция реализована в двух видах - статическом и объектом:

- `static void sendPrice(Context context, String cardNumber, BigDecimal totalPrice)` - функция отправки события содержащего сумму чека `totalPrice` для карту с номером `cardNumber`.
- `static void sendBonusesWriteOff(Context context, String cardNumber, BigDecimal bonuses)` - функция отправки события о списании бонусов (`totalPrice`) с карты с номером `cardNumber`.
- `static void sendAllBonusesWriteOff(Context context, String cardNumber)` - функция отправки события о списании всех бонусов с карты `cardNumber`.
- `void sendPrice(BigDecimal totalPrice)` - функция отправки события содержащего сумму чека (`totalPrice`).
- `void sendBonusesWriteOff(BigDecimal bonuses)` - функция отправки события о списании бонусов (`bonuses`).
- `void sendAllBonusesWriteOff()` - функция отправки события о списании всех бонусов

Для использования трех последних функций создайте объект следующим конструктором:

`LoyaltyBonusesActions(Context context, String cardNumber)` - конструктор сохраняющий номер карты клиента для дальнейших действий.

При использовании данных функций отправляются события в класс `LoyaltyBonusesReceiver`.

9.4 Получение событий о бонусах

Для получения событий о бонусах необходимо зарегистрировать класс `LoyaltyBonusesReceiver`.

Для того, чтобы обработать события приходящие из класса `LoyaltyBonusesActions` используется интерфейс `Listener` класса `LoyaltyBonusesReceiver`, в котором реализованы следующие функции:

- `void receivedTotalPrice(String cardNumber, BigDecimal totalPrice)` - получена цена чека `totalPrice` для карты `cardNumber`.
- `void receivedBonusesWriteOffAction(String cardNumber, BigDecimal bonuses)` - получено событие списания бонусов в размере `bonuses` с карты `cardNumber`
- `void receivedAllBonusesWriteOffAction(String cardNumber)` - получено событие списывания всех бонусов с карты `cardNumber`.

  Для регистрации данного ресивера используйте следующие action'ы:

  - `ru.kassatka.comepay_sdk.loyalty.action.send.total_price`
  - `ru.kassatka.comepay_sdk.loyalty.action.write_off.bonuses`
  - `ru.kassatka.comepay_sdk.loyalty.action.write_off.all_bonuses`