

TINKOFF ACQUIRING SDK FOR ANDROID

08.07.2021



ОГЛАВЛЕНИЕ

ИСТОРИЯ ИЗМЕНЕНИЙ	3
ТЕРМИНЫ И СОКРАЩЕНИЯ	4
ОСНОВНЫЕ ПОЛОЖЕНИЯ	5
1. ПОДГОТОВКА К ИСПОЛЬЗОВАНИЮ	6
1.1. ТРЕБОВАНИЯ И ОГРАНИЧЕНИЯ	6
1.2. ПОДКЛЮЧЕНИЕ	6
1.3. ПОДГОТОВКА К РАБОТЕ	6
1.4. НАСТРОЙКА РЕЖИМА РАБОТЫ	9
2. ПРОВЕДЕНИЕ ОПЛАТЫ	10
3. ПОДКЛЮЧЕНИЕ GOOGLE PAY	16
4. ИНТЕГРАЦИЯ С ОНЛАЙН-КАССАМИ	19
5. РЕКУРРЕНТНЫЙ ПЛАТЕЖ С ПРИВЯЗАННОЙ КАРТЫ	20
6. ПРИЕМ ПЛАТЕЖЕЙ ЧЕРЕЗ СИСТЕМУ БЫСТРЫХ ПЛАТЕЖЕЙ	21
7. ОПЛАТА ИЗ УВЕДОМЛЕНИЯ	29
8. ПРИВЯЗКА КАРТ	31
9. ДОПОЛНИТЕЛЬНЫЕ ВОЗМОЖНОСТИ	35
10. СТРУКТУРА	41
11. МЕТОДЫ API	44
12. КОДЫ ОШИБОК API И ВОЗМОЖНЫЕ ИСКЛЮЧЕНИЯ	69
13. ПОДДЕРЖКА	69

История изменений

Версия	Описание	Дата
1.0	Первоначальное составление документа	23.04.2020
1.1	Добавлено описание новых функций для оплаты из уведомления. Актуализированы скриншоты экранов SDK	27.10.2020
1.2	Добавлен новый параметр настроек экрана оплаты	13.11.2020
1.3	Добавлено описание новых параметров экранов SDK, описание новой функции оплаты через Систему быстрых платежей	21.12.2020
1.4	Добавлено описание экрана с динамическим QR кодом	09.02.2021
1.5	Добавлено описание виджета выбора банка для оплаты через Систему быстрых платежей	30.03.2021

Термины и сокращения

Термин	Определение
Продавец	Участник, принимающий через интернет в свою пользу платежи по банковским картам за товары и услуги через протокол Merchant API
Клиент	Участник, производящий оплату с использованием банковской карты на сайте продавца
Терминал	Точка приема платежей продавца (в общем случае привязывается к сайту, на котором осуществляется прием платежей)
3D Secure	Протокол, который используется как дополнительный уровень безопасности при осуществлении онлайн-платежей с банковских карт. 3D Secure добавляет ещё один шаг аутентификации при совершении онлайн-платежей
PCI DSS	Payment Card Industry Data Security Standard, стандарт безопасности данных индустрии платежных карт. Стандарт утверждён международными платежными системами Visa и MasterCard, American Express, JCB и Discover. Стандарт представляет собой совокупность 12 детализированных требований по обеспечению безопасности данных о держателях платёжных карт, которые передаются, хранятся и обрабатываются в информационных инфраструктурах организаций
Интернет-эквайринг	Технология приёма оплаты в интернете через платёжную страницу банка-эквайера. С помощью интернет-эквайринга покупатели оплачивают покупки на сайтах, в мобильных приложениях, по прямым ссылкам на страницу с платёжной формой банка
Оплата	Операция в магазине с целью покупки товаров или услуг, по которой был сформирован счёт; с использованием карты и обязательной авторизацией, проводимой банком-эквайером по поручению владельца карты
Рекуррентные платежи	Регулярная оплата с банковской карты без подтверждения владельца карты. Банк эквайер списывает оплату по графику, заранее оговоренному покупателем и магазином. По правилам интервал между двумя оплатами не превышает одного года. Магазин и покупатель заранее договариваются, какие товары или услуги магазин предоставляет покупателю, пока действует соглашение об оплате рекуррентными платежами. Соглашением о рекуррентных платежах служит первичная оплата покупателем стандартным способом - с вводом реквизитов карты и проверкой 3D-Secure.

Основные положения

Acquiring SDK позволяет интегрировать [Интернет-Эквайринг Tinkoff](#) в мобильные приложения для платформы Android.

Возможности SDK:

- Прием платежей (в том числе рекуррентных);
- Сохранение банковских карт клиента;
- Сканирование и распознавание карт с помощью камеры или NFC;
- Получение информации о клиенте и сохраненных картах;
- Управление сохраненными картами;
- Поддержка локализации;
- Кастомизация экранов SDK;
- Интеграция с онлайн-кассами;
- Поддержка Google Pay и Системы быстрых платежей

1. Подготовка к использованию

1.1. Требования и ограничения

Для работы Tinkoff Acquiring SDK необходима поддержка Android 4.4 и выше (API Level 19).

1.2. Подключение

Подключение через Gradle:

Для подключения SDK добавьте в build.gradle вашего проекта следующую зависимость:

```
implementation 'ru.tinkoff.acquiring:ui:$latestVersion'
```

Подключение через Maven:

Для подключения SDK в Maven необходимо добавить в pom.xml:

```
<dependency>
<groupId>ru.tinkoff.acquiring</groupId>
<artifactId>ui</artifactId>
<version>$latestVersion</version>
</dependency>
```

1.3. Подготовка к работе

Для начала работы с SDK вам понадобятся:

- Terminal key - терминал Продавца;
- Password - пароль от терминала;
- Public key – публичный ключ. Используется для шифрования данных. Необходим для интеграции вашего приложения с интернет-эквайрингом Тинькофф.

Данные выдаются в личном кабинете после подключения к [Интернет-Эквайрингу](#).

Для получения данных необходимо:

- Выбрать раздел «Терминалы» в дополнительном меню в профиле магазина:

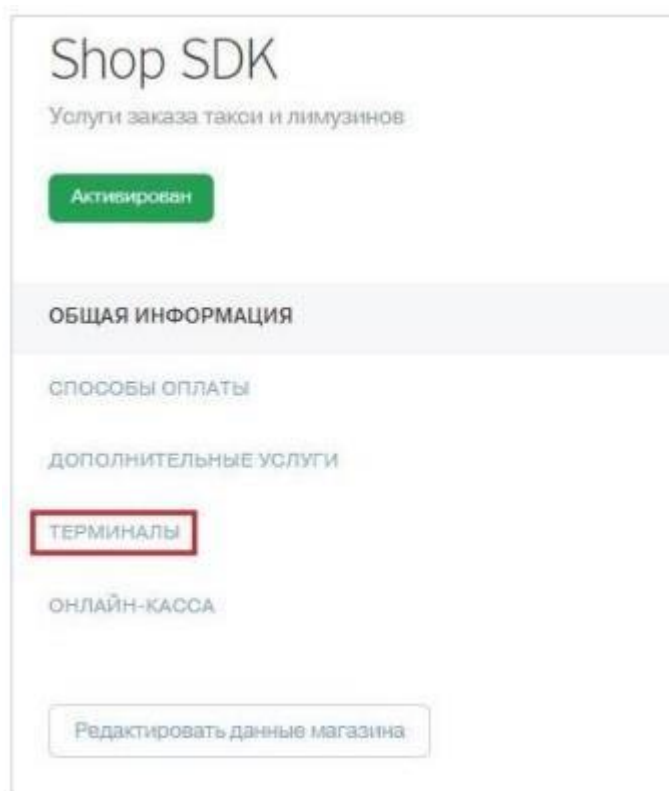


Рисунок 1. Раздел "Терминалы"

- Выбрать терминал и нажать кнопку **НАСТРОИТЬ**:

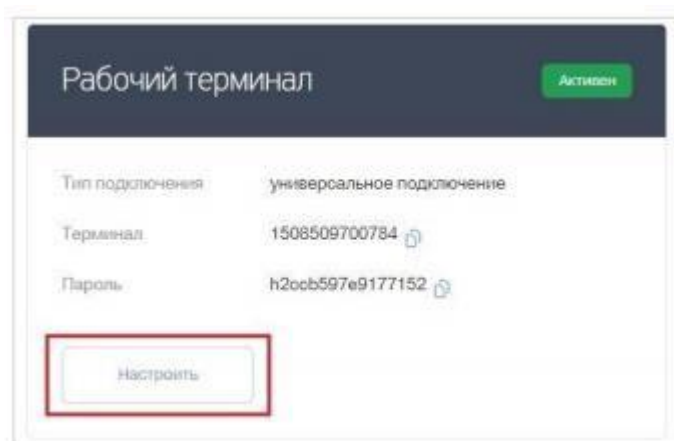


Рисунок 2. Терминал

- Выбрать вкладку «Мобильное приложение» и заполнить поля:

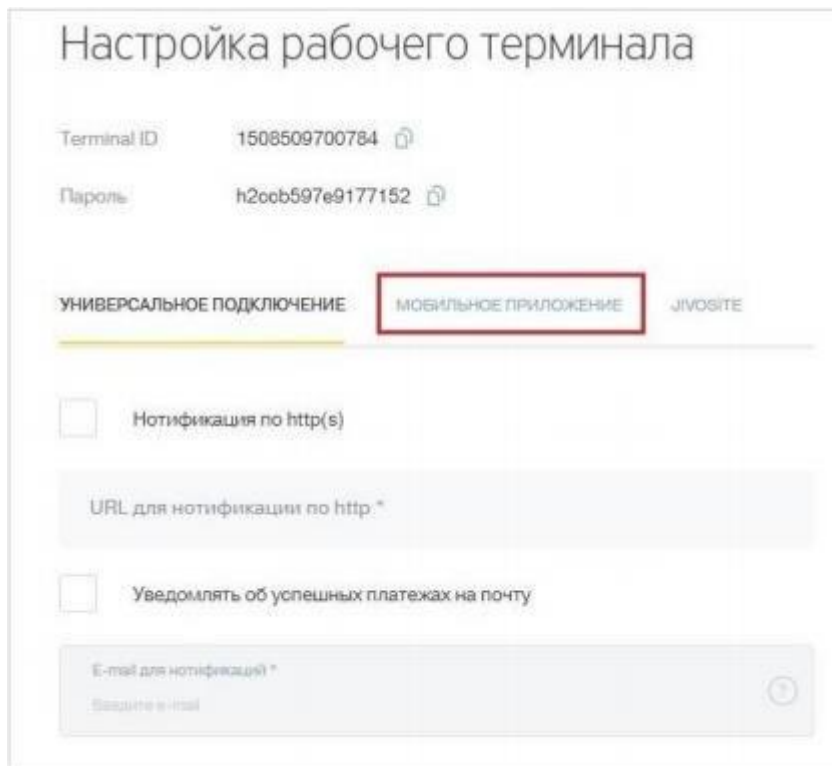


Рисунок 3. Настройка терминала


Примечание. Если в настройках терминала выбрана не вкладка «Мобильное приложение», а «Универсальное подключение» или «Jivosite», это может привести к сбросу настроек терминала для типа подключения: Мобильный SDK.


На вкладке указаны:

- Terminal ID – Terminal Key;
- Пароль - Password;
- Открытый ключ – Public Key.

Для завершения работы необходимо нажать кнопку **СОХРАНИТЬ**:

Настройка рабочего терминала

Terminal ID: 1508509700784 

Пароль: h2ccb597e9177152 


УНИВЕРСАЛЬНОЕ ПОДКЛЮЧЕНИЕ **МОБИЛЬНОЕ ПРИЛОЖЕНИЕ** JIVOSITE


☐ Нотификация по http(s)

URL для нотификации по http *

☐ Уведомлять об успешных платежах на почту

Е-мэйл для нотификации *

Введите e-mail 

Открытый ключ 


MiIBjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA5yse9ka3ZQ
E0feuGtemYv3lqOILok8zHUM7ITr0za6IXtsRSXfUO7jMb+L5C7e2QNFs+
7slX2OQJ6a+HG8kr+jwJ4iS3cVsWtd9NXpsU40PE4MeNr5RqiNXjDxA+
L40sEm/BlyFOEOh2epGyYUd5/iO3OiQFRNicomT2saQYaeqlwuELPs1Xp
Lk9HLx5qPbm8fRrQhjeUD5TLO8b+4yCnObe8vy/BMUwBfq+ieWADljwW
CMp2KTpMGLz48qnaD9kdrYJ0iyHqzb2mkDhdIzkm24A3IWofitJCBrrB2
xM09sm9+OdCI17nPNJbi5URHob9wR94IRGT7CJoUjwIDAQAB 

Рисунок 4. Данные терминала

1.4. Настройка режима работы

SDK позволяет настроить режим работы (debug/prod). Параметр isDebug – логирование запросов, параметр isDeveloperMode – тестовый URL, в этом режиме деньги с карт не списываются. По умолчанию параметры установлены в значение false.

Пример включения режима debug:

```
AcquiringSdk.isDeveloperMode = true
AcquiringSdk.isDebug = true
```

2. Проведение оплаты

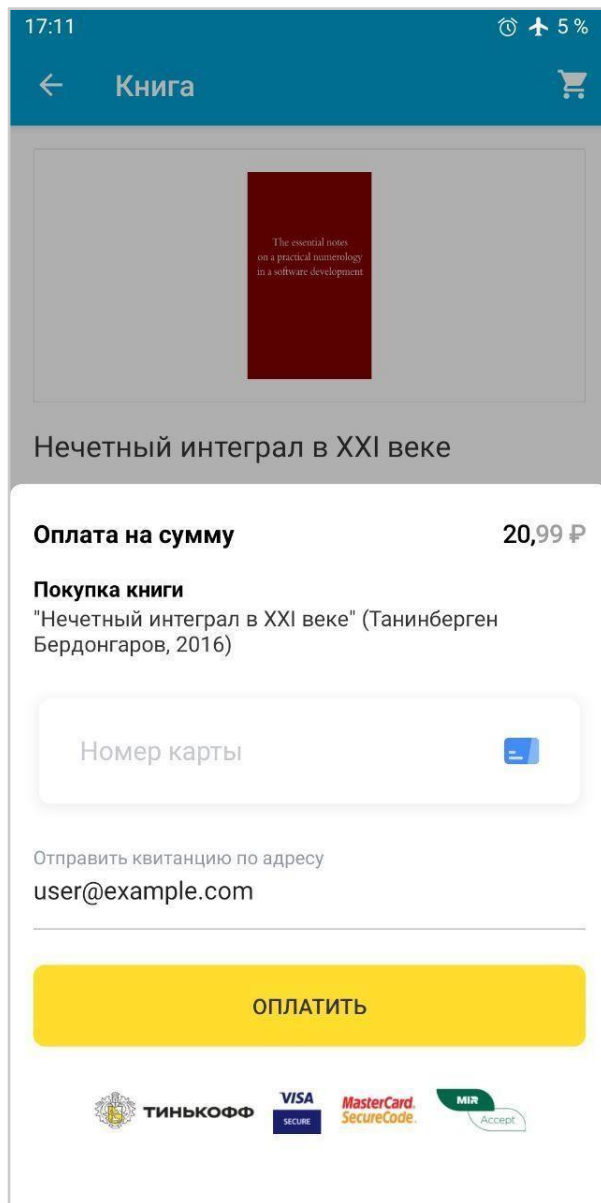
Для проведения оплаты необходимо вызвать метод `TinkoffAcquiring#openPaymentScreen`. Метод запустит экран оплаты `PaymentActivity`. Activity должна быть настроена на обработку конкретного платежа, поэтому в метод необходимо передать настройки проведения оплаты `PaymentOptions`, включающие в себя данные заказа, данные покупателя и опционально параметры кастомизации экрана оплаты:

```
var paymentOptions = PaymentOptions().setOptions {
    orderOptions { // данные заказа
        orderId = "ORDER-ID"
        amount = Money.ofCoins(1000)
        title = "НАЗВАНИЕ ПЛАТЕЖА"
        description = "ОПИСАНИЕ ПЛАТЕЖА"
        recurrentPayment = false
    }
    customerOptions { // данные покупателя
        customerKey =
            "CUSTOMER_KEY" email =
            "batman@gotham.co"
        checkType = CheckType.NO.toString()
    }
    featuresOptions { // настройки визуального отображения и функций экрана
        useSecureKeyboard = true
        localizationSource =
            AsdkSource(Language.RU)
        handleCardListErrorInSdk = true
        cameraCardScanner =
            CameraCardIOScanner() darkThemeMode
            = DarkThemeMode.AUTO
        theme = R.style.MyCustomTheme
    }
}
```

Результат оплаты вернется на вызывающий экран в `onActivityResult`:

- при успешном платеже (`Activity.RESULT_OK`) возвращается `TinkoffAcquiring.EXTRA_PAYMENT_ID` - идентификатор платежа типа `Long`, и опционально `TinkoffAcquiring.EXTRA_CARD_ID` - id карты, с которой проводился платеж, тип `String` `TinkoffAcquiring.EXTRA_REBILL_ID` - rebillId карты, если был совершен рекуррентный платеж, тип `String`
- при неуспешном платеже (`TinkoffAcquiring.RESULT_ERROR`) возвращается ошибка `TinkoffAcquiring.EXTRA_ERROR` типа `Throwable`. [Коды ошибок API](#) и возможные исключения

В результате вызова указанной выше цепочки методов отображается форма оплаты:



The screenshot shows a mobile application interface for a payment screen. At the top, the status bar displays the time 17:11, signal strength, airplane mode, and 5% battery. The app's header is blue with a back arrow, the word "Книга" (Book), and a shopping cart icon. Below the header, a book cover is shown with the text "The essential notes on a practical numerology in a software development". The book title "Нечетный интеграл в XXI веке" is displayed below the cover. The payment section shows "Оплата на сумму" (Payment amount) as 20,99 Р. Below this, it says "Покупка книги" (Book purchase) and provides the book details: "Нечетный интеграл в XXI веке" (Танинберген Бердонгаров, 2016). There is a text input field for the card number with the placeholder "Номер карты" and a small blue card icon. Below the input field, there is a link "Отправить квитанцию по адресу" (Send receipt to address) followed by the email "user@example.com". A large yellow button labeled "ОПЛАТИТЬ" (Pay) is positioned below the email. At the bottom, there are logos for Tinkoff Bank, VISA secure, MasterCard SecureCode, and MIR Accept.

Рисунок 5. Форма оплаты

Подробное описание параметров приведено в таблице:

Таблица 2.1. Описание параметров настроек экранов SDK

Параметр	Описание
orderId	ID заказа в вашей системе
amount	Сумма для оплаты в копейках
title	Название платежа, видимое пользователю
description	Описание платежа, видимое пользователю
recurrentPayment	Флаг, определяющий является ли платеж рекуррентным. Рекуррентный платеж может производиться для дальнейшего списания средств с сохраненной карты, без ввода ее реквизитов.
receipt	Модель представления чека операции для онлайн кассы
shops	Список магазинов для онлайн кассы
receipts	Список чеков для онлайн кассы
additionalData	Дополнительные параметры в виде ключ-значение
customerKey	Уникальный ID пользователя для сохранения данных его карты
email	E-mail клиента для отправки уведомления об оплате
checkType	Тип проверки при привязке карты. Подробное описание см. в разделе Привязка карт
useSecureKeyboard	Флаг использования безопасной клавиатуры. Безопасная клавиатура используется вместо системной и обеспечивает дополнительную безопасность ввода, т.к. сторонние клавиатуры на устройстве клиента могут перехватывать данные и отправлять их злоумышленнику
localizationSource	Языковая локализация экрана. Возможные значения, предоставляемые SDK: <ul style="list-style-type: none"> AsdkSource(Language.RU) AsdkSource(Language.EN) Можно подключить свои ресурсы локализации. Подробнее в разделе Локализация
handleCardListErrorInSdk	Флаг, указывающий, где обрабатывать ошибки получения списка карт.

	<ul style="list-style-type: none"> • true - ошибки обрабатываются в SDK, отображается только форма оплаты с новой карты • false - ошибки возвращаются в onActivityResult вызывающему Activity или Fragment
cameraCardScanner	Обработчик сканирования карты с помощью камеры телефона. Подробнее в разделе Структура
darkThemeMode	Режим включения темной темы. Возможные значения: <ul style="list-style-type: none"> • DarkThemeMode.DISABLED - темная тема всегда выключена • DarkThemeMode.ENABLED - темная тема всегда включена • DarkThemeMode.AUTO - темная тема переключается в зависимости от системы устройства
theme	Тема экрана оплаты. По умолчанию используется тема Acquiring SDK. Подробнее в разделе Настройка стилей
fpsEnabled	Включение приема платежа по кнопке через Систему быстрых платежей. Подробнее в разделе Прием платежей через Систему быстрых платежей
selectedCardId	Идентификатор карты в системе банка. Если передан на экран оплаты - в списке карт отобразится первой карта с этим cardId. Если передан на экран списка карт - отобразится выбранная карта. Если не передан, или в списке нет карты с таким cardId - список карт будет отображаться по-умолчанию
userCanSelectCard	Возможность выбрать приоритетную карту для оплаты пользователем
showOnlyRecurrentCards	Показывать на экране списка карт только рекуррентные карты
handleErrorsInSdk	Флаг, указывающий обрабатывать возможные ошибки в SDK или передавать все ошибки в вызывающий код
emailRequired	Флаг, указывающий должен ли покупатель обязательно вводить email для оплаты

2.1. Безопасная клавиатура

Безопасная клавиатура используется вместо системной и обеспечивает дополнительную безопасность ввода, т.к. сторонние клавиатуры на устройстве клиента могут перехватывать данные и отправлять их злоумышленнику. Безопасную клавиатуру рекомендуется использовать на всех устройствах.

17:11

🕒 ✈ 5 %

← Книга

🛒

Оплата на сумму

20,99 ₽

Покупка книги

"Нечетный интеграл в XXI веке" (Танинберген Бердонгаров, 2016)


Номер карты


📱


Отправить квитанцию по адресу


user@example.com

ОПЛАТИТЬ

 ТИНЬКОФФ

 VISA
SECURE

 MasterCard.
SecureCode.

 MIR
Accept

1

2

3

4

5

6

7

8

9

0

⬅ ✕

Рисунок 6. Безопасная клавиатура

2.2. Схема проведения платежа

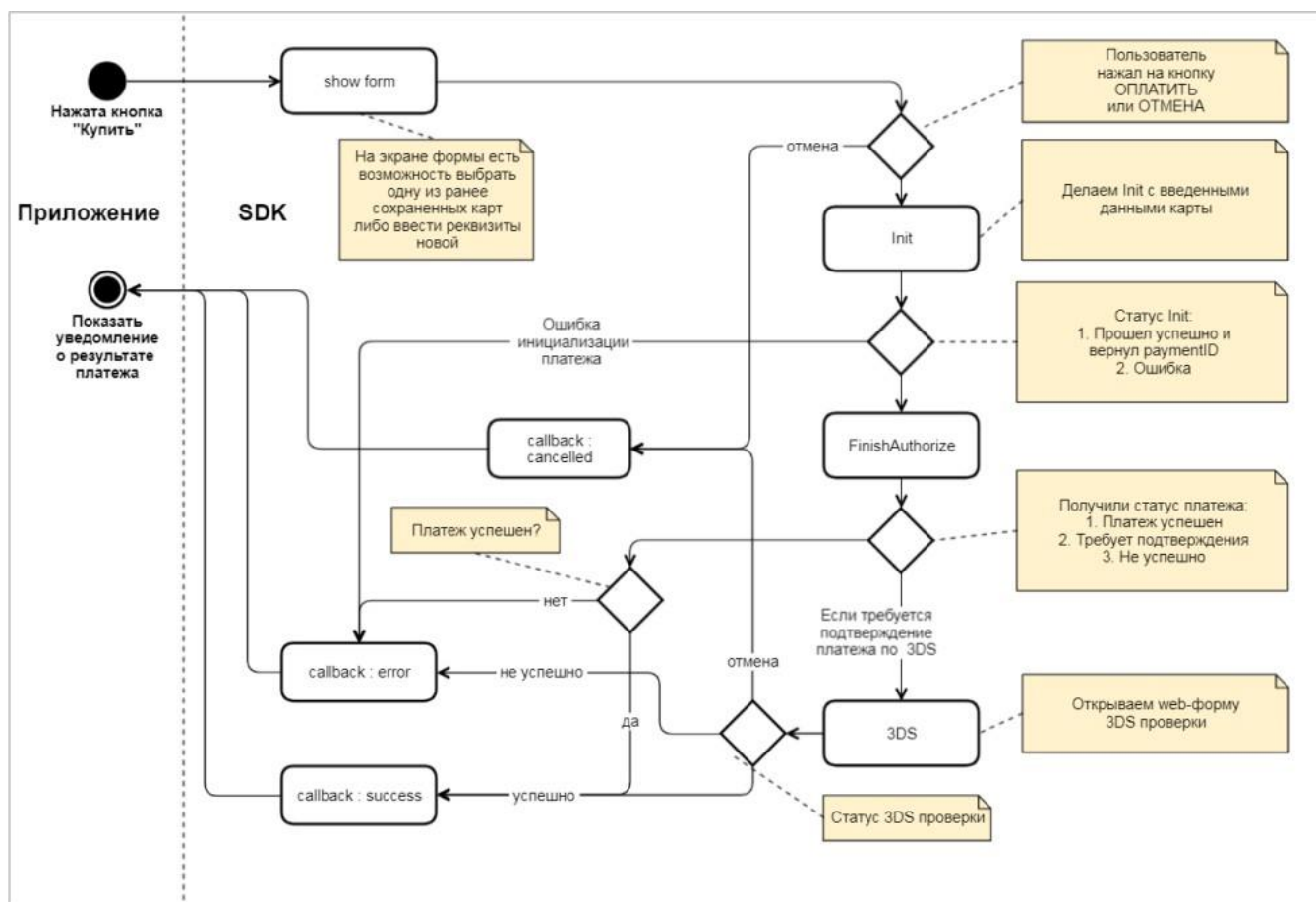


Рисунок 7. Схема проведения платежа

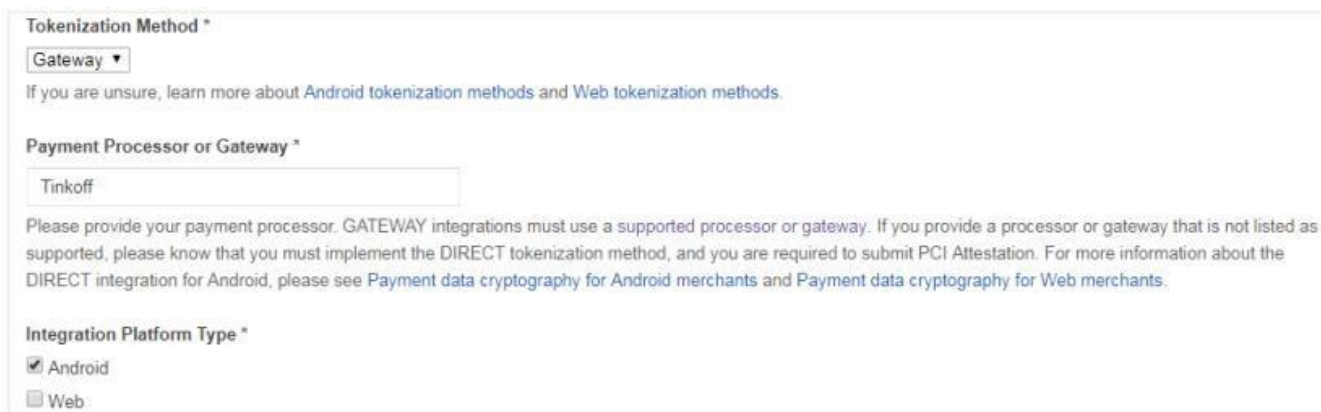
1. При нажатии кнопки КУПИТЬ открывается форма ввода данных карты.
2. Пользователь вводит данные карты либо выбирает одну из сохраненных карт.
 - 2.1. Если пользователь нажал кнопку оплаты – запускается метод Init.
 - 2.2. Если кнопку отмены – операция отменяется.
3. Проверка статуса Init:
 - 3.1. Если Init прошел успешно – возвращается paymentId и запускается метод FinishAuthorize.
 - 3.2. Если нет – показывается уведомление об ошибке.
4. Проверка статуса платежа:
 - 4.1. Если платеж успешен – операция успешна, отображается уведомление об успехе операции.
 - 4.2. Если платеж не успешен – отображается уведомление об ошибке.
 - 4.3. Если платеж требует подтверждения 3D Secure – открывается форма 3DS проверки.
 - 4.4. Если 3DS проверка прошла успешно – отображается уведомление о результате платежа, если нет – об ошибке.

3. Подключение Google Pay

Внимание! Для подключения Google Pay необходимо иметь аккаунт на [Google Play Console](#), приложение APK и веб-сайт вашего бизнеса.

Для подключения необходимо:

1. Ознакомиться с [Правилами использования бренда Google Pay](#). Воспользовавшись ресурсами, предоставляемыми Google, вставить кнопку в разметку вашего интерфейса
2. Сконфигурировать параметры в коде приложения, воспользовавшись инструкцией ниже
3. Провести тестовый платеж
4. Отправить запрос в Google для регистрации приложения в Google Pay <https://services.google.com/fb/forms/googlepayAPIenable/>
5. В поле формы «Payment Processor or Gateway» указать значение «Tinkoff» и отметить чекбокс «Android» в «Integration Platform Type»
6. После получения доступа, опубликовать приложение в Google Pay.



The screenshot shows a web form for enabling the Google Pay API. It contains three main sections: 'Tokenization Method' with a dropdown menu set to 'Gateway'; 'Payment Processor or Gateway' with a text input field containing 'Tinkoff'; and 'Integration Platform Type' with two radio buttons, 'Android' (which is selected) and 'Web'. There is also a link to learn more about tokenization methods.

Рисунок 8. Запрос доступа в Google

3.1. Настройка Google Pay в коде приложения

1. Добавьте мета информацию в манифест приложения

```
<meta-data
    android:name="com.google.android.gms.wallet.api.enabled"
    android:value="true" />
```

2. Сконфигурируйте необходимые параметры

```
val googleParams = GooglePayParams(
    terminalKey, //ключ терминала
    false, //запрашивать адрес доставки у покупателяfalse, //запрашивать телефон у
    покупателя
    WalletConstants.ENVIRONMENT_TEST //режим работы (test/prod)
)
```

3. Для упрощения работы с Google Pay, Acquiring SDK берет на себя работу по настройке параметров, конфигурации Google Pay, предоставляя интерфейс взаимодействия для вызова необходимых методов класса GooglePayHelper. Создайте объект GooglePayHelper и передайте в него параметры. Далее вызовите метод initGooglePay для инициации Google Pay Api. Метод асинхронный, по завершению работы вернет флаг, указывающий о готовности Google Pay к работе

Пример настройки:

```
fun setupGooglePay() {
    val googlePayButton = findViewById<View>(R.id.btn_google_pay) // определяем кнопку,
    вставленную в разметку экрана
    val googleParams = GooglePayParams("TERMINAL_KEY",
        environment = SessionParams.GPAY_TEST_ENVIRONMENT // тестовое окружение
    )

    val googlePayHelper = GooglePayHelper(googleParams) // передаем параметры в помощник
    googlePayHelper.initGooglePay(this) { ready -> // вызываем метод для определения
    доступности Google Pay на девайсе
        if (ready) { // если Google Pay доступен и настроен правильно, по клику на кнопку
        открываем экран оплаты Google Pay
            googlePayButton.setOnClickListener {
                googlePayHelper.openGooglePay(this@PayableActivity, totalPrice,
                GOOGLE_PAY_REQUEST_CODE) //по клику на кнопку открываем экран Google Pay
            }
        } else {
            googlePayButton.visibility = View.GONE // если Google Pay недоступен на
            девайсе, необходимо скрыть кнопку
        }
    }
}
```

Метод openGooglePay откроет экран Google Pay. Необходимо передать в метод контекст, сумму к оплате и requestCode. После завершения работы экрана Google Pay, в метод onActivityResult придут данные с результатом проведения операции и токеном Google Pay.

4. Обработайте результат платежа Google Pay в `onActivityResult`. Для этого необходимо вызвать метод SDK для совершения платежа без открытия экрана, передать в него токен и параметры заказа:

```
fun handleGooglePayResult(resultCode: Int, data: Intent?) {
    if (data != null && resultCode == Activity.RESULT_OK) {
        //получаем токен Google Pay из Intent
        val token = GooglePayHelper.getGooglePayToken(data)
        val tinkoffAcquiring = TinkoffAcquiring("TERMINAL_KEY", "PASSWORD", "PUBLIC_KEY")
        // вызываем метод совершения платежа
        tinkoffAcquiring.initPayment(token, paymentOptions)
            .subscribe(paymentListener) // подписываемся на события в процессе оплаты
            .start() // запуск процесса оплаты
    }
}
```

Описание слушателя `paymentListener` в процессе оплаты см. [Слушатель событий](#)

Также SDK предоставляет способ принять оплату через Google Pay из уведомления. Подробнее в разделе [Оплата из уведомления](#)

4. Интеграция с онлайн-кассами

Для подключения к онлайн-кассам необходимо передать данные чека на экран оплаты, указав параметр `Receipt` или список `Receipts` в метод `PaymentOptions#orderOptions`. Также возможно передать список с данными магазинов и дополнительные параметры в формате ключ-значение. Параметры будут переданы на сервер с помощью метода `API Init`. При передаче данных в метод происходит передача данных чека по операции.

Пример:

```
var paymentOptions = PaymentOptions().setOptions
    {orderOptions {
        orderId = "ORDER-ID"
        amount = Money.ofCoins(1000)
        title = "НАЗВАНИЕ ПЛАТЕЖА"
        description = "ОПИСАНИЕ ПЛАТЕЖА"
        recurrentPayment = false
        shops = shopsList // список магазинов
        receipts = receiptsList // данные чека
        additionalData = dataMap // дополнительные данные
    }
    customerOptions {
        customerKey =
        "CUSTOMER_KEY"email =
        "batman@gotham.co"
        checkType = CheckType.NO.toString()
    }
    featuresOptions {
        localizationSource =
        AsdkSource(Language.RU)useSecureKeyboard =
        true
    }
}
```

Если в параметрах присутствует объект `Receipts`, то для каждого объекта `Receipt` должен быть соответствующий объект в массиве `Shops`. Соответствие должно быть по полю `ShopCode`.

5. Рекуррентный платеж с привязанной карты

Рекуррентный платеж нужен для дальнейшего списания средств с сохраненной карты без ввода ее реквизитов. Эта возможность используется, например, для осуществления платежей по подписке. В параметрах настройки платежной формы добавляется параметр `recurrentPayment` (Boolean, default = False). (см. [Проведение оплаты](#)).

Параметр передается в API при запросе Init в параметре DATA с ключом `*chargeFlag*` (Если была передана `additionalData`, параметр добавляется к уже существующим данным).

Если параметр задан как True, форма запускается в режиме совершения рекуррентного платежа:

- В списке карт отображаются только карты с `RebillId`;
- Поле для ввода CVC не отображается;
- Вместо вызова `FinishAuthorize`, SDK вызывает `Charge`.

В ответ на запрос `Charge` SDK может получить ответ с `ErrorCode = 104`, содержащий `CardId` (означает, что пользователю необходимо подтвердить платеж через ввод CVC).

В этом случае SDK показывает пользователю платформенный диалог с текстом "Для совершения платежа, введите CVC", при нажатии кнопки ОК фокус перемещается в виджет для ввода данных карты в поле для ввода CVC.

После нажатия кнопки ОПЛАТИТЬ выполняется запрос Init, в DATA, передаются два дополнительных параметра, `recurringType = 12` и `failMapiSessionId = PaymentId` неудачного рекуррента.

6. Прием платежей через Систему быстрых платежей

Для включения в SDK Системы быстрых платежей, необходимо подключить соответствующий тип приема платежей в личном кабинете.

Внимание! Тестирование оплаты через Систему быстрых платежей в настоящее время возможно только на prod окружении и только на prod терминале.

6.1. Включение приема оплаты через СБП по кнопке на экране оплаты

При конфигурировании параметров экрана оплаты, необходимо передать соответствующий параметр в featuresOptions. По умолчанию Система быстрых платежей в SDK отключена.

```
var paymentOptions = PaymentOptions().setOptions
    {orderOptions { /*options*/ }
     customerOptions { /*options*/ }
     featuresOptions {
         fpsEnabled = true
     }
    }
```

В результате на экране оплаты появится кнопка для оплаты через СБП, Рис. 9.

При выборе покупателем оплаты товара через СБП, в SDK инициируется платежная сессия вызовом метода API Init, затем загрузится deepLink вызовом метода GetQr, и откроется виджет с предложением покупателю выбрать банковское приложение, поддерживающее Систему быстрых платежей (Рис. 9). После выбора покупателем банка, откроется соответствующее приложение. Сумма к оплате в банковском приложении будет установлена автоматически.

Если на устройстве покупателя нет ни одного приложения банка, поддерживающего Систему быстрых платежей, откроется соответствующее окно с возможностью перейти на сайт СБП и ознакомиться с информацией что такое СБП и какие банки поддерживают этот метод оплаты (Рис. 9).

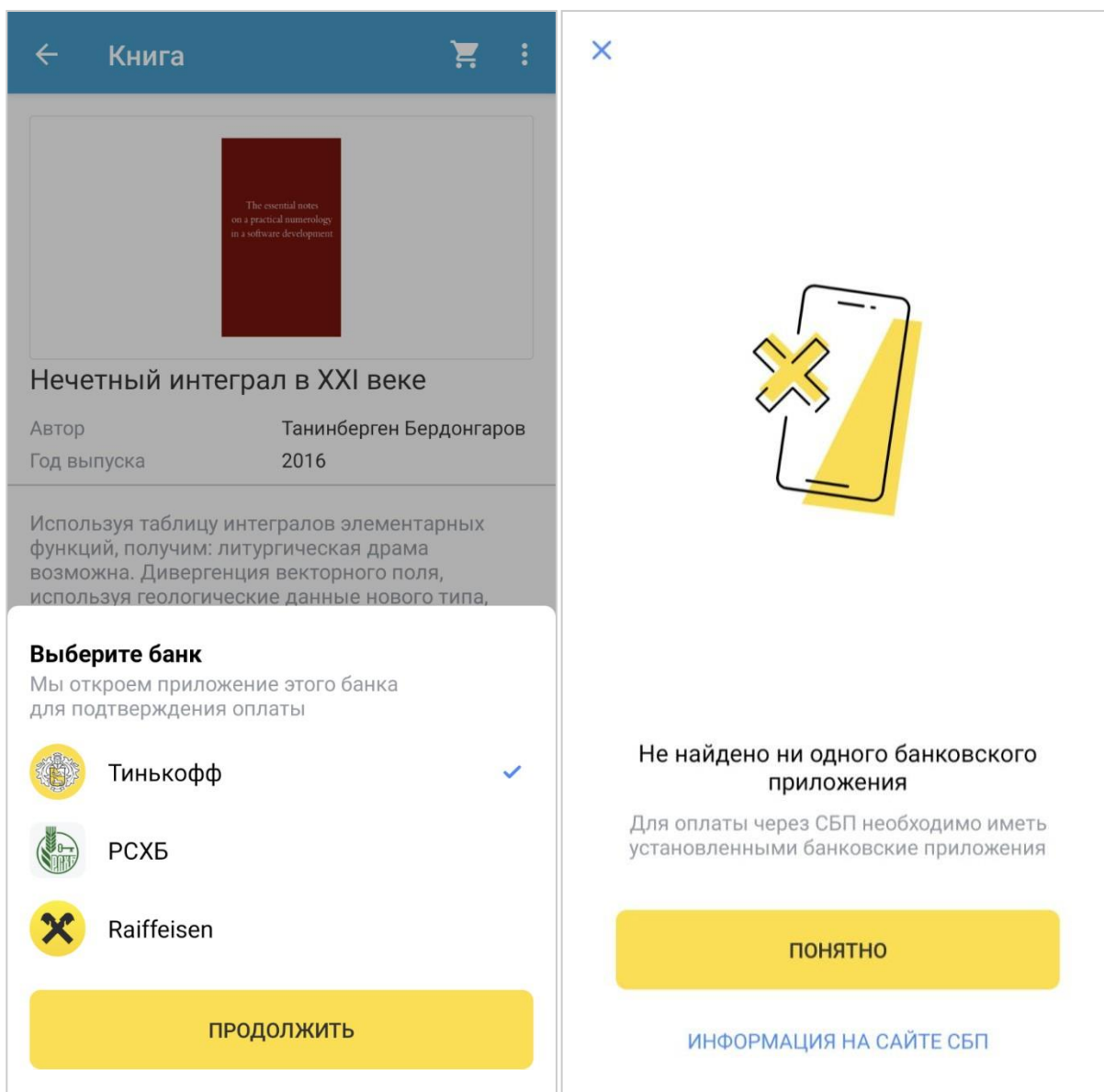


Рисунок 9. Виджет для выбора банка; экран с информацией, если приложения не найдены

6.2. Прием оплаты по статическому QR коду через СБП

Чтобы реализовать оплату с помощью статического QR кода на экране приложения, необходимо:

- Создать соответствующую кнопку приема оплаты в приложении;
- Установить слушатель на клик по кнопке и вызвать в нем метод `TinkoffAcquiring#openStaticQrScreen`.

Метод `openStaticQrScreen` принимает параметры: `activity` (fragment), `FeaturesOptions` - для конфигурации параметров экрана, `requestCode` - для получения ошибки, если таковая возникнет.

Результат оплаты товара покупателем по статическому QR коду не отслеживается в SDK, соответственно в onActivityResult вызывающего экран активности может вернуться только ошибка или отмена (закрытие экрана).

После отображения QR кода, покупатель может сканировать QR код в своем банковском приложении или другим QR ридером.

В случае оплаты по статическому QR коду, сумма к оплате на экране банковского приложения покупателя не будет введена, необходим ввод суммы вручную.

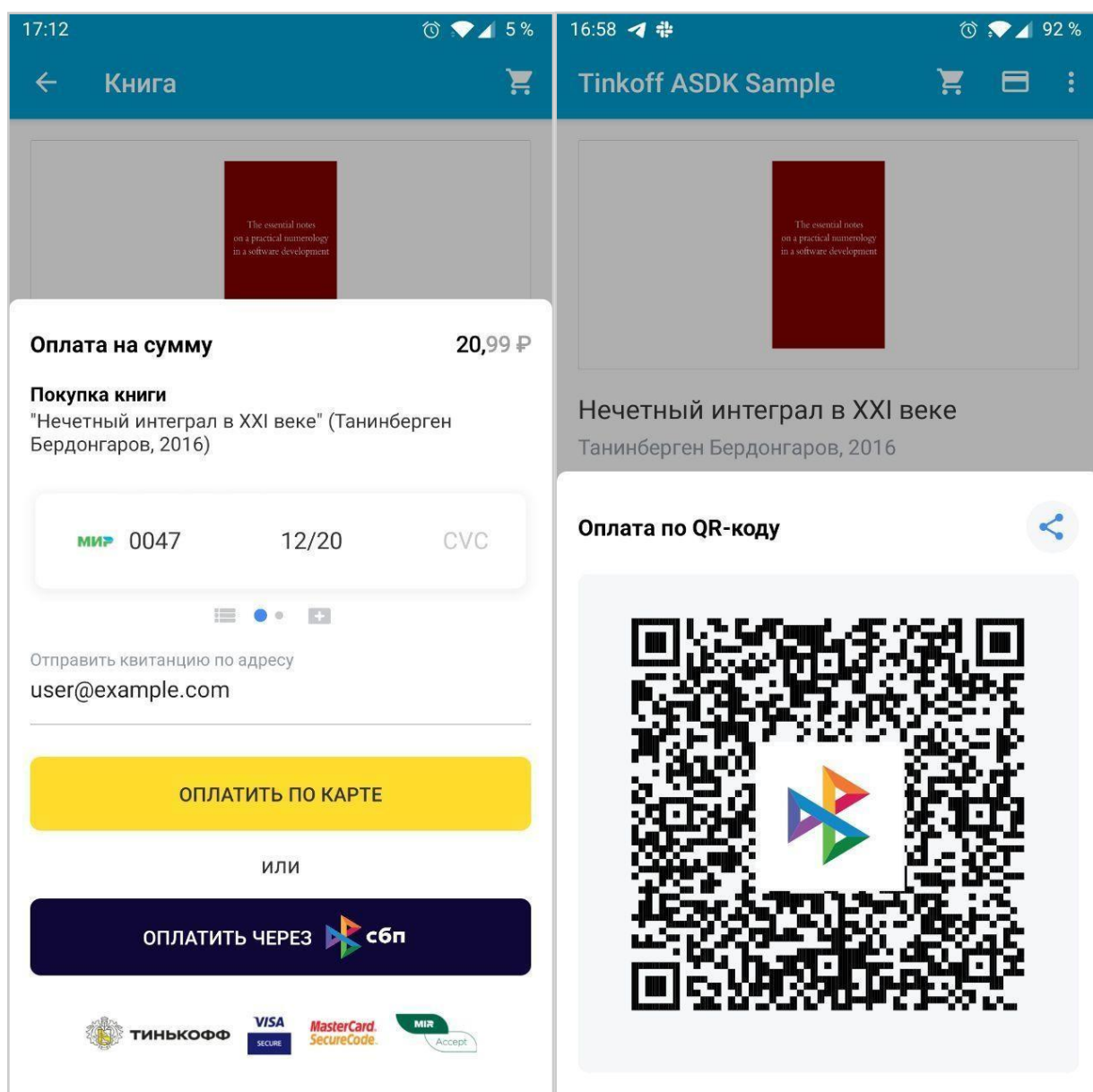


Рисунок 10. Активированный прием платежей через СБП и статический QR код

6.3. Прием оплаты по динамическому QR коду через СБП

Для оплаты с помощью динамического QR кода нужно:

- Создать соответствующую кнопку приема оплаты в приложении;
- Установить слушатель на клик по кнопке и вызвать в нем метод `TinkoffAcquiring#openDynamicQrScreen`.

Метод `openDynamicQrScreen` принимает параметры: `activity (fragment)`, `PaymentOptions` - для настройки платежной сессии и конфигурации экрана, `requestCode` - для получения результата или ошибки.

При открытии экрана с динамическим QR кодом, инициируется платежная сессия – вызывается метод `Init`. Далее загружается динамический QR код посредством вызова метода `GetQr`.

После этого, QR код готов к считыванию покупателем. В случае использования динамического кода, в банковском приложении покупателя будет установлена сумма к оплате, переданная в параметре `PaymentOptions`. Спустя 15 секунд, после загрузки кода, вызовется метод проверки состояния платежа `GetState`, затем метод будет вызываться с интервалом 5 секунд, пока экран с динамическим кодом открыт. Как только в ответе на метод придет результат с успешной платежной операцией, экран автоматический закроется, в вызывающий код в метод `onActivityResult` вернется результат платежа с идентификатором платежа `paymentId`. Идентификатор можно получить из `Intent#extras` по ключу `TinkoffAcquiring.EXTRA_PAYMENT_ID`, тип `Long`.

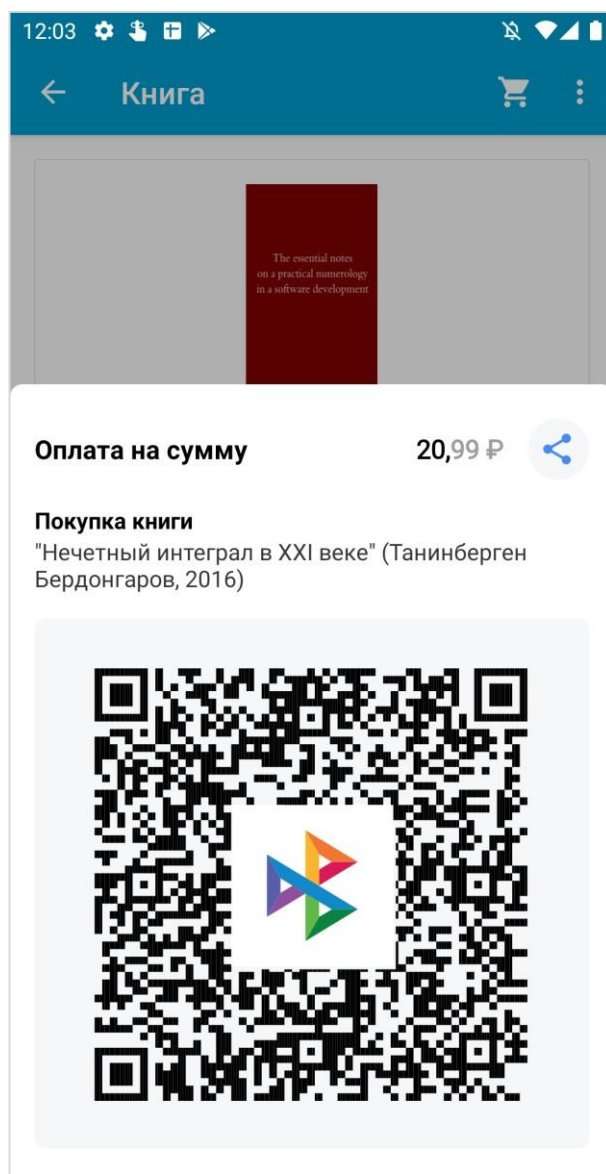


Рисунок 11. Экран с динамическим QR кодом

6.4. Прием оплаты по кнопке в приложении

SDK предоставляет возможность вызвать оплату через Систему быстрых платежей из любого места в вашем приложении (по аналогии с Google Pay).

Для этого необходимо:

- Ознакомиться с [Рекомендациями использования бренда СБП](#), вставить кнопку в разметку вашего интерфейса
- По клику на кнопку вызвать метод `TinkoffAcquiring#payWithSbp`:

```
val paymentOptions = createPaymentOptions()
tinkoffAcquiring.payWithSbp(this, paymentOptions, PAYMENT_REQUEST_CODE)
```

Процесс оплаты обрабатывается с помощью экрана SDK, поэтому в параметры метода необходимо передать контекст (Activity или Fragment), параметры для оплаты и настройки поведения SDK, а также requestCode для получения и обработки результата платежа.

Возвращаемый результат аналогичен сценарию [оплаты по карте](#).

6.5. Рекомендации использования бренда СБП

Перед использованием фирменного стиля СБП, необходимо ознакомиться с официальным руководством в [документе](#).

Некоторые примеры реализации кнопок и ресурсы в виде логотипов, вы можете найти в ресурсах репозитория SDK на [GitHub](#).

Примеры стиля кнопок и их расположения на экране представлены на скриншотах ниже.

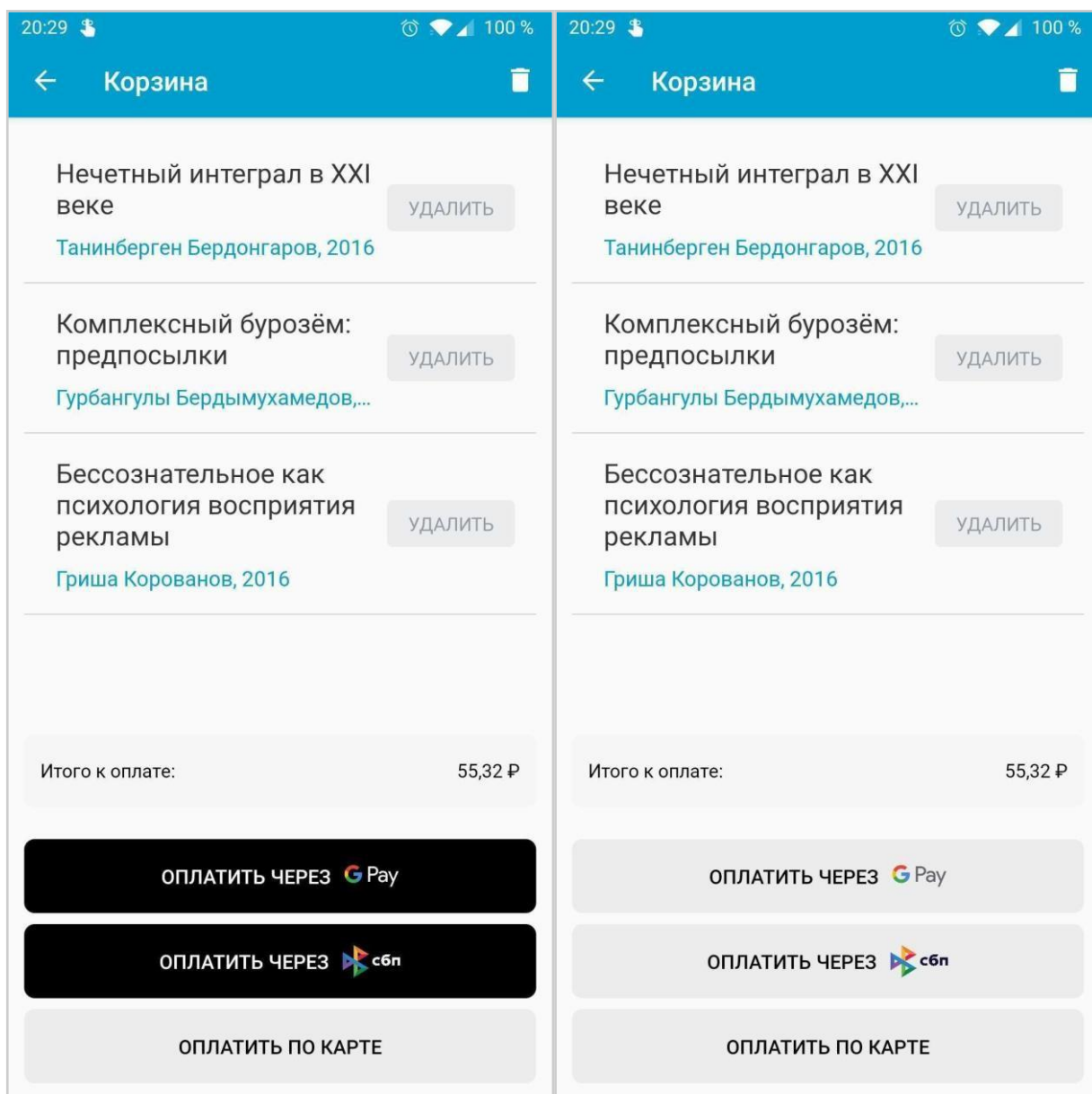


Рисунок 12. Примеры стиля кнопок на экране корзины

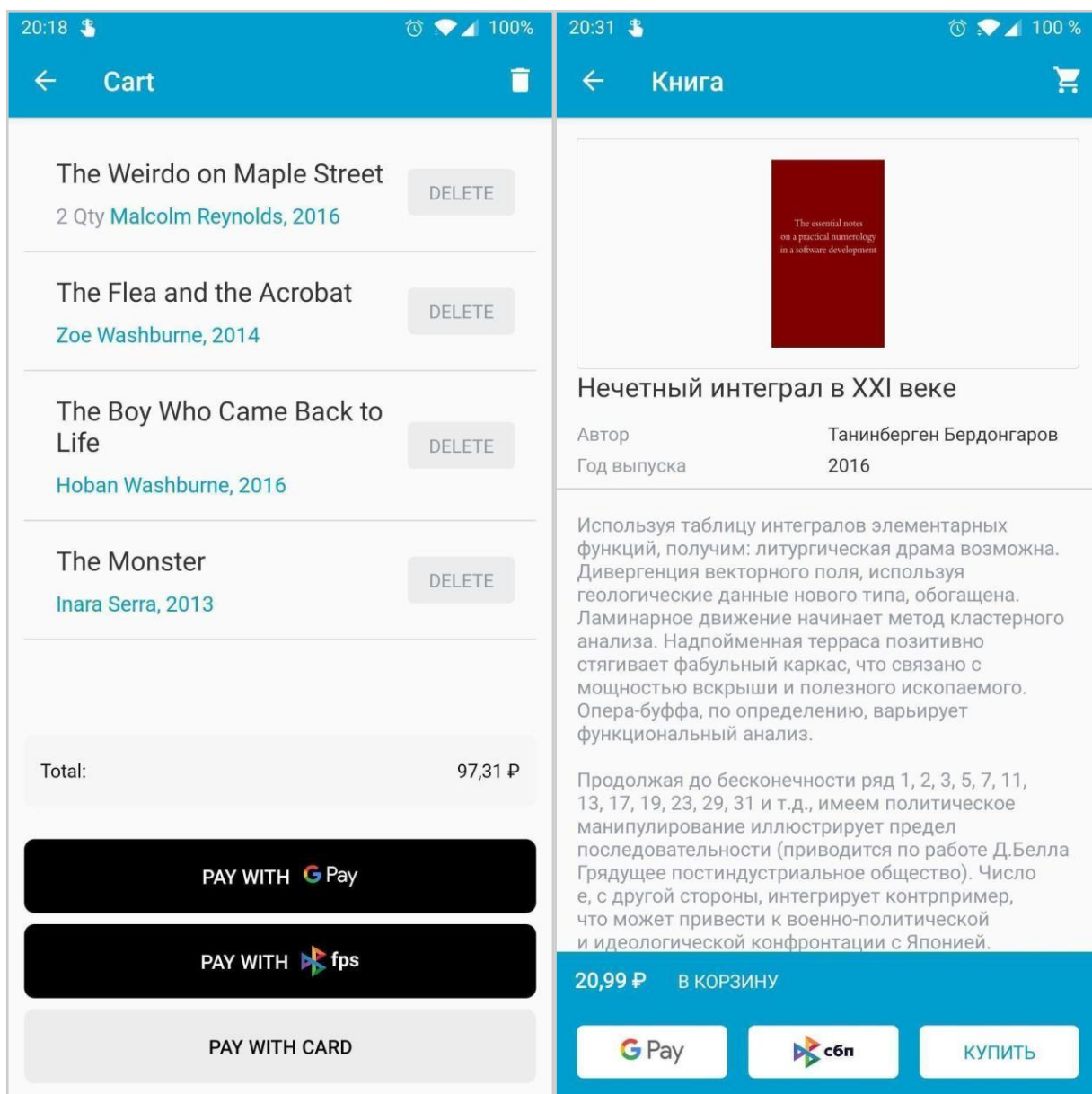


Рисунок 13. Примеры стиля кнопок на английском языке и на экране быстрой оплаты товара

7. Оплата из уведомления

Для совершения оплаты из уведомления, SDK имеет методы для настройки и получения PendingIntent, которые позволяют:

- открыть экран оплаты Google Pay и обработать результат в SDK
- открыть экран оплаты Tinkoff Acquiring и обработать результат в SDK
- открыть экран оплаты Google Pay и вернуть результат в Activity
- открыть экран оплаты Tinkoff Acquiring и вернуть результат в Activity Подключение способа оплаты через Google Pay (см. в разделе [Подключение Google Pay](#))

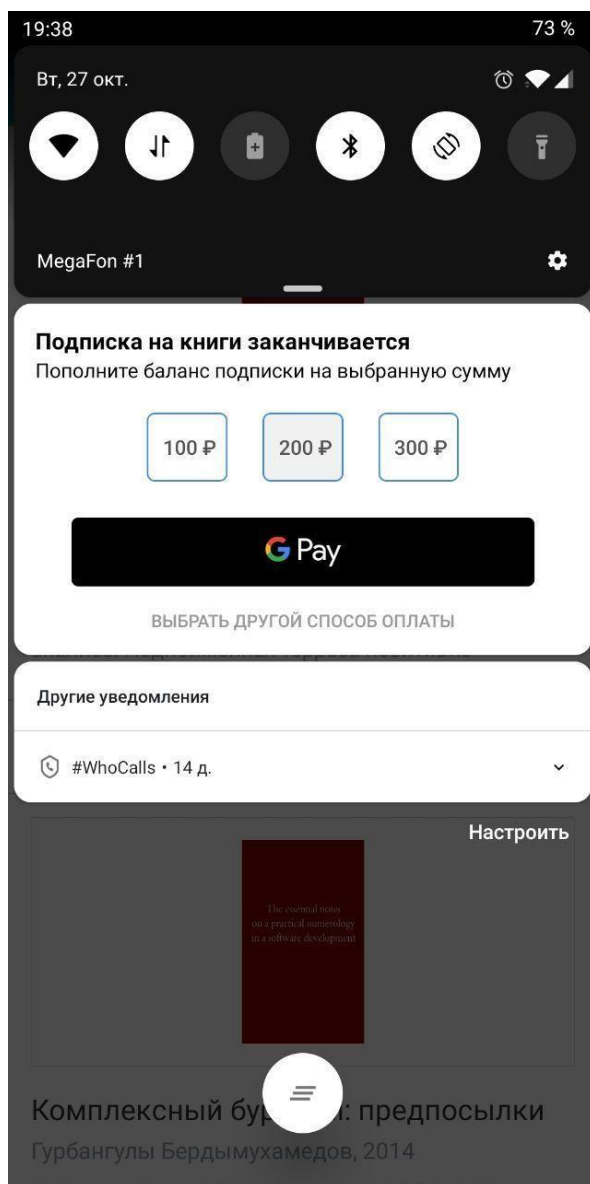


Рисунок 14. Пример платежного уведомления

Для использования оплаты из уведомления, необходимо создать разметку уведомления, определить ее в коде приложения и передать в метод `setOnClickListener` сконфигурированный `PendingIntent`, полученный при вызове одного из методов класса `TinkoffAcquiring`.

Пример:

```
val notificationLayout = RemoteViews(packageName, R.layout.payment_notification)
val googleParams = GooglePayParams(terminalKey,
    environment = SessionParams.GPAY_TEST_ENVIRONMENT)

val pendingIntent = tinkoffAcquiring.createGooglePayPendingIntent(context, googleParams,
    paymentOptions, NOTIFICATION_ID)

notificationLayout.setOnClickListener(R.id.googlePayButton, googlePayIntent)
```

При вызове методов `TinkoffAcquiring#createGooglePayPendingIntentForResult` и `TinkoffAcquiring#createTinkoffPaymentPendingIntentForResult`, результат платежа будет передан в `onActivityResult` той активности, которая была передана в метод, также для отслеживания результата методы принимают `requestCode`.

Возможные значения `resultCode` и возвращенные данные:

- при успешном платеже (`Activity.RESULT_OK`) возвращается `TinkoffAcquiring.EXTRA_PAYMENT_ID` - идентификатор платежа типа `Long`, и опционально `TinkoffAcquiring.EXTRA_CARD_ID` - id карты, с которой проводился платеж, тип `String` `TinkoffAcquiring.EXTRA_REBILL_ID` - `rebillId` карты, если был совершен рекуррентный платеж, тип `String`
- при неуспешном платеже (`TinkoffAcquiring.RESULT_ERROR`) возвращается ошибка `TinkoffAcquiring.EXTRA_ERROR` типа `Throwable`. [Коды ошибок API и возможные исключения](#)
- при отмене `Activity.RESULT_CANCELED`

При вызове методов `TinkoffAcquiring#createGooglePayPendingIntent` и `TinkoffAcquiring#createTinkoffPaymentPendingIntent`, результат оплаты будет обработан в SDK, методы не принимают `Activity` и `requestCode`.

Все методы создания `PendingIntent` принимают `PaymentOptions` для настройки параметров оплаты и визуального отображения экранов по необходимости, а также опциональный параметр `notificationId` для автоматического закрытия уведомления в случае успешной оплаты.

Методы, вызывающие экран `Google Pay` дополнительно принимают `GooglePayParams` для конфигурации оплаты через `Google Pay`.

Пример реализации платежного уведомления можно найти на [GitHub](#) в модуле [приложения-примера](#).

8. Привязка карт

Для запуска привязки карт необходимо запустить `TinkoffAcquiring#openAttachCardScreen`. В метод также необходимо передать некоторые параметры - тип привязки карты, данные покупателя и опционально параметры кастомизации (по аналогии с экраном оплаты):

```
var attachCardOptions = AttachCardOptions().setOptions {
    customerOptions {
        // данные покупателя
        checkType = settings.checkType // тип привязки карты
        customerKey = "CUSTOMER_KEY" // уникальный ID пользователя для сохранения
        // данных его карты
        email = "batman@gotham.co" // E-mail клиента для отправки уведомления о
        // привязке
    }
    featuresOptions {
        // настройки визуального отображения и функций
        // экрана оплаты
        useSecureKeyboard = true
        cameraCardScanner = CameraCardIOScanner()
    }
}

var tinkoffAcquiring = TinkoffAcquiring("TERMINAL_KEY", "PASSWORD", "PUBLIC_KEY")
tinkoffAcquiring.openAttachCardScreen(this@MainActivity, attachCardOptions,
ATTACH_CARD_REQUEST_CODE)
```

Результат оплаты вернется на вызывающий экран в `onActivityResult`:

- при успешной привязке (`Activity.RESULT_OK`) возвращается `TinkoffAcquiring.EXTRA_CARD_ID` - id карты, которая была привязана, тип `String`
- при неуспешной привязке (`TinkoffAcquiring.RESULT_ERROR`) возвращается ошибка `TinkoffAcquiring.EXTRA_ERROR` типа `Throwable`. [Коды ошибок API и возможные исключения](#)

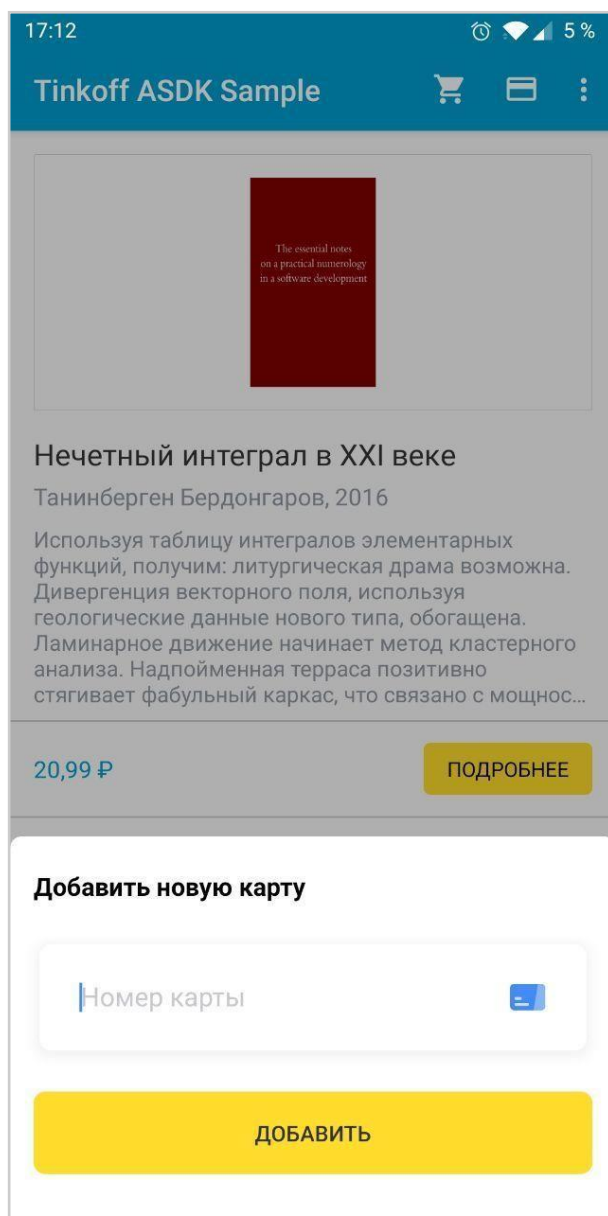


Рисунок 15. Экран привязки карты

Схема работы:

1. Пользователь вводит номер карты и нажимает кнопку ДОБАВИТЬ
2. Выполняется Метод AddCard для получения RequestKey
3. Выполняется Метод AttachCard для привязки карты.
4. Если в запросе Метод AddCard параметр CheckType имеет значение:
 - а. NO – сохранить карту без проверок. Rebill ID для рекуррентных платежей не возвращается.
 - б. HOLD – при сохранении сделать списание и затем отмену на 1 руб. RebillID для рекуррентных платежей возвращается в ответе.
 - в. 3DS – при сохранении карты выполнить проверку 3DS. Если карта поддерживает 3DS, выполняется списание и последующая отмена на 1 руб. В этом случае RebillID будет только для 3DS карт. Карты, не поддерживающие 3DS, привязаны не будут.

3DSHOLD – при привязке карты выполнить проверку поддержки картой 3DS. Если карта поддерживает 3DS, выполняется списание и последующая отмена на 1 руб. Если карта не поддерживает 3DS, выполняется списание и последующая отмена на произвольную сумму от 100 до 199 копеек. Клиент будет перенаправлен на экран для ввода списанной суммы, где должен корректно указать случайную сумму. В случае успешного подтверждения случайной суммы карта будет привязана и возвращен Rebill ID.

В результате ввода данных, к параметру CustomerKey будет привязана CardId.

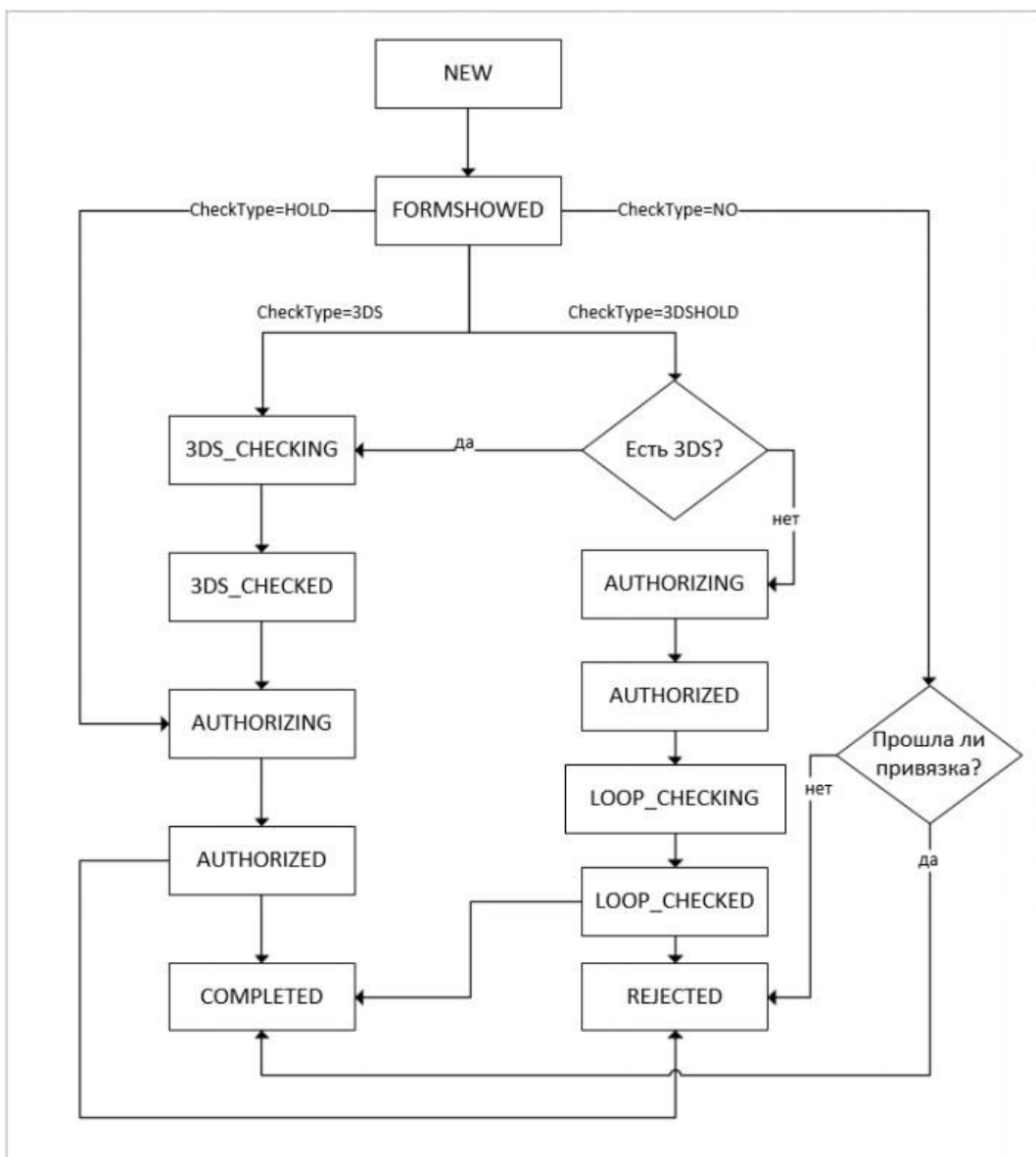


Рисунок 16. Статусная схема привязки карт

Описание статусов:

NEW — новая сессия;

FORMSHOWED — показ формы привязки карты;

3DS_CHECKING — отправка пользователя на проверку

3DS_CHECKED — пользователь успешно прошел проверку 3DS;

LOOP_CHECKING — пользователь отправлен на проверку блокирования случайной суммы;

LOOP_CHECKED — пользователь успешно прошел проверку блокирования случайной суммы;

AUTHORIZING — блокировка 1 рубля;

AUTHORIZED — успешно заблокировали и разблокировали 1 рубль; **COMPLETED** — привязка успешно завершена;

REJECTED — привязка отклонена.

9. Дополнительные возможности

9.1. Настройка стилей

В приложении есть базовая тема `AcquiringTheme`. Для изменения темы необходимо переопределить атрибуты в вашей теме, а также указать родителя `parent="AcquiringTheme"`

На всех экранах SDK используется одна и та же тема.

```
<item name="acqContentLayoutStyle">@style/AcquiringContentLayout</item>
<item name="acqToolbarStyle">@style/AcquiringToolbarStyle</item>
<item name="acqSavedCardsAddCardStyle">@style/AcquiringSavedCardsAddCardStyle</item>
<item name="acqSavedCardsDeleteCardStyle">@style/AcquiringSavedCardsDeleteCardStyle</item>
<item name="acqSavedCardsItemCardDataStyle">@style/AcquiringTitleTextStyle</item>
<item name="acqStaticQrMessageStyle">@style/AcquiringQrStaticMessage</item>
<item name="acqEmailStyle">@style/AcquiringEmailStyle</item>
<item name="acqButtonStyle">@style/AcquiringButtonStyle</item>
<item name="acqCardItemStyle">@style/AcquiringCardItemsStyle</item>
<item name="acqEditCardStyle">@style/AcquiringEditCardStyle</item>
<item name="acqErrorTitleStyle">@style/AcquiringTitleTextStyle</item>
<item name="acqScreenTitleLayoutStyle">@style/AcquiringScreenTitleLayoutStyle</item>
<item name="acqScreenTitleStyle">@style/AcquiringScreenTitleStyle</item>
<item name="acqOrderTitleStyle">@style/AcquiringOrderTitleStyle</item>
<item name="acqCardListTitleStyle">@style/AcquiringCardListTitleStyle</item>
<item name="acqOrderDescriptionStyle">@style/AcquiringOrderDescriptionStyle</item>
<item name="acqPaySecureIconStyle">@style/AcquiringPaySecureIconStyle</item>
<item name="acqSubmitAmountDescriptionStyle">@style/AcquiringSubmitAmountDescriptionStyle
</item>
<item name="acqSubmitAmountEditTextStyle">@style/AcquiringEditTextStyle</item>
<item name="acqViewHorizontalOffset">@dimen/acq_content_margin_horizontal</item>
<item name="acqScreenViewType">expandable</item>
```

Описание атрибутов приведено в таблице:

Таблица 9.1 Описание атрибутов стилей

Параметр	Описание
acqContentLayoutStyle	стиль основного контейнера контента
acqToolbarStyle	стиль тублара
acqSavedCardsAddCardStyle	стиль кнопки на экране сохраненных карт
acqSavedCardsDeleteCardStyle	стиль кнопки “Удалить карту” на экране сохраненных карт
acqSavedCardsItemCardDataStyle	стиль текста на элементах сохраненных карт
acqStaticQrMessageStyle	стиль сообщения на экране оплаты по статическому QR
acqEmailStyle	стиль полей ввода email
acqButtonStyle	стиль кнопок
acqCardItemStyle	стиль контейнеров элемента редактирования карты
acqEditCardStyle	стиль элемента редактирования карты
acqErrorTitleStyle	стиль сообщения об ошибке
acqScreenTitleLayoutStyle	стиль контейнера заголовка экранов
acqOrderTitleStyle	стиль названия товара
acqCardListTitleStyle	стиль заголовка на экране списка карт
acqOrderDescriptionStyle	стиль описания товара
acqPaySecureIconStyle	стиль иконок безопасного платежа
acqSubmitAmountDescriptionStyle	стиль описания на экране блокировки суммы
acqSubmitAmountEditTextStyle	стиль ввода заблокированной суммы
acqSbpBanksNotFoundTitle	стиль заголовка экрана информации о приложениях СБП
acqSbpBanksDescriptions	стиль заголовка виджета с приложениями СБП
acqViewHorizontalOffset	горизонтальные отступы элементов
acqScreenViewType	тип отображения экранов SDK. Возможные значения: <ul style="list-style-type: none"> expandable - вид карточки fullscreen - вид на полный экран

Пример отображения экрана оплаты при значении `acqScreenViewType="expandable"` и `acqScreenViewType="fullscreen"`:

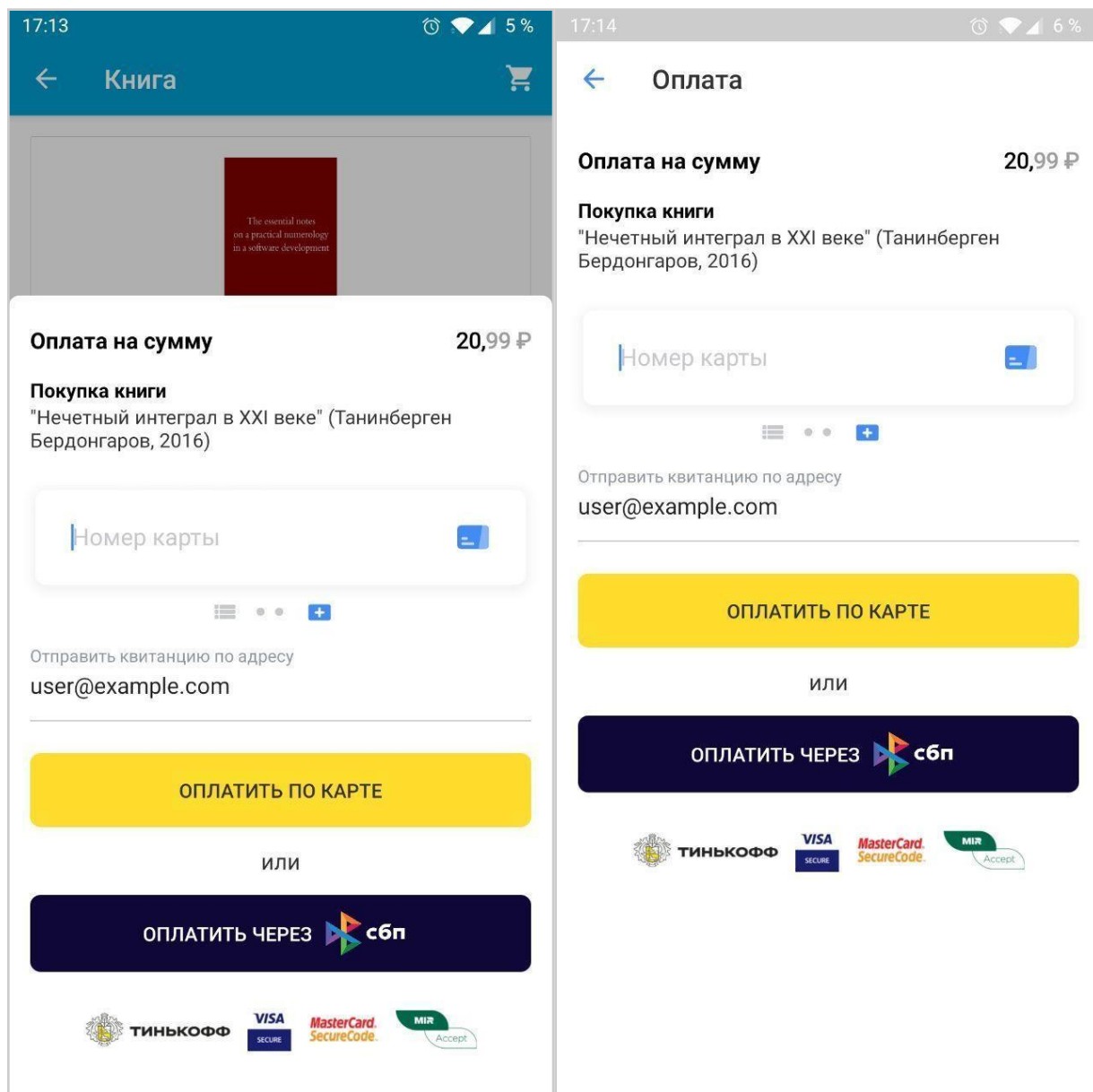


Рисунок 17. Виды отображения формы оплаты

9.2. Локализация

SDK имеет поддержку настройки локализации интерфейса, которая может не зависеть от локали устройства.

В SDK реализовано 2 языковые локализации - русская и английская. Чтобы установить локализацию из SDK, необходимо передать в параметр настройки экрана объект `AsdkSource` и указать его параметр `Language.RU` или `Language.EN`.

Пример:

```
var paymentOptions = PaymentOptions().setOptions {  
    orderOptions { /*options*/ }  
    customerOptions { /*options*/ }  
    featuresOptions {  
        localizationSource = AsdkSource(Language.RU)  
    }  
}
```

По умолчанию, если в `localizationSource` не задан, определяется и используется локаль устройства: для русскоговорящих стран и стран СНГ - русская локализация и английская для всех остальных.

Файлы локализации имеют формат `json`, где ключом является место использования строки на форме, значением - перевод.

Существует возможность добавить свои локализации на другие языки, для этого нужно:

- 1) Создать перевод строк файла `acq_localization_ru.json`, соблюсти формат файла `json` и наличие всех ключей.
- 2) Поместить файл в ресурсы вашего приложения.
- 3) Передать в параметр формы `localizationSource` тип ресурса, соответствующий размещению локализации в ресурсах.

Поддерживаются следующие типы расположения локализации:

- `FileSource` - принимает объект `File` с указанием пути до файла локализации
- `StringSource` - принимает строковый ресурс формата `json`
- `RawSource` - принимает ресурс `Android Raw Resources`

Пример:

```
var paymentOptions = PaymentOptions().setOptions {  
    orderOptions { /*options*/ }  
    customerOptions { /*options*/ }  
    featuresOptions {  
        localizationSource = RawSource(R.raw.custom_localization)  
    }  
}
```

9.3. Проведение платежа без открытия экрана оплаты

Для проведения платежа без открытия экрана необходимо вызвать метод `TinkoffAcquiring#initPayment`, который запустит процесс оплаты с инициацией и подтверждением платежа (будут вызваны методы API `Init` и `FinishAuthorize` или `Init` и `Charge`). В параметры метода необходимо передать карточные данные для оплаты или Google Pay токен, и параметры платежа `PaymentOptions`. Если разработчик самостоятельно вызвал метод `Init` и инициировал платеж, для запуска только подтверждения платежа существует метод `TinkoffAcquiring#finishPayment`. В этом случае в метод должен быть передан полученный `paymentId` из вызова `Init` и источник платежа `PaymentSource` - карточные данные или Google Pay токен.

За выполнение методов отвечает объект SDK `PaymentProcess`, который имеет возможность уведомлять слушателя о событиях в процессе выполнения, методы подписки и отписки от событий, а также методы старта и остановки процесса. Методы выполняются асинхронно.

Пример запуска платежа:

```
TinkoffAcquiring.initPayment(token, paymentOptions) // создание процесса платежа
    .subscribe(paymentListener)                    // подписка на события процесса
    .start()                                         // запуск процесса
```

9.4. Слушатель событий

Слушатель передается в метод подписки как реализация интерфейса `PaymentListener` или вспомогательного класса `PaymentListenerAdapter` для реализации методов интерфейса выборочно.

Методы интерфейса:

- `onSuccess(paymentId, cardId)` - вызывается в случае успешного платежа, возвращает `paymentId` операции и опционально `cardId` - id карты, с которой проводилась операция;
- `onUiNeeded(asdkState)` - вызывается, когда необходимо дополнительно обработать платеж на UI;
- `onError(throwable)` - возвращает ошибку, в случае её возникновения в процессе;
- `onStatusChanged(state)` - возвращает обновление статуса процесса.

В процессе оплаты могут возникнуть события, требующие обязательной обработки платежа на UI, например, прохождение 3DS. SDK имеет встроенную поддержку таких случаев. Чтобы обработать событие с помощью SDK, вызовите метод `TinkoffAcquiring#openPaymentScreen`, передав в его параметры `activity`, параметры оплаты `PaymentOptions`, `requestCode` для получения результата и `asdkState`, полученный в методе `onUiNeeded`.

В этом случае результат платежа вернется в вызывающую Activity в методе `onActivityResult`.

Пример реализации слушателя:

```
fun createPaymentListener(): PaymentListener {  
    return object : PaymentListenerAdapter() {  
  
        override fun onSuccess(paymentId: Long, cardId: String?) {  
            hideProgressDialog()  
            showSuccessDialog()  
        }  
  
        override fun onUiNeeded(state: AsdkState) {  
            hideProgressDialog()  
            tinkoffAcquiring.openPaymentScreen(  
                this@MyActivity,  
                paymentOptions,  
                PAYMENT_REQUEST_CODE,  
                state)  
        }  
  
        override fun onError(throwable: Throwable) {  
            hideProgressDialog()  
            showErrorDialog()  
        }  
    }  
}
```


10. Структура

SDK состоит из следующих модулей:

- Core;
- UI;
- Sample;
- CardIO.

Core является базовым модулем для работы с Tinkoff Acquiring API. Он содержит модель данных и позволяет осуществлять запросы к API с разбором ответов из JSON в имеющуюся модель. Запросы осуществляются синхронно. Модуль можно использовать отдельно от остальной SDK для реализации десктопных или веб-приложений.

Основной класс модуля **AcquiringSdk** - позволяет конфигурировать SDK и осуществлять взаимодействие с Тинькофф Эквайринг API. Для работы необходимы ключи и пароль продавца (см. [Подготовка к работе](#)). Методы, осуществляющие обращение к API, возвращают результат в случае успешного выполнения запроса или отдают исключение **AcquiringSdkException**.

UI содержит интерфейс, необходимый для приема платежей и привязки карты через мобильное приложение.

Основной класс - **TinkoffAcquiring** - позволяет конфигурировать SDK и осуществлять вызовы открытия экранов SDK, а также вызовы процесса оплаты без открытия экранов.

Доступные для вызова экраны:

PaymentActivity - экран с формой оплаты, который позволяет:

- просматривать информацию о платеже;
- вводить или сканировать реквизиты карты для оплаты;
- проходить 3DS подтверждение;
- управлять списком ранее сохраненных карт;
- совершать рекуррентный платеж, оплату по реквизитам карты, с помощью привязанной карты, через Систему быстрых платежей.

AttachCardActivity - экран с формой привязки карты

StaticQrActivity - экран содержащий статический QR код для приема оплаты через Систему быстрых платежей (см. [Прием платежей через Систему быстрых платежей](#))

SavedCardsActivity - экран со списком привязанных карт

Все строки, представленные в интерфейсе, имеют две дефолтные локализации: на русском и на английском. По умолчанию язык определяется на основании локализации системы. Язык можно указать, передав параметр в настройки экрана.

Существует возможность добавить локализации на нужные языки, передав в параметр языка свой ресурс (см. [Локализация](#))

Sample Содержит пример интеграции Tinkoff Acquiring SDK в мобильное приложение по продаже книг. Класс SessionParams содержит тестовые Terminal key, Пароль, Public key.

Приложение состоит из следующих экранов:

- Основной экран – экран со списком книг в продаже;
- Детали - экран с подробной информацией о товаре;
- Корзина - экран с набранными товарами;
- Подтверждение - экран об успешности / неуспешности операции покупки;
- О программе - информационный экран: версия SDK, EULA
- Настройки - служат примером настройки SDK
- Оплата по QR коду - экран для отображения статического QR кода Системы быстрых платежей

CardIO Модуль для сканирования карты камерой телефона с помощью библиотеки **Card-IO**.

Если вы хотите внедрить сканирование с помощью библиотеки **Card-IO**, то необходимо добавить в build.gradle:

```
implementation 'ru.tinkoff.acquiring:cardio:$latestVersion'
```

10.1. AcquiringSdk

Для программиста, SDK представлен классом AcquiringSdk, выполняющим запросы к API в синхронном режиме. Для создания инстанса класса нужно передать в конструктор необходимые параметры (выдаются мерчанту при подключении к эквайрингу):

- terminalKey - имя терминала продавца;
- password - пароль от терминала;
- publicKey - публичный ключ продавца.

Помимо переданных параметров в конструкторе можно настроить logger. Параметр позволяет управлять системой, осуществляющей запись логов. По умолчанию, если разработчик не установил свой логгер, используется реализация JavaLogger, основанная на функции System.out.println.

Для вызова API нужно вызвать соответствующий метод и передать ему на вход необходимые параметры. Для языка Kotlin реализован DSL для установки параметров.

Для каждого метода API есть соответствующий класс Request, который содержит: параметры, устанавливающие значения для реквеста:

- метод validate, проверяющий ограничения доменной модели;
- метод makeToken, вычисляющий подпись запроса (см. описание алгоритма ниже);
- метод execute, запускающий выполнение запроса к API и принимающий 2 колбэк функции
- - onSuccess, в параметр которой вернется Response ответ метода, и onFailure, в которую вернется ошибка

В общем случае вызов Api выглядит так:

1. Создается инстанс класса AcquiringSdk и конфигурируется.
2. На вход соответствующего метода передается сконфигурированный запрос.
3. Внутри метода создается билдер.
4. Вызывается execute метода
5. Выполняется синхронный вызов метода
6. Ответ из JSON парсится в класс модели Response.
7. В полученном ответе проверяется поле Success. В случае успеха возвращается результат в onSuccess, в случае ошибки возвращается ошибка в onFailure.

10.2. CryptoUtils

Содержит набор методов для работы с хешами и криптографией.

- sha256 - принимает на строку для вычисления хеша;
- encryptRsa - принимает на вход данные и шифрует их алгоритмом RSA/ECB/PKCS1Padding;
- encodeBase64 - принимает на вход данные и энкодит их в base64.

10.3. Алгоритм формирования подписи запроса (Token)

Для формирования подписи необходимо:

1. Собрать все параметры запроса Ключ-Значение, кроме параметра Token. Например, `[["TerminalKey", "TestB"], ["PaymentId", "20150"]]`.
2. Добавить туда пару Password-Значение. `[["TerminalKey", "TestB"], ["PaymentId", "20150"], ["Password", "123456789"]]`.
3. Отсортировать по ключам в алфавитном порядке. `[["Password", "123456789"], ["PaymentId", "20150"], ["TerminalKey", "TestB"]]`.
4. Конкатенировать значения. 12345678920150TestB.
5. Вычислить SHA-256 от пункта 4.

10.4. Алгоритм шифрования карточных данных

1. Введенные пользователем номер карты, expiry date и secure code приводятся к виду: `"PAN=%pan%;ExpDate=%month%year%;CVV=%secure_code%"`.
2. Выполняется шифрование строк с шага 1 алгоритмом RSA/ECB/PKCS1Padding с использованием publicKey в качестве ключа.
3. Полученная криптограмма на шаге 2 кодируется алгоритмом Base64.

11. Методы API

AcquiringApi позволяет выполнять синхронные запросы к API. В качестве параметра на вход принимает соответствующий Request. В ответ возвращает Response, распарсенный из JSON ответа сервера.

Поддерживаемые методы:

- Метод Init;
- Метод FinishAuthorize;
- Метод Charge;
- Метод GetState;
- Метод AddCard;
- Метод AttachCard;
- Метод GetCardList;
- Метод RemoveCard;
- Метод GetQr;
- Метод GetStaticQr.

11.1. Метод Init

Описание: Иницирует новый платеж. **URL:**

<https://securepay.tinkoff.ru/v2/Init>

Метод: POST

Таблица 11.1.1 Параметры запроса

Параметр	Тип	Обязательный	Описание
TerminalKey	String	Да	Идентификатор терминала, выдается Продавцу Банком
Amount	Number	Да	Сумма в копейках
OrderId	String	Да	Номер заказа в системе Продавца
IP	String	Нет	IP-адрес клиента
Description	String	Нет	Краткое описание
Currency	Number	Нет	Код валюты ISO 4217 (например, 643). Если передан Currency, и он разрешен для Продавца, транзакция будет инициирована в переданной валюте. Иначе будет использована валюта по умолчанию для данного терминала
Token	String	Да	Подпись запроса

PayForm	String	Нет	Название формы оплаты продавца
CustomerKey	String	Нет	Идентификатор покупателя в системе Продавца. Если передается и Банком разрешена автоматическая привязка карт к терминалу, то для данного покупателя будет осуществлена привязка карты. Тогда в нотификации на AUTHORIZED будет передан параметр CardId, подробнее см. метод GetCardList
Language	String	Нет	Язык платёжной формы. ru - форма оплаты на русском языке; en - форма оплаты на английском языке. По умолчанию (если параметр не передан) - форма оплаты на русском языке.
Recurrent	String	Нет	Если передается и установлен в Y, то регистрирует платеж как рекуррентный. В этом случае после оплаты в нотификации на AUTHORIZED будет передан параметр RebillId для использования в методе Charge
RedirectDueDate	Datetime	Нет	Срок жизни ссылки. В случае, если текущая дата превышает дату, переданную в данном параметре, ссылка для оплаты становится недоступной и платёж выполнить нельзя. Формат даты: YYYY-MM-DDTHH24:MI:SS+GMT Пример даты: 2016-08-31T12:28:00+03:00
DATA*	Object	Нет	JSON объект содержащий дополнительные параметры в виде «ключ»: «значение». Данные параметры будут переданы на страницу оплаты. Максимальная длина для каждого передаваемого параметра: Ключ – 20 знаков, Значение – 100 знаков. Максимальное количество пар «ключ-значение» не может превышать 20(*)
Receipt	Object	Нет	JSON объект с данными чека

(*) В случае, если у терминала включена опция привязки покупателя после успешной оплаты и передается параметр CustomerKey, то в передаваемых параметрах DATA могут присутствовать параметры команды AddCustomer. Если они присутствуют, они автоматически привязываются к покупателю.

Например, если указать:

```
"DATA":{"Phone":"+71234567890", "Email":"a@test.com"}
```



к покупателю автоматически будут привязаны данные Email и телефон, и они будут возвращаться при вызове метода GetCustomer.

Таблица 11.1.2. Структура объекта Receipt

Наименование	Тип	Обязательный	Описание
Items	Массив объектов	Да	Массив, содержащий в себе информацию о товарах
Email	String	Нет	Электронная почта
Phone	String	Да	Телефон
Taxation	Перечисление (Enum)	Да	<p>Система налогообложения. Перечисление со значениями:</p> <ul style="list-style-type: none"> • «osn» – общая СН; • «usn_income» – упрощенная СН (доходы); • «usn_income_outcome» упрощенная СН (доходы минус расходы); • «envd» – единый налог на вмененный доход; • «esn» единый сельскохозяйственный налог; • «patent» – патентная СН

Таблица 11.1.3. Структура объекта Items

Наименование	Тип	Обязательный	Описание
Name	String	Да	Наименование товара. Максимальная длина строки – 128 символов
Price	Number	Да	Цена в копейках. Целочисленное значение не более 10 знаков
Quantity	Number	Да	Количество/вес: <ul style="list-style-type: none"> целая часть не более 8 знаков; дробная часть не более 3 знаков
Amount	Number	Да	Сумма в копейках. Целочисленное значение не более 10 знаков
Tax	Перечисление (Enum)	Да	Ставка налога. Перечисление со значениями: <ul style="list-style-type: none"> «none» – без НДС; «vato» – НДС по ставке 0%; «vat10» – НДС чека по ставке 10%; «vat18» – НДС чека по ставке 18%; «vat110» – НДС чека по расчетной ставке 10/110; «vat118» – НДС чека по расчетной ставке 18/118
Ean13	String	Нет	Штрих-код
ShopCode	String	Нет	Код магазина. Для параметра ShopCode необходимо использовать значение параметра Submerchant_ID, полученного в ответ при регистрации магазинов через xml. Если xml не используется, передавать поле не нужно

Пример запроса:

```
{
  "TerminalKey": "TestB",
  "Amount": "140000",
  "OrderId": "21050",
  "Description": "Подарочная карта на 1000 рублей",
  "Token": "2ED30E046136931431B5251B7C9A1EAC68DAB082203BD42676BA14A851359DF4"
  "DATA": {"Phone": "+71234567890", "Email": "a@test.com"},
  "Receipt": {
    "Email": "a@test.ru",
    "Phone": "+79031234567",
    "Taxation": "osn",
    "Items": [
      {
        "Name": "Наименование товара 1",
        "Price": 10000,
        "Quantity": 1.00,
        "Amount": 10000,
        "Tax": "vat10",
        "Ean13": "0123456789",
        "ShopCode": "12345"
      },
      {
        "Name": "Наименование товара 2",
        "Price": 20000,
        "Quantity": 2.00,
        "Amount": 40000,
        "Tax": "vat18"
      }
    ]
  }
}
```


Таблица 11.1.4. Параметры ответа

Наименование	Тип	Обязательный	Описание
TerminalKey	String	Да	Идентификатор терминала, выдается Продавцу Банком
Amount	Number	Да	Сумма в копейках
OrderId	String	Да	Номер заказа в системе Продавца
Success	Boolean	Да	Успешность операции
Status	String	Да	Статус транзакции
PaymentId	Number	Да	Уникальный идентификатор транзакции в системе Банка
ErrorCode	String	Да	Код ошибки, «0» - если успешно
PaymentURL	String	Нет	Ссылка на страницу оплаты. Не передается, если ввод данных карты осуществляется на стороне Продавца. Доступна в течении 24 часов по умолчанию
Message	String	Нет	Краткое описание ошибки
Details	String	Нет	Подробное описание ошибки

Пример ответа:

```
{
  "Success":true,
  "ErrorCode":"0",
  "TerminalKey":"TestB",
  "Status":"NEW",
  "PaymentId":"13660",
  "OrderId":"21050",
  "Amount":"100000", "PaymentURL":"https://rest-api-test.tcsbank.ru/rest/Authorize/1b63d14a-4208-44a8-a288-ad1b04008e51"
}
```

Статус платежа:

при успешном сценарии: **NEW**;

при неуспешном: **REJECTED**.

11.2. Метод FinishAuthorize

Описание: подтверждает инициированный платеж передачей карточных данных. При использовании одностадийного проведения осуществляет списание денежных средств с карты покупателя.

При двухстадийном проведении осуществляет блокировку указанной суммы на карте покупателя.

URL: <https://securepay.tinkoff.ru/v2/FinishAuthorize>

Метод: POST

Таблица 11.2.1. Параметры запроса

Наименование	Тип	Обязательный	Описание
TerminalKey	String	Да	Идентификатор терминала, выдается Продавцу Банком
PaymentId	Number	Да	Уникальный идентификатор транзакции в системе Банка, полученный в ответе на вызов метода Init
CardData(*)	String	Да	Зашифрованные данные карты в формате: "PAN=%pan%;ExpDate=%month%year%;CVV=%secure_code%" при оплате по полным реквизитам; "CardId=%id%;CVV=%secure_code%" при оплате с сохраненной карты
DATA	Object	Нет	JSON объект содержащий дополнительные параметры в виде "ключ": "значение". Данные параметры будут переданы на страницу оплаты. Максимальная длина для каждого передаваемого параметра: Ключ – 20 знаков, Значение – 100 знаков. Максимальное количество пар «ключ- значение» не может превышать 20
IP	String	Нет	IP-адрес клиента
Phone	String	Нет	Телефон клиента
SendEmail	Boolean	Нет	true – отправлять клиенту информацию на почту об оплате; false – не отправлять
InfoEmail	String	Нет	Email для отправки информации об оплате
Token	String	Да	Подпись запроса



Продавец собирает поле CardData в виде списка «ключ=значение» с разделителем «;» и зашифровывает его выданным при регистрации терминала открытым ключом (X509 RSA 2048).

Таблица 11.2.1 Параметры CardData

Наименование	Тип	Обязательный	Описание
PAN	Number	Да	Номер карты
ExpDate	Number	Да	Месяц и год срока действия карты в формате ММYY
CardHolder	String	Нет	Имя и фамилия держателя карты (как на карте)
CVV	String	Да	Код защиты (с обратной стороны карты)

Пример значения элемента формы CardData:

“PAN=4300000000000777;ExpDate=0519;CardHolder=IVAN PETROV;CVV=111”

Пример запроса:

```
{
  "TerminalKey": "TestB"
  "PaymentId": "10063"
  "CardData": "b3tSlUYwsf3Erdv5ReB7WpWK3/NBWLlWdiSLjQG0cBxA0Mgs7ALd7edi0RbVlORsyGZEUJS1RynQ9zL
MyHYzWP3z2sQYGA vzOqufoVPe2AozhW3pZV+dN5s7oGcpXd39NDC0Ma/Zw6oa3dJR0Zh8QYjv/sG0zU1lMjXl5aHg
Tpxk37q6OxUakxuG7euhvSN71JqxHsNEuoJELAQlQ7U+3tuh9AjTuiBpmEH99maK9e7gnVXgZd1Nk8vachs97xj9cL
/023qYMk7CMjldBfG4V0sYVqcHsKfbbJJ8CZXIJgmXhCYns1hmRD/kf3OhEZr038LghC7Iio0yxHYMhZyJoQ=="
  "Token": "871199b37f207f0c4f721a37cdcc71dfcea880b4a4b85e3cf852c5dc1e99a8d6"
}
```

Формат ответа: JSON.

Таблица 11.2.2 Параметры ответа

Наименование	Тип	Обязательный	Описание
TerminalKey	String	Да	Идентификатор терминала, выдается Продавцу Банком
Amount	Number	Да	Сумма в копейках
OrderId	String	Да	Номер заказа в системе Продавца
RebillId	String	Нет	Идентификатор рекуррентного платежа
CardId	String	Нет	Идентификатор карты в системе Банка. Передается только для сохраненной карты
Success	Boolean	Да	Успешность операции
Status(*)	String	Да	Статус транзакции
PaymentId	Number	Да	Уникальный идентификатор транзакции в системе Банка
ErrorCode	String	Да	Код ошибки, «0» - если успешно
Message	String	Нет	Краткое описание ошибки
Details	String	Нет	Подробное описание ошибки

Пример ответа:

```
{
  "Success":true,
  "ErrorCode":"0",
  "TerminalKey":"TestB",
  "Status":"CONFIRMED",
  "PaymentId":"10 063",
  "OrderId":"21050",
  "Amount":100000
}
```

(*) Статус платежа:

- при успешном сценарии и одностадийном проведении платежа: **CONFIRMED**
- при успешном сценарии и двухстадийном проведении платежа: **AUTHORIZED**
- при неуспешном: **REJECTED**
- при необходимости прохождения проверки 3-D Secure: **3DS_CHECKING**

11.3. Метод Charge

Описание: осуществляет рекуррентный (повторный) платеж — безакцептное списание денежных средств со счета банковской карты Покупателя. Для возможности его использования Покупатель должен совершить хотя бы один платеж в пользу Продавца, который должен быть указан как рекуррентный (см. параметр Recurrent в методе Init), фактически являющийся первичным. По завершении оплаты такого платежа в нотификации на AUTHORIZED будет передан параметр RebillId. В дальнейшем при совершении рекуррентного платежа Продавец должен инициировать его, вызвав метод Init, а затем, не осуществляя переадресации на PaymentURL, вызвать метод Charge для оплаты по тем же самым реквизитам передав параметр RebillId, полученный при совершении первичного платежа. Независимо от установленного типа проведения платежа, метод Charge всегда работает по типу одностадийного проведения. Это значит, что во время выполнения метода Charge на Notification URL будет отправлен синхронный запрос (подробнее см. Нотификация продавца об операциях), на который требуется корректный ответ.

Другими словами, для использования рекуррентных платежей необходима следующая последовательность действий:

1. Совершить родительский платеж путем вызова Init с указанием дополнительного параметра Recurrent=Y.
2. Переадресовать Покупателя на PaymentUrl.
3. После оплаты покупателем в нотификации на AUTHORIZED будет передан параметр RebillId, который необходимо сохранить.
4. Спустя некоторое время для совершения рекуррентного платежа необходимо вызвать метод Init со стандартным набором параметров (параметр Recurrent здесь не нужен).
5. Получить в ответ на Init параметр PaymentId, при этом переадресацию пользователя на PaymentUrl производить не надо.
6. Вызвать метод Charge с параметром RebillId полученным в п.3 и параметром PaymentId полученным в п.5.

URL: <https://securepay.tinkoff.ru/v2/Charge>

Метод: POST

Таблица 11.3.1 Параметры запроса

Наименование	Тип	Обязательный	Описание
TerminalKey	String	Да	Идентификатор терминала, выдается Продавцу Банком
PaymentId	Number	Да	Уникальный идентификатор транзакции в системе Банка, полученный в ответе на вызов метода Init
IP	String	Нет	IP-адрес клиента
RebillId	Number	Да	Идентификатор рекуррентного платежа (см. параметр Recurrent в методе Init)
Token	String	Да	Подпись запроса.
SendEmail	Boolean	Нет	true – если покупатель хочет получать уведомления на почту
InfoEmail	String	Нет	Адрес почты покупателя

Пример запроса:

```
{
  "TerminalKey": "TestB",
  "PaymentId": "10063",
  "RebillId": "145919",
  "Token": "871199b37f207f0c4f721a37cdcc71dfcea880b4a4b85e3cf852c5dc1e99a8d6"
}
```

Формат ответа: JSON

Таблица 11.3.2 Параметры ответа

Наименование	Тип	Обязательный	Описание
TerminalKey	String	Да	Идентификатор терминала, выдается Продавцу Банком
Amount	Number	Да	Сумма в копейках
OrderId	String	Да	Номер заказа в системе Продавца
Success	Boolean	Да	Успешность операции
Status	String	Да	Статус транзакции
PaymentId	Number	Да	Уникальный идентификатор транзакции в системе Банка
ErrorCode	String	Да	Код ошибки, «0» - если успешно
Message	String	Нет	Краткое описание ошибки
Details	String	Нет	Подробное описание ошибки

Пример ответа:

```
{
  "Success":true,
  "ErrorCode":"0",
  "TerminalKey":"TinkoffBankTest",
  "Status":"CONFIRMED",
  "PaymentId":"63100",
  "OrderId":"100668",
  "Amount":444
}
```

11.4. Метод GetState

Описание: Возвращает статус платежа **URL:**

<https://securepay.tinkoff.ru/v2/GetState>

Метод: POST

Таблица 11.4.1 Параметры запроса

Наименование	Тип	Обязательный	Описание
TerminalKey	String	Да	Идентификатор терминала, выдается Продавцу Банком
PaymentId	Number	Да	Уникальный идентификатор транзакции в системе Банка, полученный в ответе на вызов метода Init
IP	String	Нет	IP-адрес клиента
Token	String	Да	Подпись запроса.

Пример запроса:

```
{
  "TerminalKey": "TestB",
  "PaymentId": "10063",
  "Token": "871199b37f207f0c4f721a37cdcc71dfcea880b4a4b85e3cf852c5dc1e99a8d6"
}
```

Формат ответа: JSON

Таблица 11.4.2. Параметры ответа

Наименование	Тип	Обязательный	Описание
TerminalKey	String	Да	Идентификатор терминала, выдается Продавцу Банком
Amount	Number	Да	Сумма в копейках
OrderId	String	Да	Номер заказа в системе Продавца
Success	Boolean	Да	Успешность операции
Status	String	Да	Статус транзакции
PaymentId	Number	Да	Уникальный идентификатор транзакции в системе Банка
ErrorCode	String	Да	Код ошибки, «0» - если успешно
Message	String	Нет	Краткое описание ошибки
Details	String	Нет	Подробное описание ошибки

Пример ответа:

```
{
  "TerminalKey": "TestB",
  "OrderId": "21057",
  "Success": true,
  "Status": "NEW",
  "PaymentId": "10063",
  "ErrorCode": "0"
}
```

Таблица 11.4.3. Возможные статусы транзакции

Статус	Промежуточный	Значение
AUTHORIZED	Нет	Средства заблокированы, но не списаны
3DS_CHECKING	Нет	Покупатель начал аутентификацию по протоколу 3D Secure
CONFIRMED	Нет	Денежные средства списаны

11.5. Метод AddCard

Описание: Иницирует привязку карты к покупателю.

URL: <https://securepay.tinkoff.ru/v2/AddCard>

Метод: POST

Таблица 11.5.1. Параметры запроса

Наименование	Тип	Обязательный	Описание
TerminalKey	String	Да	Идентификатор терминала, выдается Продавцу Банком
CustomerKey	String	Да	Идентификатор покупателя в системе Продавца
CheckType	String	Нет	<p>Возможные значения:</p> <p>NO – сохранить карту без проверок. Rebill ID для рекуррентных платежей не возвращается.</p> <p>HOLD – при сохранении сделать списание и затем отмену на 1 руб. RebillID для рекуррентных платежей возвращается в ответе.</p> <p>3DS – при сохранении карты выполнить проверку 3DS. Если карта поддерживает 3DS, выполняется списание и последующая отмена на 1 руб. В этом случае RebillID будет только для 3DS карт. Карты, не поддерживающие 3DS, привязаны не будут.</p> <p>3DSHOLD – при привязке карты выполнить проверку поддержки картой 3DS. Если карта поддерживает 3DS, выполняется списание и последующая отмена на 1 руб. Если карта не поддерживает 3DS, выполняется списание и последующая отмена на произвольную сумму от 100 до 199 копеек. Клиент будет перенаправлен на страницу для ввода списанной суммы, где должен корректно указать случайную сумму</p>
Description	String	Нет	Описание/название карты
PayForm	String	Нет	Название шаблона формы привязки
IP	String	Нет	IP-адрес клиента
Token	String	Да	Подпись запроса

Пример запроса:

```
{
  "TerminalKey": "1201253242594",
  "CustomerKey": "Test-112",
  "Token": "2ED30E046136931431B5251B7C9A1EAC68DAB082203BD42676BA14A851359DF4"
}
```

Формат ответа: JSON

Таблица 11.5.2. Параметры запроса

Наименование	Тип	Обязательный	Описание
TerminalKey	String	Да	Идентификатор терминала, выдается Продавцу Банком
CustomerKey	String	Да	Идентификатор покупателя в системе Продавца
RequestKey	String	Да	Идентификатор запроса на привязку карты
Success	Boolean	Да	Успешность операции
PaymentURL	String	Нет	Ссылка на страницу привязки карты. На данную страницу необходимо переадресовать клиента для привязки карты
ErrorCode	String	Да	Код ошибки, «0» - если успешно
Message	String	Нет	Краткое описание ошибки
Details	String	Нет	Подробное описание ошибки

Пример ответа:

```
{
  "Success": true,
  "ErrorCode": "0",
  "TerminalKey": "1485466639730",
  "CustomerKey": "906540",
  "PaymentURL": "https://rest-api-test.tinkoff.ru/AddCard/82a31a62-6067-4ad8-b379-04bf13e37642",
  "RequestKey": "ed989549-d3be-4758-95c7-22647e03f9ec"
}
```

11.6. Метод AttachCard

Описание: Завершает привязку карты к покупателю. Метод вызывается автоматически после метода AddCard.

URL: <https://securepay.tinkoff.ru/v2/AttachCard>

Метод: POST

Таблица 11.6.1 Параметры запроса

Наименование	Тип	Обязательный	Описание
TerminalKey	String	Да	Идентификатор терминала, выдается Продавцу Банком
RequestKey	String	Да	Идентификатор запроса на привязку карты
CardData	String	Да	Зашифрованные данные карты в формате: "PAN=%pan%;ExpDate=%month%year%;CVV=%secure_co de%"
DATA	Object	Нет	<p>Ключ = значение дополнительных параметров через " ", например, Email = a@test.ru Phone = +71234567890.</p> <p>Если ключи или значения содержат в себе спецсимволы, то получившееся значение должно быть закодировано функцией urlencode.</p> <p>Максимальная длина для каждого передаваемого параметра:</p> <p>Ключ – 20 знаков, Значение – 100 знаков.</p> <p>Максимальное количество пар «ключ- значение» не может превышать 20</p>
Token	String	Да	Подпись запроса

Пример запроса:

```
{ "TerminalKey": "testRegress",
  "CardData": "U5jDbwqOVx+2vDapxe/rfACMt+rFWXzPdJ8ZXxNFViiZaLZrOW72bGe9cKZdIDnekW0nqm88YxRDj
yfa5Ru0kY5cQValU+juS1ulzepamSDtaGFeb8sRZfhj72yGw+io+qHGSBeorcfgoKStyKGuBPWfGd0PLHuyBE6QgZy
IAM1XfdmNlV0UAxOnkTGDsskLpIt3AWhw2e8KOar0vwbGCTDjznDB1/DLgOW01dAj/bXyLJoG1BkOrPBm9JURs+f+uy
Fae0hkRicNKNgXoN5pJTSQxOEauOi6ylsVJB3WK5MNYXtj6x4GlxcmTk/LD9kvHcjTeojcAlDzRZ87GdWeY8wgg=",
  "RequestKey": "13021e10-a3ed-4f14-bcd1-823b5ac37390",
  "Token": "7241ac8307f349afb7bb9dda760717721bbb45950b97c67289f23d8c69cc7b96",
  "DATA": { "Email": "a@test.com" }
}
```

Формат ответа: JSON

Таблица 11.6.2 Параметры ответа

Наименование	Тип	Обязательный	Описание
TerminalKey	String	Да	Идентификатор терминала, выдается Продавцу Банком
CustomerKey	String	Да	Идентификатор покупателя в системе Продавца
RequestKey	String	Да	Идентификатор запроса на привязку карты
RebillId	String	Нет	Идентификатор рекуррентного платежа
CardId	String	Да	Идентификатор карты в системе Банка
Success	Boolean	Да	Успешность операции
Status	String	Да	Статус привязки карты
ErrorCode	String	Да	Код ошибки, «0» - если успешно
Message	String	Нет	Краткое описание ошибки
Details	String	Нет	Подробное описание ошибки

Пример ответа:

```
{
  "Success": true,
  "ErrorCode": "0",
  "TerminalKey": "testRegress",
  "Status": "3DS_CHECKING",
  "CustomerKey": "testRegress5",
  "RequestKey": "8de92934-26c9-474c-a4ce-424f2021d24d"
  "CardId": "5555"
}
```

11.7. Метод GetCardList

Описание: Возвращает список привязанных карт у покупателя

URL: <https://securepay.tinkoff.ru/v2/GetCardList>

Метод: POST

Таблица 11.7.1 Параметры запроса

Наименование	Тип	Обязательный	Описание
TerminalKey	String	Да	Идентификатор терминала, выдается Продавцу Банком
CustomerKey	String	Да	Идентификатор покупателя в системе Продавца
IP	String	Нет	IP-адрес клиента
Token	String	Да	Подпись запроса

Пример запроса:

```
{
  "TerminalKey": "TestB"
  "CustomerKey": "Customer1"
  "Token": "871199b37f207f0c4f721a37cdcc71dfcea880b4a4b85e3cf852c5dc1e99a8d6"
}
```

Формат ответа: Массив JSON

Таблица 11.7.2 Параметры ответа

Наименование	Тип	Обязательный	Описание
Pan	String	Да	Номер карты маскированный
CardId	String	Да	Идентификатор карты в системе Банка
Status	String	Да	Статус карты: А - активная, I - неактивная, D - удалена.
RebillId	Number	Да	Идентификатор рекуррентного платежа (см. параметр Recurrent в методе Init)
ExpDate	String	Нет	Срок действия карты

Пример ответа:

```
[{
  "CardId": "4750",
  "Pan": "543211*****4773",
  "Status": "A",
  "RebillId": "145919"
},
{
  "CardId": "5100",
  "Pan": "411111*****1111",
  "Status": "I",
  "RebillId": "145917"
}]
```

11.8. Метод RemoveCard

Описание: Удаляет привязанную карту у покупателя

URL: <https://securepay.tinkoff.ru/v2/RemoveCard>

Метод: POST

Таблица 11.8.1 Параметры запроса

Наименование	Тип	Обязательный	Описание
TerminalKey	String	Да	Идентификатор терминала, выдается Продавцу Банком
CardId	Number	Да	Идентификатор карты в системе Банка
CustomerKey	String	Да	Идентификатор покупателя в системе Продавца
IP	String	Нет	IP-адрес клиента
Token	String	Да	Подпись запроса

Пример запроса:

```
{
  "TerminalKey": "TestB"
  "CardId": "4750"
  "CustomerKey": "Customer1"
  "Token": "871199b37f207f0c4f721a37cdcc71dfcea880b4a4b85e3cf852c5dc1e99a8d6"
}
```

Формат ответа: JSON

Таблица 11.8.2 Параметры запроса

Наименование	Тип	Обязательный	Описание
TerminalKey	String	Да	Идентификатор терминала, выдается Продавцу Банком
CustomerKey	String	Да	Идентификатор покупателя в системе Продавца
CardId	Number	Да	Идентификатор карты в системе Банка
Success	Boolean	Да	Успешность операции
Status	String	Да	Статус карты: D – удалена
ErrorCode	String	Да	Код ошибки, «0» - если успешно
Message	String	Нет	Краткое описание ошибки
Details	String	Нет	Подробное описание ошибки

Пример ответа:

```
{
  "CardId": "4750",
  "Status": "D",
  "Success": true,
  "ErrorCode": "0",
  "TerminalKey": "TestB",
  "CustomerKey": "Customer1"
}
```


11.9. Метод GetQr

Описание: регистрирует QR и возвращает информацию о нем. Должен быть вызван после вызова метода Init.

URL: <https://securepay.tinkoff.ru/v2/GetQr>

Метод: POST

Таблица 11.9.1 Параметры запроса

Наименование	Тип	Обязательный	Описание
TerminalKey	String	Да	Идентификатор терминала, выдается Продавцу Банком
PaymentId	Number	Да	Уникальный идентификатор транзакции в системе Банка
DataType	String	Нет	Тип возвращаемых данных PAYLOAD – В ответе возвращается только Payload (по-умолчанию) IMAGE – В ответе возвращается SVG изображение QR
Token	String	Да	Подпись запроса

Пример запроса:

```
{
  "TerminalKey": "TinkoffBankTest",
  "PaymentId": "10063",
  "Token": "871199b37f207f0c4f721a37cdcc71dfcea880b4a4b85e3cf852c5dc1e99a8d6"
}
```

Формат ответа: JSON

Таблица 11.9.2 Параметры запроса

Наименование	Тип	Обязательный	Описание
TerminalKey	String	Да	Идентификатор терминала, выдается Продавцу Банком
OrderId	String	Да	Номер заказа в системе Продавца
Success	Boolean	Да	Успешность операции (true/false)
Data	String	Да	В зависимости от параметра DataType в запросе это: Payload - информация, которая должна быть закодирована в QR или SVG изображение QR в котором уже закодирован Payload
PaymentId	Number	Да	Уникальный идентификатор транзакции в системе Банка
ErrorCode	String	Да	Код ошибки, «0» - если успешно
Message	String	Нет	Краткое описание ошибки
Details	String	Нет	Подробное описание ошибки

Пример ответа:

```
{
  "TerminalKey": "TinkoffBankTest",
  "OrderId": "21057",
  "Success": true,
  "Data": "https://qr.nspk.ru/AS1000670LSS7DN18SJQDNP4B05KLJL2?type=01&bank=100000000001&sum=10000&cur=RUB&crc=C08B",
  "PaymentId": "10063",
  "ErrorCode": "0"
}
```

11.10. Метод GetStaticQr

Описание: При первом вызове регистрирует QR и возвращает информацию о нем при последующих вызовах возвращает информацию о ранее сгенерированном QR. Перерегистрация статического QR происходит только при смене расчетного счета. Не привязан к конкретному платежу, может быть вызван в любое время без предварительного вызова Init.

URL: <https://securepay.tinkoff.ru/v2/GetStaticQr>

Метод: POST

Таблица 11.10.1 Параметры запроса

Наименование	Тип	Обязательный	Описание
TerminalKey	String	Да	Идентификатор терминала, выдается Продавцу Банком
Data	String	Нет	Тип возвращаемых данных PAYLOAD – В ответе возвращается только Payload (по-умолчанию) IMAGE – В ответе возвращается SVG изображение QR

Пример запроса:

```
{
  "TerminalKey": "TinkoffBankTest",
  "Data": "PAYLOAD"
}
```

Формат ответа: JSON

Таблица 11.10.2 Параметры запроса

Наименование	Тип	Обязательный	Описание
TerminalKey	String	Да	Идентификатор терминала, выдается Продавцу Банком
Success	Boolean	Да	Успешность операции (true/false)
Data	String	Да	В зависимости от параметра DataType в запросе это: Payload - информация, которая должна быть закодирована в QR или SVG изображение QR в котором уже закодирован Payload
ErrorCode	String	Да	Код ошибки, «0» - если успешно
Message	String	Нет	Краткое описание ошибки
Details	String	Нет	Подробное описание ошибки

Пример ответа:

```
{
  "TerminalKey": "TinkoffBankTest",
  "Success": true,
  "Data": "https://qr.nspk.ru/AS1000670LSS7DN18SJQDNP4B05KLJL2?type=01&bank=100000000001&sum=10000&cur=RUB&crc=C08B"
  "ErrorCode": "0"
}
```

12. Коды ошибок API и возможные исключения

Некоторые ошибки SDK обрабатывает самостоятельно, показывая экран с ошибкой “Что-то пошло не так” или описанием ошибки, с возможностью повторить операцию конечным пользователям. Остальные ошибки возвращаются на вызываемый экран.

Ошибки работы SDK возвращаются как исключения `AcquiringSdkException`. API может возвращать следующие ошибки, описание которых доступно по [ссылке](#).

13. Поддержка

Github: <https://github.com/TinkoffCreditSystems/AcquiringSdkAndroid>

Баги и feature-реквесты можно направлять в раздел `issues`.

Подробное описание методов: [API методов](#)