УДК 004

Интеграция Google Pay в существующее приложение для Android

Ульянов Егор Андреевич
Приамурский государственный университет имени Шолом-Алейхема
Студент

Аннотация
В данной статье рассматривается метод интеграции сервиса GooglePay в android приложение. Разработка будет происходить в среде программирования AndroidStudio. Практическим результатом является мобильное приложение с подключенным сервисом GooglePay.
Ключевые слова: google, приложение, андроид, оплата

Integration of Google Pay into an existing Android app

Ulianov Egor Andreevich
Sholom-Aleichem Priamursky State University
Student

Abstract
This article discusses a method for integrating the GooglePay service into an android application. Development will take place in the AndroidStudio programming environment. The practical result is a mobile application with a connected GooglePay service.
Keywords: google, app, android, pay

Практически каждый человек играет в мобильные игры и некоторые из таких людей покупают на свои аккаунты дополнительную помощь, это может быть жизнь в игре, либо улучшение оружие, в зависимости от игры. И это делается через внутренний магазин в игре. Конечно, клиент может выбрать свою карту и ввести свои данные, но почему бы не упростить задачу для клиента, используя платежные реквизиты, которые уже существуют в их мобильном телефоне, например GooglePay. Внедрение подобных платежных шлюзов в приложение увеличит шансы того, что пользователь действительно заплатит. Почти каждый пользователь Android, так или иначе, по какой-то причине зависит от Google, например от Google Assistant для напоминаний, Keep для заметок, а недавно и от Google Pay для платежей.

Цель данной статьи внедрить в существующее мобильное приложение возможность оплачивать продукт с помощью «google pay».

А.А. Сковорода и Д.Ю. Гамаюнов рассмотрели множество мобильных приложений с вредоносной базой и предложили методы для автоматической классификации приложений [1]. В своей работе К.В.Рубинов протестировал

большое множество интерфейсов в мобильных приложении и разработал свой подход к тестированию [2]. В.Г.Орлов и С.С. Шаврин рассмотрели мобильные приложения для использования в системах мониторинга и диспетчеризации технологических служб [3]. С.В. Глазков и А.Л. Ронжин рассмотрели способы формализации и анализа контекста приложений, исполняемых на мобильных гетерогенных устройствах [4].

Начнем с использования API Google Pay для интеграции платежного шлюза, для этого необходимо прописать строку под «dependencies» узлом в «build.gradle» файле уровня приложения (рис.1).

```
implementation "com.google.android.gms:play-services-wallet:18.0.0"
```

Рисунок 1 – добавление в gradle

Так же нужно добавить следующие метаданные в файл манифеста, чтобы включить API (рис.2).

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

Рисунок 2 – добавление мета данных

Теперь, когда закончили интеграцию, пришло время настроить API, для этого необходимо создать экземпляр клиентского объекта, через который можно выполнять вызовы Google Pay API (рис.3).

```
1  private lateinit var paymentsClient: PaymentsClient
2  fun createPaymentsClient(activity: Activity): PaymentsClient {
3      val walletOptions = Wallet.WalletOptions.Builder()
4              .setEnvironment(WalletConstants.ENVIRONMENT_TEST).build()
5      return Wallet.getPaymentsClient(activity, walletOptions)
6  }
```

Рисунок 3 - настройка API

Чтобы создать запрос, сначала нужно создать объект JSON, включая все типы карт и аутентификации платежей, которые поддерживаются (рис.4).

```
private val baseCardPaymentMethod = JSONObject().apply {
    put("type", "CARD")
    put("parameters", JSONObject().apply {
        put("allowedCardNetworks", JSONArray(listOf("VISA", "MASTERCARD")))
        put("allowedAuthMethods", JSONArray(listOf("PAN_ONLY", "CRYPTOGRAM_3DS")))
    })
}
```

Рисунок 4 – Запрос JSON

При использовании аутентификации по основному номеру счета (PAN) номер карточного счета, а также месяц и год истечения срока действия отправляются в платежную систему. Это позволяет вашим пользователям использовать ранее сохраненные карты. Аутентификация на основе динамической криптограммы работает в сочетании с токенизацией карты, механизмом, который заменяет информацию о карте вашего клиента случайным значением, которое никто, кроме эмитента карты вашего клиента, не может понять. Это в сочетании с динамическими криптограммами гарантирует, что токен, отправляемый процессору, всегда будет отличаться для каждой транзакции, что значительно повышает безопасность этих операций.

Наряду с этим, также нужно указать, к какой версии API Google Pay происходит подключение (рис.5).

```
private val baseCardPaymentMethod = JSONObject().apply {
    put("type", "CARD")
    put("parameters", JSONObject().apply {
        put("allowedCardNetworks", JSONArray(listOf("VISA", "MASTERCARD")))
        put("allowedAuthMethods", JSONArray(listOf("PAN_ONLY", "CRYPTOGRAM_3DS")))
    })
}
```

Рисунок 5 - Версия API

Теперь, когда есть все, что нужно для создания запроса, пришло время создать его и проверить, возможна ли оплата через Google Pay API (рис.6).

```
val readyToPayRequest =
        IsReadyToPayRequest.fromJson(googlePayBaseConfiguration.toString())

val readyToPayTask = paymentsClient.isReadyToPay(readyToPayRequest)
task.addOnCompleteListener { task ->
    try {
        task.getResult(ApiException::class.java)?.let(::setGooglePayAvailable)
    } catch (exception: ApiException) {
        // Error determining readiness to use Google Pay.
        // Inspect the logs for more details.
    }
}

private fun setGooglePayAvailable(available: Boolean) {
    if (available) {
        //enable Google pay button so that user can click on it to pay
    } else {
       // Unable to pay using Google Pay. Update your UI accordingly.
    }
}
```

Рисунок 6 – функции Google pay

Здесь сначала создаем метод «IsReadyToPayRequest», который используем для вызова функции «isreadyToPay», результатом которой является «task» объект. Затем «task» должен добавить объект, «addOnCompleteListener» чтобы проверить, доступен ли Google Pay API или нет. Наконец, когда получаем успешный ответ, можно включить кнопку Google Pay.

Для создания транзакции понадобятся два набора данных, например «transactionInfo» и «merchantInfo».

«transactionInfo» - здесь можно определить такие детали, как сумма, которую клиент должен заплатить и в какой валюте (рис.7).

```
private val transactionInfo = JSONObject().apply {
    put("totalPrice", "123.45")
    put("totalPriceStatus", "FINAL")
    put("currencyCode", "USD")
}
```

Рисунок 7 - метод transactionInfo

«merchantInfo» - здесь можно определить детали продавца, такие как их имя и идентификатор продавца (рис.8).

```
private val merchantInfo = JSONObject().apply {
    put("merchantName", "Example Merchant")
    put("merchantId", "01234567890123456789")
}
```

Рисунок 8 - метод merchantInfo

Теперь пришло время объединить созданную ранее платежную конфигурацию с «transactionInfo» и «merchantInfo» (рис.9).

```
private val paymentDataRequestJson = JSONObject(googlePayBaseConfiguration.toString()).apply {
    put("allowedPaymentMethods", JSONArray().put(cardPaymentMethod))
    put("transactionInfo", transactionInfo)
    put("merchantInfo", merchantInfo)
}
```

Рисунок 9 – объединение методов

Наконец, когда создали «paymentDataRequest» путем пропускания «paymentDataRequestJson», то выполнение этой задачи с помощью «resolveTask» функции, которая имеет три параметра «Task<TResult>», «Activity» и «Int» как «request-id» (рис.10).

```
val paymentDataRequest =
        PaymentDataRequest.fromJson(paymentDataRequestJson.toString())

AutoResolveHelper.resolveTask(
        paymentsClient.loadPaymentData(paymentDataRequest),
        this, LOAD_PAYMENT_DATA_REQUEST_CODE)
```

Рисунок 10 – метод paymentDataRequest

Теперь пользователь продолжит транзакцию Google Pay, и после ее завершения вы получите ответ транзакции в формате «onActivityResult». Так же можно обрабатывать все успешные и неудачные состояния, как показано ниже (рис.11).

```
public override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent) {
    when (requestCode) {
        LOAD_PAYMENT_DATA_REQUEST_CODE -> {
            when (resultCode) {
                Activity.RESULT_OK -> {

                }

                Activity.RESULT_CANCELED -> {

                }

                AutoResolveHelper.RESULT_ERROR -> {

                }
            }
        }
    }
}
```

Рисунок 11 – Ответы транзакции

На это оплата через систему Google pay успешно добавлена, именно такие куски кода нужны в приложении, чтобы происходила оплата при нажатии кнопки.

В данной статье рассматривалась возможность интеграции google pay в android приложение.

Библиографический список

1. Сковорода А.А., Гамаюнов Д.Ю. Анализ мобильных приложений с использованием моделей привилегий и api-вызовов вредоносных приложений// Современные научные исследования и инновации. 2018. № 3-5 (63). С. 42-44.
2. Рубинов К.В. Подход к тестированию программных интерфейсов приложений мобильных устройств // Труды Международного симпозиума «Надежность и качество». 2019. №5. С. 16-22.
3. Орлов В.Г., Шаврин С.С. Беспроводные мобильные приложения в системах мониторинга и диспетчеризации технологических служб // Автоматика. Вычислительная техника. 2019. №1. С. 67-71
4. Глазков С.В., Ронжин А.Л. Методы анализа контекста приложений в мобильных гетерогенных устройствах// Автоматика. Вычислительная техника. 2018. №3. С. 34-39