Pagos y servicios de seguridad

Pablo esteban

Samuel Olmedo Ortiz

Angel Fabrizio Franyutti Pulido

Pagos en móviles

- Los pagos móviles (tanto carteras digitales como transferencias con el móvil) son transacciones reguladas que tienen lugar a través de un dispositivo móvil.

- Los pagos móviles pueden usarse tanto en contextos personales o P2P como en negocios físicos.

FORMAS DE PAGO POR EL TIEMPO

Forma de pago por SMS: El costo de la compra se agregaba a la factura del servicio móvil del usuario.

1990

1997

WAP, Pago a través de navegación en Internet móvil. Los usuarios podían realizar compras en línea utilizando sus teléfonos móviles.

NFC:Pago a través de tecnología de comunicación de campo cercano. Simplemente acercando su teléfono

2000/2

2010

Aplicaciones de pago móvil:Los usuarios pueden vincular sus tarjetas de crédito o débito, y utilizarla para realizar compras en línea o en tiendas

Pagos por medio de códigos QR una forma de tecnología de barrido de datos que permite a los usuarios realizar pagos escaneando

2014

2016/7

Pagos biométricos.Son formas de pago donde se interactuá directamente con el usuario de forma fisica como los pueden ser por las huellas o el rostro para hacer y autorizar pagos

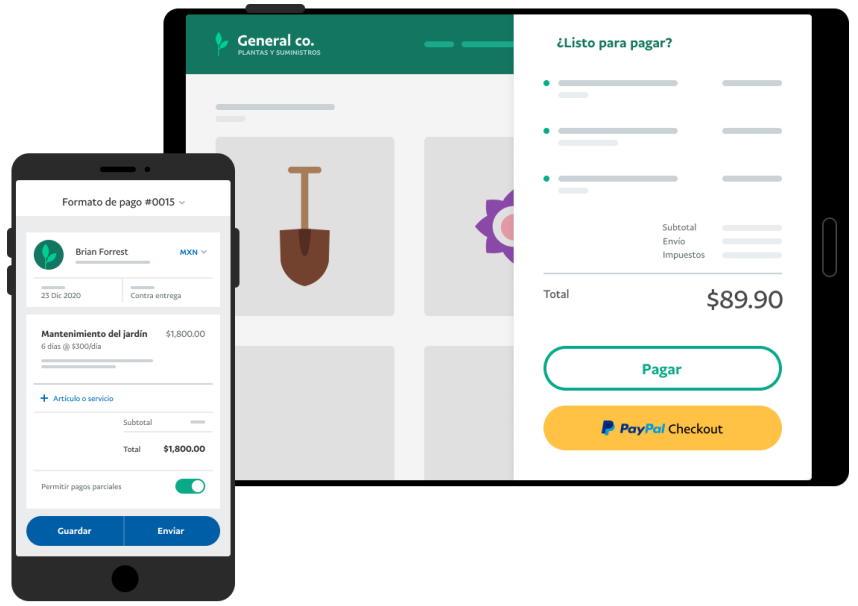
Proveedores de pago móvil

El lanzamiento de Apple Pay en 2014 fue un hito importante en la evolución de los pagos móviles, ya que la solución de Apple permitía a los usuarios almacenar sus tarjetas de crédito y débito en su dispositivo y realizar pagos sin necesidad de llevar una cartera física.

Apple Pay, Google Wallet y Samsung Pay.

PayPal

Un servicio que te permite pagar, enviar dinero y aceptar pagos sin tener que introducir tus datos financieros continuamente.



Google pay

Google Pay te permite poder realizar pagos a través de tu móvil, smartwatch o **cualquier otro dispositivo Android con NFC**. Además, también es la tecnología con la que puedes hacer algunos pagos online, como por ejemplo en la tienda oficial de Google. Aun así, su función principal sigue siendo la de realizarlos a través de dispositivos.

Google Pay

Proceso de implementar pagos en Google play

Groovy Kotlin

```
dependencies {
    def billing_version = "5.1.0"

    implementation "com.android.billingclient:billing:$billing_version"
}
```

Paso 1:Implementación de la biblioteca de pagos de Google play

Kotlin Java

```
private PurchasesUpdatedListener purchasesUpdatedListener = new PurchasesUpdatedListener() {
    @Override
    public void onPurchasesUpdated(BillingResult billingResult, List<Purchase> purchases) {
        // To be implemented in a later section.
    }
};

private BillingClient billingClient = BillingClient.newBuilder(context)
    .setListener(purchasesUpdatedListener)
    .enablePendingPurchases()
    .build();
```

Paso 2:Llamada a la función de pagos

Kotlin Java

```java
billingClient.startConnection(new BillingClientStateListener() {
    @Override
    public void onBillingSetupFinished(BillingResult billingResult) {
        if (billingResult.getResponseCode() ==  BillingResponseCode.OK) {
            // The BillingClient is ready. You can query purchases here.
        }
    }
    @Override
    public void onBillingServiceDisconnected() {
        // Try to restart the connection on the next request to
        // Google Play by calling the startConnection() method.
    }
});
```

Paso 3:Inicia la conexión de Google play para proceder el pago

Paso 4:Hace un listado de productos

Kotlin Java

```java
QueryProductDetailsParams queryProductDetailsParams =
    QueryProductDetailsParams.newBuilder()
        .setProductList(
            ImmutableList.of(
                Product.newBuilder()
                    .setProductId("product_id_example")
                    .setProductType(ProductType.SUBS)
                    .build()))
        .build();

billingClient.queryProductDetailsAsync(
    queryProductDetailsParams,
    new ProductDetailsResponseListener() {
        public void onProductDetailsResponse(BillingResult billingResult,
                List<ProductDetails> productDetailsList) {
            // check billingResult
            // process returned productDetailsList
        }
    }
)
```

Kotlin Java

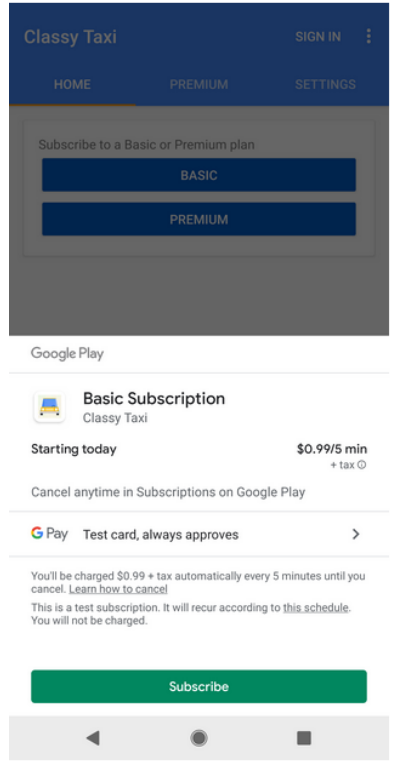
```java
// An activity reference from which the billing flow will be launched.
Activity activity = ...;

ImmutableList productDetailsParamsList =
    ImmutableList.of(
        ProductDetailsParams.newBuilder()
             // retrieve a value for "productDetails" by calling queryProductDetailsAsync()
            .setProductDetails(productDetails)
            // to get an offer token, call ProductDetails.getSubscriptionOfferDetails()
            // for a list of offers that are available to the user
            .setOfferToken(selectedOfferToken)
            .build()
    );

BillingFlowParams billingFlowParams = BillingFlowParams.newBuilder()
    .setProductDetailsParamsList(productDetailsParamsList)
    .build();

// Launch the billing flow
BillingResult billingResult = billingClient.launchBillingFlow(activity, billingFlowParams);
```

Paso 5:Aquí empieza el flujo de la compra



Paso 6:Cuando se actualiza el estado de la compra

Kotlin Java

```java
@Override
void onPurchasesUpdated(BillingResult billingResult, List<Purchase> purchases) {
    if (billingResult.getResponseCode() == BillingResponseCode.OK
        && purchases != null) {
        for (Purchase purchase : purchases) {
            handlePurchase(purchase);
        }
    } else if (billingResult.getResponseCode() == BillingResponseCode.USER_CANCELED) {
        // Handle an error caused by a user cancelling the purchase flow.
    } else {
        // Handle any other error codes.
    }
}
```

Kotlin Java

```java
void handlePurchase(Purchase purchase) {
    // Purchase retrieved from BillingClient#queryPurchasesAsync or your PurchasesUpdatedListener.
    Purchase purchase = ...;

    // Verify the purchase.
    // Ensure entitlement was not already granted for this purchaseToken.
    // Grant entitlement to the user.

    ConsumeParams consumeParams =
        ConsumeParams.newBuilder()
            .setPurchaseToken(purchase.getPurchaseToken())
            .build();

    ConsumeResponseListener listener = new ConsumeResponseListener() {
        @Override
        public void onConsumeResponse(BillingResult billingResult, String purchaseToken) {
            if (billingResult.getResponseCode() == BillingResponseCode.OK) {
                // Handle the success of the consume operation.
            }
        }
    };

    billingClient.consumeAsync(consumeParams, listener);
}
```

Paso 7:Aquí se actualiza el estado del producto que fue comprado en la app

La seguridad en móviles

- Se refiere a las estrategias, infraestructuras y software que se utilizan para proteger cualquier dispositivo móvil que viaja junto con los usuarios.

Historia de seguridad movil

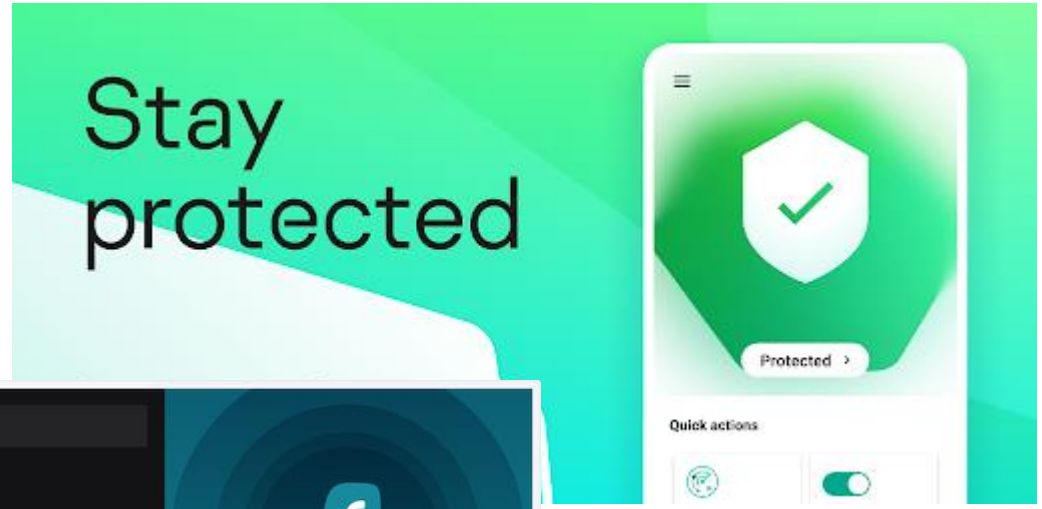
- Primeros dispositivos móviles (década de 1990 a principios de la década de 2000): la principal preocupación era protegerlos contra robos o pérdidas.

- Teléfonos inteligentes y los primeros sistemas operativos móviles (finales de la década de 2000):la seguridad se convirtió en una preocupación mayor, y el malware y otras amenazas se volvieron más comunes.
- Soluciones de seguridad móvil (década de 2010):Estos incluyen aplicaciones antivirus, firewalls y herramientas de administración de dispositivos móviles (MDM) .

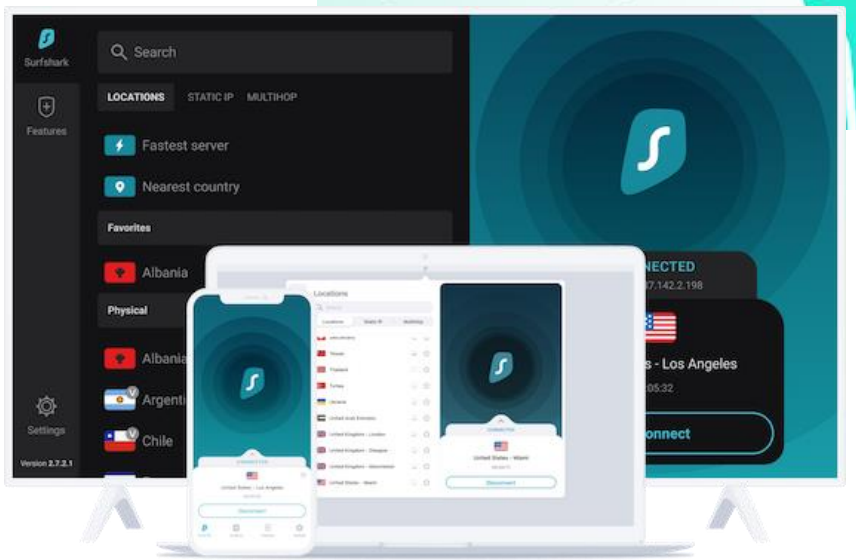
- Cifrado y biometría (finales de la década de 2010):El cifrado se volvió más utilizado, lo que dificultó que los atacantes interceptaran datos confidenciales. La autenticación biométrica, como la huella dactilar y el reconocimiento facial
- Desafíos actuales de seguridad móvil (década de 2020):Las amenazas incluyen malware, ataques de phishing y violaciones de datos.

Empresas de seguridad movil

- Kaspersky es una de las empresas de ciberseguridad de propiedad privada más grandes del mundo



- Surfshark es una empresa de ciberseguridad que se enfoca en desarrollar soluciones humanizadas de privacidad y seguridad.



Aplica permisos basados en firmas

Cuando compartes datos entre dos apps que controlas o posees, usa permisos *basados en firmas*. Estos permisos no requieren la confirmación del usuario y, en cambio, comprueban que la app que accede a los datos esté firmada con la misma clave. Por lo tanto, estos permisos ofrecen una experiencia del usuario más segura y simplificada.

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.myapp">
    <permission android:name="my_custom_permission_name"
                android:protectionLevel="signature" />
```

Inhabilita el acceso a los proveedores de contenido de tu app

A menos que pretendas enviar datos desde tu app a una diferente que no te pertenece, impide explícitamente que las apps de otros desarrolladores accedan a los objetos `ContentProvider` de tu app. Esta opción de configuración resulta particularmente importante si se puede instalar tu app en dispositivos que ejecutan Android 4.1.1 (nivel de API 16) o versiones anteriores, ya que el atributo `android:exported` del elemento `<provider>` es `true` de forma predeterminada en esas versiones de Android.

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.myapp">
    <application ... >
        <provider
            android:name="android.support.v4.content.FileProvider"
            android:authorities="com.example.myapp.fileprovider"
            ...
            android:exported="false">
            <!-- Place child elements of <provider> here. -->
        </provider>
        ...
    </application>
</manifest>
```

Usa canales de mensajes HTML

Si tu app usa compatibilidad con la interfaz de JavaScript en dispositivos que ejecutan Android 6.0 (API nivel 23) y versiones posteriores, usa los canales de mensajes HTML en lugar de establecer una comunicación entre un sitio web y tu app, como se muestra en el siguiente fragmento de código:

Kotlin Java

```kotlin
val myWebView: WebView = findViewById(R.id.webview)

// channel[0] and channel[1] represent the two ports.
// They are already entangled with each other and have been started.
val channel: Array<out WebMessagePort> = myWebView.createWebMessageChannel()

// Create handler for channel[0] to receive messages.
channel[0].setWebMessageCallback(object : WebMessagePort.WebMessageCallback() {

    override fun onMessage(port: WebMessagePort, message: WebMessage) {
        Log.d(TAG, "On port $port, received this message: $message")
    }
})

// Send a message from channel[1] to channel[0].
channel[1].postMessage(WebMessage("My secure message"))
```

Proporciona los permisos adecuados

Solo solicita la cantidad mínima de permisos necesarios para que tu app funcione adecuadamente. Cuando sea posible, renuncia a los permisos cuando tu app ya no los necesite.

Usa intents para diferir permisos

Siempre que sea posible, no agregues un permiso a tu app para completar una acción que puede completarse en otra app. En cambio, usa un intent a fin de diferir la solicitud a una app diferente que ya tiene el permiso necesario.

En el siguiente ejemplo, se muestra cómo usar un intent para dirigir a los usuarios a una app de contactos, en lugar de solicitar los permisos `READ_CONTACTS` y `WRITE_CONTACTS`:

Kotlin Java

```kotlin
// Delegates the responsibility of creating the contact to a contacts app,
// which has already been granted the appropriate WRITE_CONTACTS permission.
Intent(Intent.ACTION_INSERT).apply {
    type = ContactsContract.Contacts.CONTENT_TYPE
}.also { intent ->
    // Make sure that the user has a contacts app installed on their device.
    intent.resolveActivity(packageManager)?.run {
        startActivity(intent)
    }
}
```

Almacena datos de manera segura

Si bien tu app podría requerir acceso a información sensible del usuario, tus usuarios solo le otorgarán a tu app acceso a sus datos si confían en que los protegerás adecuadamente.

Guarda datos privados en el almacenamiento interno

Almacena todos los datos del usuario privados dentro del almacenamiento interno del dispositivo, que aísla la información de cada app. No es necesario que tu app solicite permiso para ver esos archivos, y otras apps no podrán acceder a ellos. Como medida de seguridad adicional, cuando el usuario desinstala una app, el dispositivo borra todos los archivos que la app guardó en el almacenamiento interno.

```
// Although you can define your own key generation parameter specification, it's
// recommended that you use the value specified here.
val keyGenParameterSpec = MasterKeys.AES256_GCM_SPEC
val mainKeyAlias = MasterKeys.getOrCreate(keyGenParameterSpec)

// Create a file with this name or replace an entire existing file
// that has the same name. Note that you cannot append to an existing file,
// and the filename cannot contain path separators.
val fileToWrite = "my_sensitive_data.txt"
val encryptedFile = EncryptedFile.Builder(
    File(DIRECTORY, fileToWrite),
    applicationContext,
    mainKeyAlias,
    EncryptedFile.FileEncryptionScheme.AES256_GCM_HKDF_4KB
).build()

val fileContent = "MY SUPER-SECRET INFORMATION"
        .toByteArray(StandardCharsets.UTF_8)
encryptedFile.openFileOutput().apply {
    write(fileContent)
    flush()
    close()
}
```

Almacena solo datos no sensibles en archivos de caché

Para proporcionar un acceso más rápido a los datos no sensibles de la app, almacénalos en la memoria caché del dispositivo. Para cachés de más de 1 MB, usa `getExternalCacheDir()`. Para cachés de 1 MB o menos, usa `getCacheDir()`. Ambos métodos te proporcionan el objeto `File` que contiene los datos en caché de tu app.

En el siguiente fragmento de código, se muestra cómo almacenar en caché un archivo que tu app descargó recientemente:

Kotlin Java

```
val cacheFile = File(myDownloadedFileUri).let { fileToCache ->
    File(cacheDir.path, fileToCache.name)
}
```

Referencias:

- https://www.ibm.com/mx-es/topics/mobile-security
- https://www.proofpoint.com/es/threat-reference/mobile-security
- https://www.cisco.com/c/es_mx/solutions/small-business/resource-center/security/mobile-device-security.html
- https://www.paypal.com/es/webapps/mpp/paypal-popup
- https://developer.android.com/google/play/billing/integrate?hl=es-419#groovy
- https://pay.google.com/intl/es_es/about/