

Compras In-App

Monetización de Apps

- Gratis con anuncios.
- App de pago.
- **Micro-pagos (compras in-app).**

¿Qué son las compras In-App?

- Permiten ofrecer funcionalidades extra o un valor añadido al usuario a cambio de un micro-pago desde la app.
- Tipos:
 - Consumibles: pistas o vidas extra en un videojuego.
 - No consumibles: eliminar anuncios.
 - Suscripciones renovables: servicios como Plex.
 - Suscripciones no renovables: pase de temporada de un deporte.

Pasos a seguir: Configuración Inicial

- Crear IAPs en iTunes Connect 🤖
- Activar IAPs para el App ID y en Xcode.

Pasos a seguir: Programación

- Refrescar listado de productos.
- Mostrar (**OBLIGATORIO**):
 - Cada IAP acompañada de su precio localizado.
 - Opción para Restaurar Compras.

Pasos a seguir: Testing

- Crear Sandbox User en iTunes Connect. ¡IAPs gratis!
- Logout en iOS > Preferencias > iTunes Store.
- Probar IAPs en dispositivo iOS (no vale simulador).

Pasos a seguir: Configuración Final

- Enviar la IAP para revisión **antes** de enviar el binario de la app.
- Una vez aprobada la IAP, debes marcar la casilla “Aprobada para la venta”
- Después de este paso puede tardar unas horas en estar disponible.

Configuración Inicial


Crear IAPs

iTunes Connect

[App Store](#)[Prestaciones](#)[TestFlight](#)[Actividad](#)[Análisis de las apps](#)[Ventas y tendencias](#)


[Compras dentro de la app](#)[Game Center](#)[Encriptación](#)[Códigos promocionales](#)

Compras dentro de la app



Tu primera compra dentro de la app debe enviarse con una versión de app nueva. Selecciónala en la sección de compras dentro de la app y haz clic en Enviar.

Cuando el binario se haya cargado y hayas enviado la primera compra dentro de la app para su revisión, podrás enviar otras compras dentro de la app usando la siguiente tabla.

Compras dentro de la app (0) 

[Ver secreto compartido](#)

Haz clic en + para añadir una compra dentro de la app

Crear IAPs

Selecciona la compra dentro de la app que quieres crear.

- ☐ **Consumible**
Un producto que se agota después de usarlo una vez y que hay que volver a comprar.
Ejemplo: Cebo para una app de pesca.

- ☒ **No consumible**
Un producto que se adquiere una vez y que no vence o se gasta con el uso.
Ejemplo: Una pista de carreras para una app de juego.

- ☐ **Suscripción con renovación automática**
Un producto que permite a los usuarios comprar contenido dinámico, durante un tiempo determinado. Este tipo de suscripción se renueva automáticamente a menos que el usuario la cancele.
Ejemplo: Suscripción mensual para una app que ofrece servicios de streaming.

- ☐ **Suscripción sin renovación automática**
Un producto que permite a los usuarios adquirir un servicio con duración limitada. El contenido de esta compra dentro de la app puede ser estático. Este tipo de suscripción no se renueva automáticamente.
Ejemplo: Suscripción anual a un catálogo de artículos de archivo.

[Más información sobre las Compras dentro de la app.](#)

Cancelar

Crear

Crear IAPs

Compras dentro de la app > Nueva compra dentro de la app

Guardar

Nombre de referencia ?

Quitar Anuncios

ID del producto ?

es.upsa.mimo.myapp.removeads

Disponibilidad ?

☐ Aprobada para la venta

Precio

[Todos los precios y monedas](#)

Precio ?

Fecha de inicio ?

Fecha final ?

EUR 1.99 (Franj... | v

[Otras monedas](#)

26 nov. 2016

Sin fecha final

Crear IAPs

Idiomas ⊕

Inglés (EE. UU.)

Nombre en pantalla ?

Remove Ads

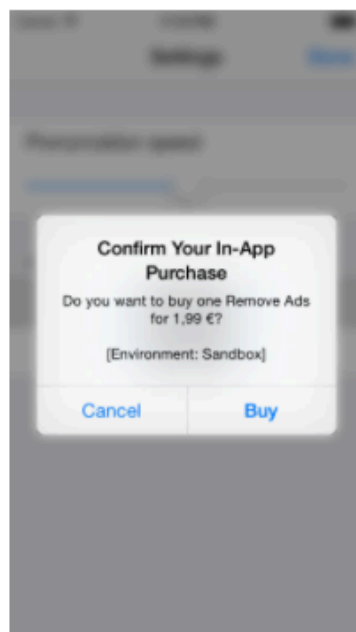
Descripción ?

Remove Ads

245

Información de revisión

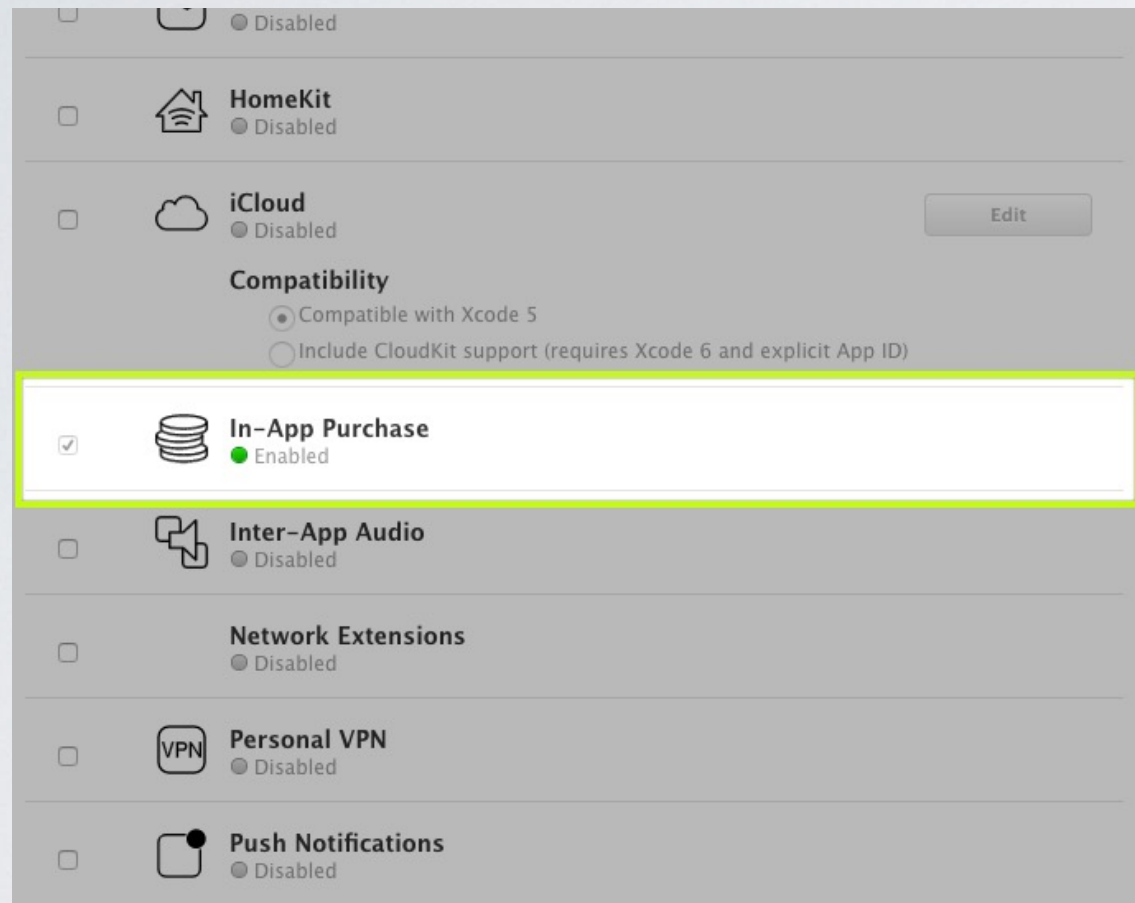
Captura de pantalla ?



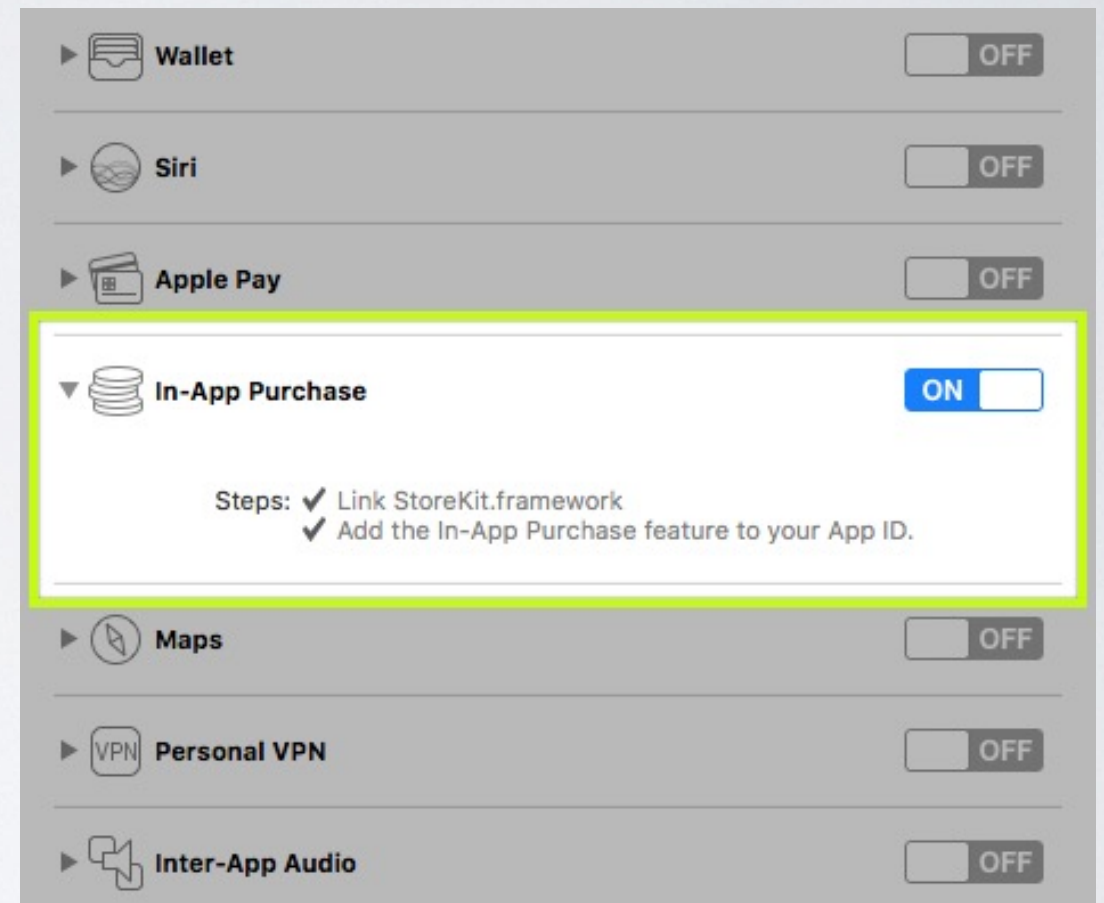
Notas de la revisión ?

4000

Habilitar IAPs



Apple Member Center
App ID



Xcode
Target Capabilities

Programación

StoreKit

- Framework de Apple para manejo de IAPs.
- API compleja para la mayoría de los casos.
- **SKProduct**: producto ofrecido para IAP.
- **SKPayment**: encapsula una petición de pago.
- **SKPaymentTransaction**: representa una compra.

RMStore

- Software Libre.
- Basado en CargoBay de @mattt.
- API sencilla.
- Apple TV: usar rama **tvos-support** (PR 183)

Refrescar listado de productos

```
let store = RMStore()
```

```
store.request(products: ["es.upsa.mimo.inapp"]
```

```
    success: (products, invalidProductIDs) in {  
        // Éxito
```

```
    }
```

```
    failure: error in {
```

```
        // Error
```

```
    })
```


Mostrar listado de productos

```
let myProductIDs = ["es.upsa.mimo.inapp"]
```

```
let myProducts = []
```

```
for (ID in myProductIDs)
```

```
{
```

```
    myProducts += store.productForIdentifier(ID)
```

```
}
```

Mostrar listado de productos (funcional)

```
let myProductIDs = ["es.upsa.mimo.inapp"]  
let myProducts = myProductIDs.map(_ in {  
    store.productForIdentifier($0)  
})
```

Comprar producto

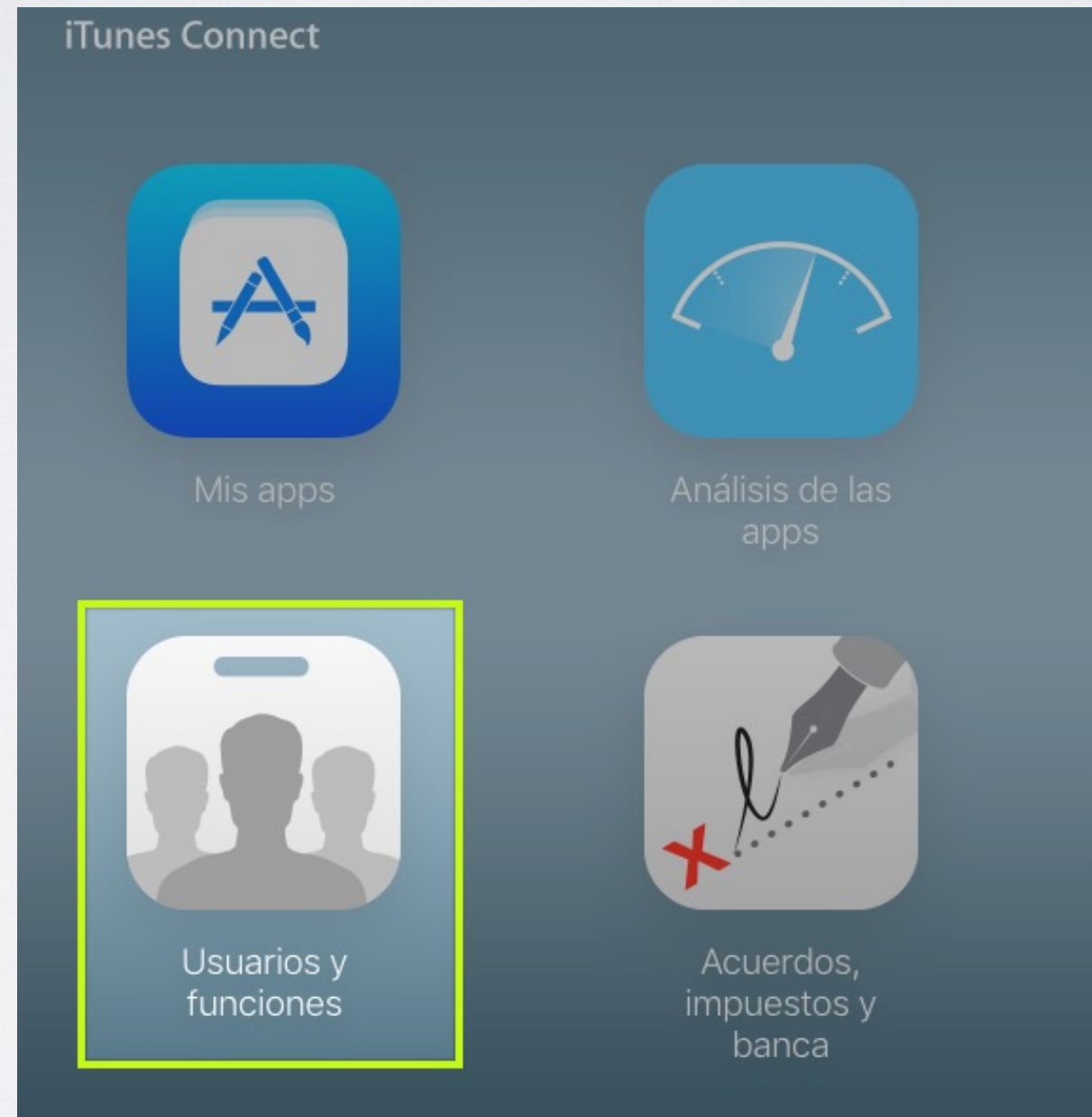
```
store.addPayment("es.upsa.mimo.inapp"  
    success: transaction in {  
        // Éxito  
    }  
    failure: (transaction, error) in {  
        // Error  
    })
```


Restaurar compras

```
store.restoreTransactions(onSuccess: transactions in {  
    // Éxito  
}  
failure: error in {  
    // Error  
})
```

Testing

Crear Sandbox User



Crear Sandbox User



Crear Sandbox User

[← Probadores de Sandbox](#)

Añadir probador de Sandbox

Cancelar

Guardar

Los probadores de Sandbox pueden probar tus apps en modo de desarrollo y hacer Compras dentro de la app sin realizar transacciones reales.

Los usuarios añadidos después del 13 de junio de 2016, también pueden realizar pruebas de transacciones utilizando Apple Pay.

Información del probador

Nombre

Usuario

Apellidos

De Prueba

Correo electrónico

usuario.deprueba+iap1@gmail.com

Contraseña

.....

Confirmar contraseña

.....

Pregunta secreta

Pregunta secreta?

Respuesta secreta

Respuesta secreta

Fecha de nacimiento

10



Febrero

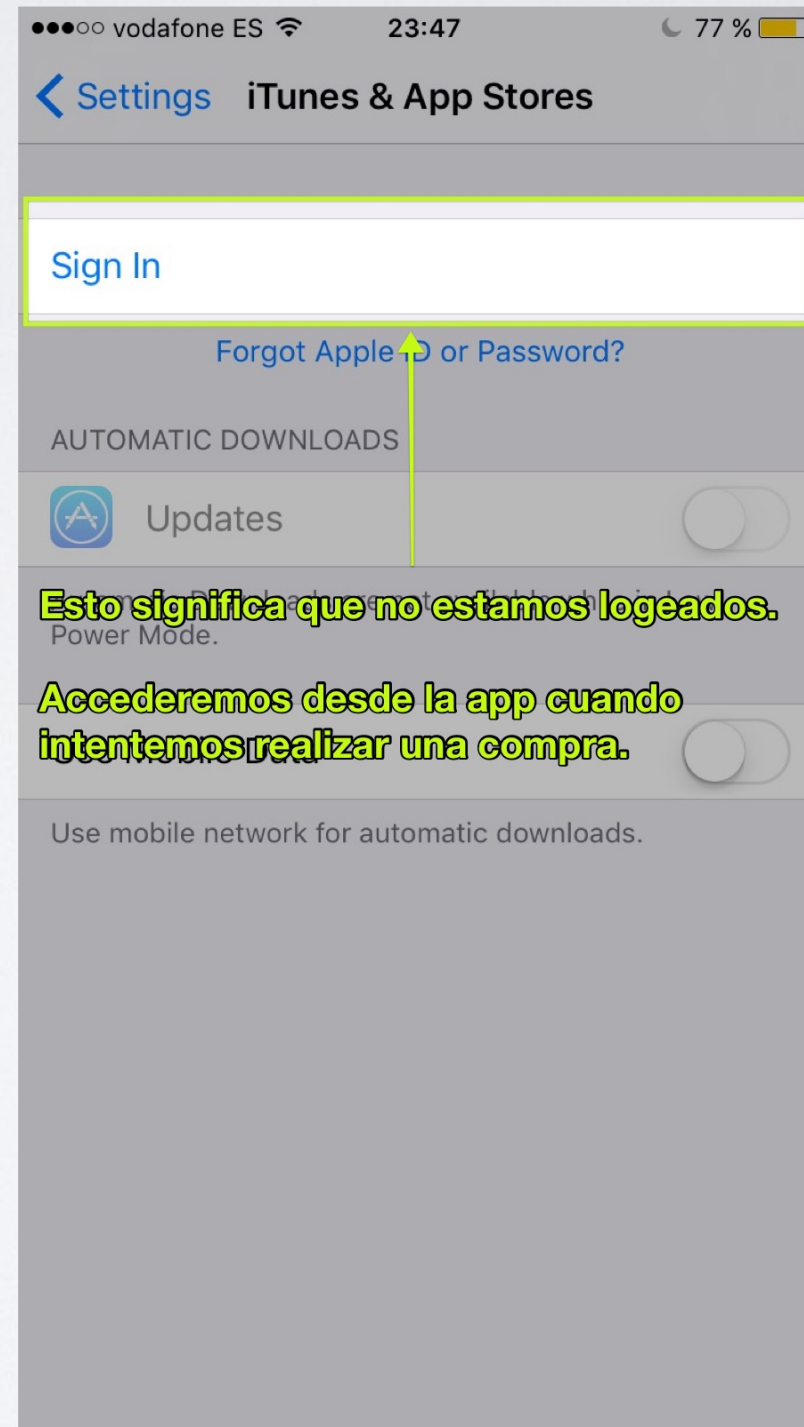
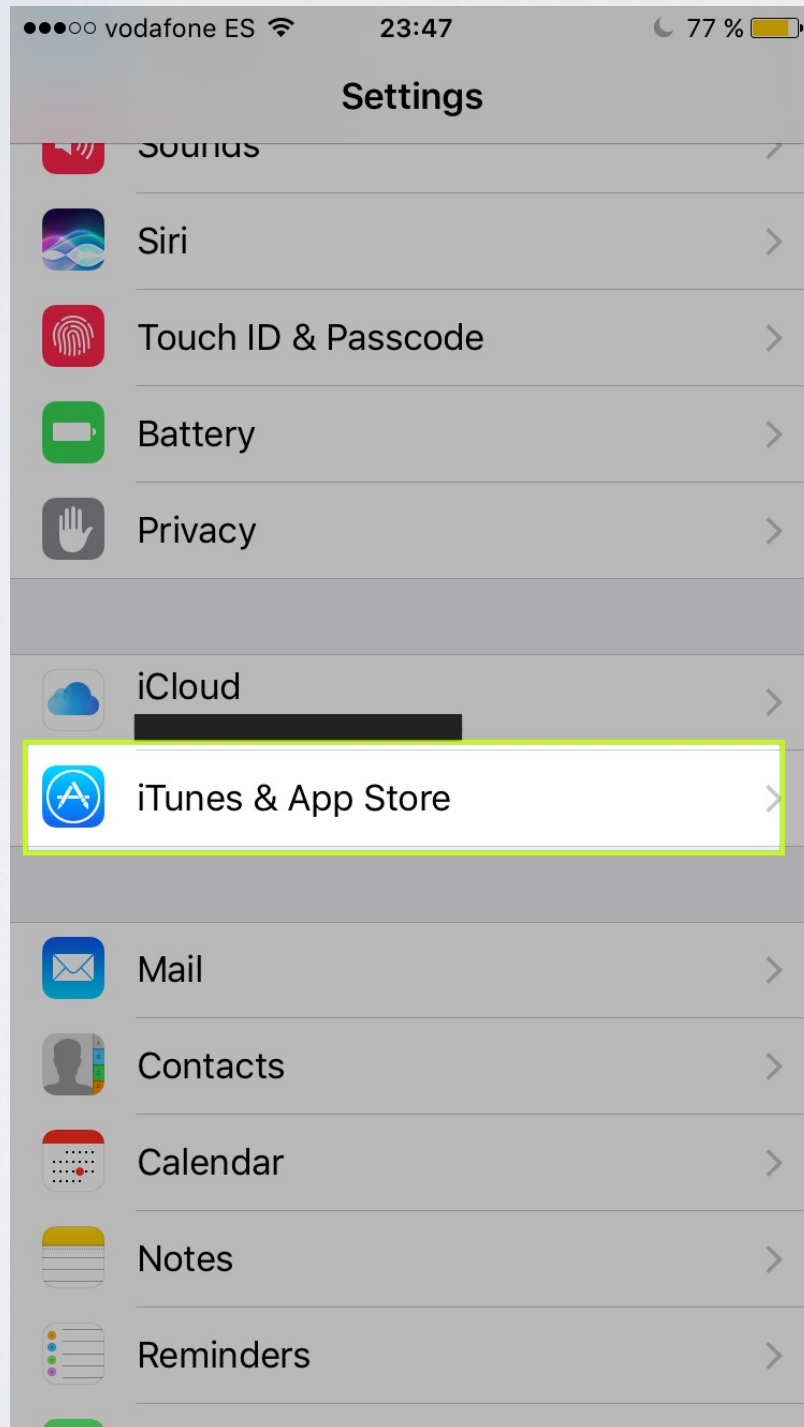


Territorio del App Store

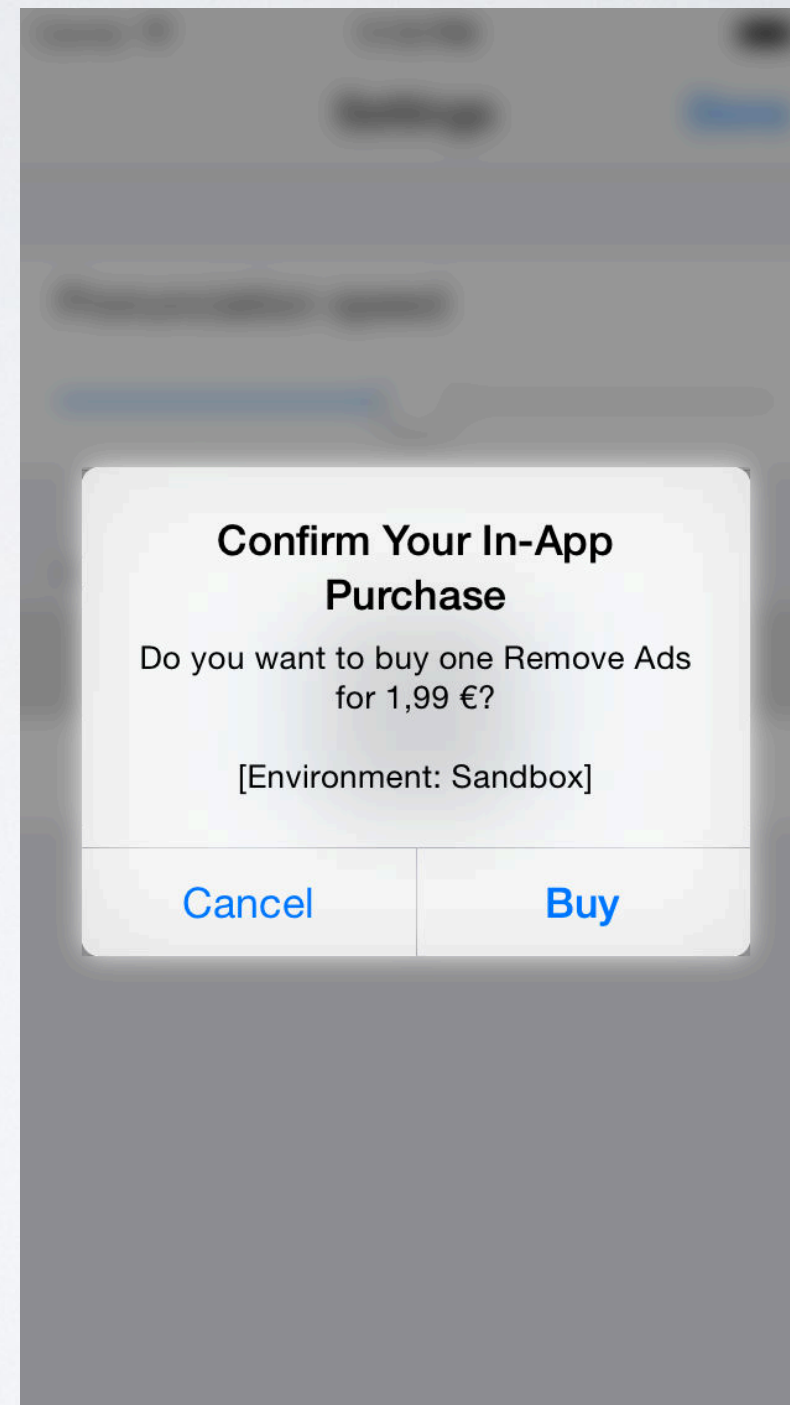
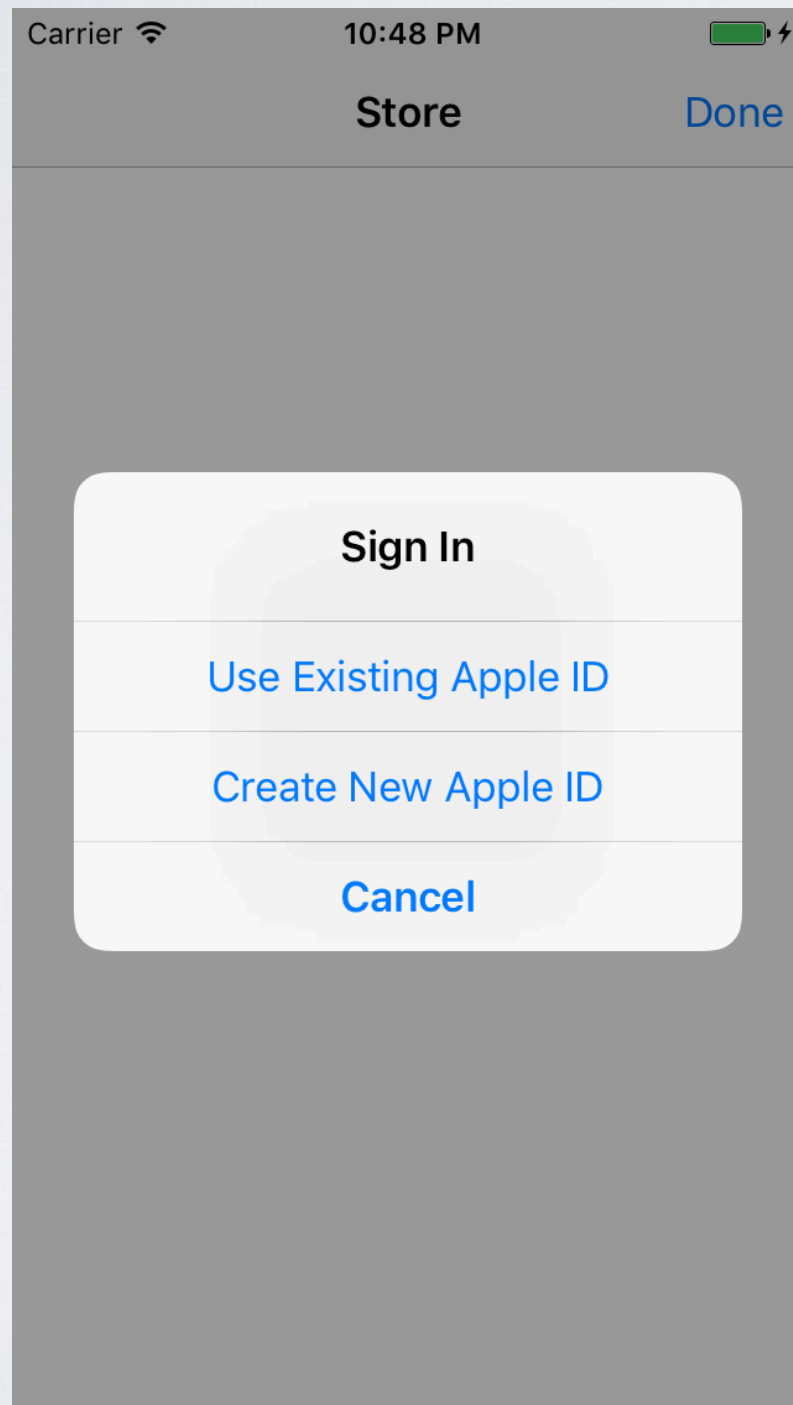
España



Logout en iOS



Probar IAPs



Configuración final

Publicar IAP

Enviar para su revisión

Nombre de referencia ?

Quitar Anuncios

Disponibilidad ?

☒ Aprobada para la venta

PROFIT



Funcionalidades adicionales:

Persistir compras

- Keychain
 - RMStoreKeychainPersistence
- UserDefaults
 - RMStoreUserDefaultsPersistence
- Personalizado
 - Clase que conforme RMStoreTransactionPersistor

Funcionalidades adicionales:

Persistir compras

```
let store = RMStore()  
store.transactionPersistor = RMStoreKeychainPersistence()
```


Funcionalidades adicionales:

Verificar compras

- Con *jailbreak* es fácil piratear IAPs.
- Verificar en un servidor propio que la transacción no es fraudulenta.
- Aconsejable **al menos** para IAPs que supongan un coste a la empresa.
- Clase que conforme **RMStoreReceiptVerifier**.

Funcionalidades adicionales:

Verificar compras

```
class MyCustomIAPVerifier
{
    func verify(transaction: SKPaymentTransaction,
               success: (() -> ()), failure: (() -> ()))
    {
        // Petición a mi servidor
        // Si hay error, invocar failure. Sino, invocar success.
    }
}
```

Funcionalidades adicionales:

Verificar compras

```
let store = RMStore()
```

```
store.receiptVerifier = MyCustomIAPVerifier()
```