# Holocron SWU Card Scanner - Cliente Móvil

Este es el cliente móvil para el escáner de cartas de Star Wars Unlimited, construido con **React Native** y TypeScript.

## Características

- 📱 Aplicación móvil multiplataforma (iOS y Android)
- 📷 Escáner de cartas usando cámara con IA
- 🔍 Reconocimiento de cartas Star Wars Unlimited
- 💾 Almacenamiento local de colecciones
- 🌐 Integración con backend para sincronización
- 🎨 Interfaz moderna y responsiva

# Primeros Pasos

> **Nota**: Asegúrate de haber completado la Configuración del Entorno antes de continuar.

## Paso 1: Iniciar Metro

Primero, necesitarás ejecutar **Metro**, la herramienta de construcción de JavaScript para React Native.

Para iniciar el servidor de desarrollo de Metro, ejecuta el siguiente comando desde la raíz de tu proyecto React Native:

```
# Usando npm
npm start

# O usando Yarn
yarn start
```

## Paso 2: Construir y ejecutar tu aplicación

Con Metro ejecutándose, abre una nueva ventana/panel de terminal desde la raíz de tu proyecto React Native, y usa uno de los siguientes comandos para construir y ejecutar tu aplicación Android o iOS:

### Android

```
# Usando npm
npm run android

# O usando Yarn
yarn android
```

### iOS

Para iOS, recuerda instalar las dependencias de CocoaPods (esto solo necesita ejecutarse en el primer clon o después de actualizar dependencias nativas).

La primera vez que crees un nuevo proyecto, ejecuta el Ruby bundler para instalar CocoaPods:

```
bundle install
```

Luego, y cada vez que actualices tus dependencias nativas, ejecuta:

```
bundle exec pod install
```

Para más información, visita la Guía de Inicio de CocoaPods.

```
# Usando npm
npm run ios

# O usando Yarn
yarn ios
```

Si todo está configurado correctamente, deberías ver tu nueva aplicación ejecutándose en el Emulador de Android, Simulador de iOS, o tu dispositivo conectado.

Esta es una forma de ejecutar tu aplicación — también puedes construirla directamente desde Android Studio o Xcode.

## Paso 3: Modificar tu aplicación

¡Ahora que has ejecutado exitosamente la aplicación, hagamos algunos cambios!

Abre `App.tsx` en tu editor de texto preferido y haz algunos cambios. Cuando guardes, tu aplicación se actualizará automáticamente y reflejará estos cambios — esto está impulsado por Fast Refresh.

Cuando quieras recargar forzosamente, por ejemplo para resetear el estado de tu aplicación, puedes realizar una recarga completa:

- **Android**: Presiona la tecla `R` dos veces o selecciona **"Reload"** desde el **Dev Menu**, accesible via `Ctrl` + `M` (Windows/Linux) o `Cmd ⌘` + `M` (macOS).
- **iOS**: Presiona `R` en el Simulador de iOS.

## ¡Felicitaciones! :tada:

Has ejecutado y modificado exitosamente tu aplicación React Native. :partying_face:

### ¿Qué sigue?

- Si quieres agregar este nuevo código React Native a una aplicación existente, revisa la Guía de Integración.
- Si tienes curiosidad de aprender más sobre React Native, revisa la documentación.

# Solución de Problemas

Si tienes problemas para que los pasos anteriores funcionen, consulta la página de Solución de Problemas.

# Aprende Más

Para aprender más sobre React Native, echa un vistazo a los siguientes recursos:

- Sitio Web de React Native - aprende más sobre React Native.
- Primeros Pasos - una **visión general** de React Native y cómo configurar tu entorno.
- Aprende lo Básico - un **tour guiado** de los **conceptos básicos** de React Native.
- Blog - lee las últimas publicaciones oficiales del **Blog** de React Native.
- `@facebook/react-native` - el **repositorio** de GitHub de código abierto para React Native.

## Dependencias del Proyecto

### Principales

- **react-native-vision-camera**: Para captura y procesamiento de imágenes de cartas
- **@react-navigation/native**: Sistema de navegación entre pantallas
- **@react-native-async-storage/async-storage**: Almacenamiento local persistente
- **react-native-vector-icons**: Iconografía de la aplicación

### Desarrollo

- **TypeScript**: Tipado estático para mejor desarrollo
- **ESLint**: Linting de código
- **Prettier**: Formateo automático de código
- **Jest**: Framework de testing

## Estructura del Proyecto

```
src/
├── components/         # Componentes reutilizables
├── screens/           # Pantallas de la aplicación
├── navigation/        # Configuración de navegación
├── services/          # Servicios para API y ML
├── types/             # Definiciones de TypeScript
├── utils/             # Utilidades y helpers
└── assets/            # Imágenes, iconos, etc.
```

## Scripts Disponibles

- `npm start` : Inicia Metro bundler
- `npm run android` : Ejecuta la app en Android
- `npm run ios` : Ejecuta la app en iOS
- `npm run lint` : Ejecuta el linter
- `npm test` : Ejecuta las pruebas

# Getting Started

> **Note**: Make sure you have completed the Set Up Your Environment guide before proceeding.

## Step 1: Start Metro

First, you will need to run **Metro**, the JavaScript build tool for React Native.

To start the Metro dev server, run the following command from the root of your React Native project:

```
# Using npm
npm start

# OR using Yarn
yarn start
```

## Step 2: Build and run your app

With Metro running, open a new terminal window/pane from the root of your React Native project, and use one of the following commands to build and run your Android or iOS app:

### Android

```
# Using npm
npm run android

# OR using Yarn
yarn android
```

### iOS

For iOS, remember to install CocoaPods dependencies (this only needs to be run on first clone or after updating native deps).

The first time you create a new project, run the Ruby bundler to install CocoaPods itself:

```
bundle install
```

Then, and every time you update your native dependencies, run:

```
bundle exec pod install
```

For more information, please visit CocoaPods Getting Started guide.

```
# Using npm
npm run ios
```

```
# OR using Yarn
yarn ios
```

If everything is set up correctly, you should see your new app running in the Android Emulator, iOS Simulator, or your connected device.

This is one way to run your app — you can also build it directly from Android Studio or Xcode.

## Step 3: Modify your app

Now that you have successfully run the app, let's make changes!

Open `App.tsx` in your text editor of choice and make some changes. When you save, your app will automatically update and reflect these changes — this is powered by Fast Refresh.

When you want to forcefully reload, for example to reset the state of your app, you can perform a full reload:

- **Android**: Press the `R` key twice or select **"Reload"** from the **Dev Menu**, accessed via `Ctrl` + `M` (Windows/Linux) or `Cmd ⌘` + `M` (macOS).
- **iOS**: Press `R` in iOS Simulator.

## Congratulations! :tada:

You've successfully run and modified your React Native App. :partying_face:

### Now what?

- If you want to add this new React Native code to an existing application, check out the Integration guide.
- If you're curious to learn more about React Native, check out the docs.

# Troubleshooting

If you're having issues getting the above steps to work, see the Troubleshooting page.

# Learn More

To learn more about React Native, take a look at the following resources:

- React Native Website - learn more about React Native.

- Getting Started - an **overview** of React Native and how setup your environment.
- Learn the Basics - a **guided tour** of the React Native **basics**.
- Blog - read the latest official React Native **Blog** posts.
- `@facebook/react-native` - the Open Source; GitHub **repository** for React Native.