

Construyendo Aplicaciones Móviles Multiplataforma con React Native: Una Guía Completa

Introducción

En la era digital actual, el desarrollo de aplicaciones móviles se ha convertido en un componente esencial para empresas y emprendedores que buscan expandir su alcance y compromiso con los usuarios. Sin embargo, el desarrollo de aplicaciones móviles presenta desafíos únicos, como la necesidad de crear versiones para múltiples plataformas sin comprometer la calidad o la experiencia del usuario.

¿Qué es React Native?

React Native es un marco de desarrollo de aplicaciones móviles multiplataforma creado por Facebook en 2015. Utiliza JavaScript como lenguaje de programación y se basa en React, un popular marco de desarrollo de interfaces de usuario para aplicaciones web.

La principal premisa detrás de React Native es permitir a los desarrolladores crear aplicaciones móviles nativas para iOS y Android utilizando un código base común. Esto significa que un equipo de desarrollo puede escribir una sola base de código en JavaScript y luego compilarla para ejecutarse como una aplicación nativa en múltiples plataformas. Esto ofrece numerosas ventajas, como una mayor eficiencia en el desarrollo, un mantenimiento simplificado y la capacidad de llegar a una audiencia más amplia sin tener que reescribir completamente la aplicación para cada plataforma.

Una de las características más destacadas de React Native es su enfoque en la creación de interfaces de usuario declarativas mediante el uso de componentes reutilizables. Estos componentes, escritos en JavaScript, pueden representar elementos de la interfaz de usuario como botones, campos de entrada y listas, y se pueden combinar para construir interfaces complejas.

Iniciando el Desarrollo en React Native

Cuando nos embarcamos en el desarrollo de una aplicación en React Native, es fundamental conocer dónde podemos acceder a documentación, ejemplos y recursos adicionales para facilitar nuestro proceso de aprendizaje y desarrollo. Afortunadamente, React Native cuenta con una robusta fuente de información disponible en su página oficial:

[React Native · Learn once, write anywhere](https://reactnative.dev/learn)

La página oficial de React Native ofrece una amplia gama de recursos para desarrolladores, que incluyen documentación detallada, guías de inicio rápido, tutoriales paso a paso y una comunidad activa de desarrolladores que comparten sus experiencias y conocimientos. Aquí podemos encontrar todo lo necesario para familiarizarnos con los conceptos básicos de React Native y comenzar a construir nuestras propias aplicaciones.

Además, la página oficial suele ser actualizada regularmente para reflejar las últimas características, mejoras y mejores prácticas en el desarrollo de aplicaciones en React Native. Esto garantiza que los desarrolladores tengan acceso a la información más actualizada y relevante en todo momento.

Requisitos Previos

Antes de sumergirnos en el desarrollo de aplicaciones con React Native, es fundamental asegurarnos de tener configurado nuestro entorno de desarrollo. Para ello, necesitamos tener instalado Node.js y un administrador de paquetes adecuado. Node.js es una plataforma de ejecución de JavaScript que nos permite ejecutar código JavaScript fuera de un navegador web.

La instalación de Node.js también incluye automáticamente el administrador de paquetes Node Package Manager (npm), que utilizaremos para instalar y gestionar las dependencias de nuestros proyectos.

Para instalar Node.js, podemos dirigirnos a la página oficial de descargas:

[Node.js — Download Node.js® \(nodejs.org\)](https://nodejs.org/en/download)

En esta página, podemos seleccionar la versión de Node.js que deseamos instalar, así como el sistema operativo que estamos utilizando. Es recomendable optar por la versión LTS (Long-Term Support) para garantizar la estabilidad y compatibilidad a largo plazo de nuestro entorno de desarrollo.

Para verificar que Node.js se ha instalado correctamente en tu sistema, puedes utilizar la línea de comandos. A continuación, te mostraré cómo hacerlo utilizando PowerShell, pero lo mismo te servirá en cualquier terminal. Ten en cuenta que la apariencia puede variar dependiendo de la configuración de tu entorno.

En mi caso estoy utilizando [Home | Oh My Posh](#)

```
node -v
```

```
npm -v
```

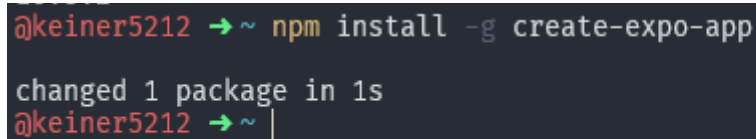
```
PowerShell 7.4.2
Loading personal and system profiles took 737ms.
@keiner5212 → ~ node -v
v20.11.1
@keiner5212 → ~ npm -v
10.5.2
@keiner5212 → ~ |
```

Instalación del Paquete expo de Forma Global

Una vez que hemos verificado la instalación de Node.js y npm en nuestro sistema, el siguiente paso es instalar el paquete expo de forma global. Expo es una plataforma de código abierto para la construcción de aplicaciones móviles nativas con React Native y herramientas web. Expo-cli es una interfaz de línea de comandos que nos permite crear, desarrollar y administrar proyectos de Expo.

Para instalar expo de manera global en tu sistema, puedes ejecutar el siguiente comando en tu terminal:

```
npm install -g create-expo-app
```



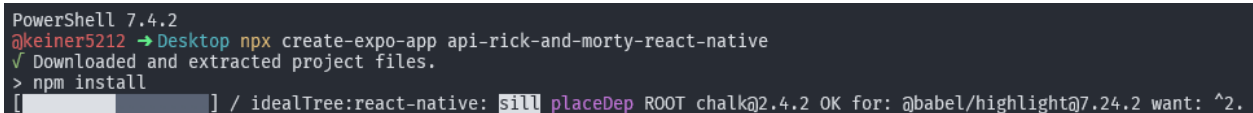
```
@keiner5212 → ~ npm install -g create-expo-app
changed 1 package in 1s
@keiner5212 → ~ |
```

Este comando utilizará npm para instalar create-expo-app de manera global, lo que significa que estará disponible en tu sistema en cualquier directorio.

Creación del Primer Proyecto

Ahora que ya tenemos todo configurado, podemos crear un proyecto ejecutando el siguiente comando en la ubicación deseada (asegúrate de estar en la carpeta en la que quieres crear el proyecto):

```
npx create-expo-app <project name>
```



```
PowerShell 7.4.2
@keiner5212 → Desktop npx create-expo-app api-rick-and-morty-react-native
√ Downloaded and extracted project files.
> npm install
[██████████] / idealTree:react-native: sill placeDep ROOT chalk@2.4.2 OK for: @babel/highlight@7.24.2 want: ^2.
```

Este comando utilizará npx para crear un nuevo proyecto de Expo con el nombre que le indiques, en este caso **api-rick-and-morty-react-native**. Asegúrate de reemplazar este nombre con el que desees para tu proyecto.

Nota: Esto automáticamente creará un repositorio de [Git](#) para el manejo de versiones.

```
✓ Your project is ready!

To run your project, navigate to the directory and run one of the following npm commands.

- cd api-rick-and-morty-react-native
- npm run android
- npm run ios # you need to use macOS to build the iOS project - use the Expo app if you need to do iOS development with
  out a Mac
- npm run web
@keiner5212 → Desktop |
```

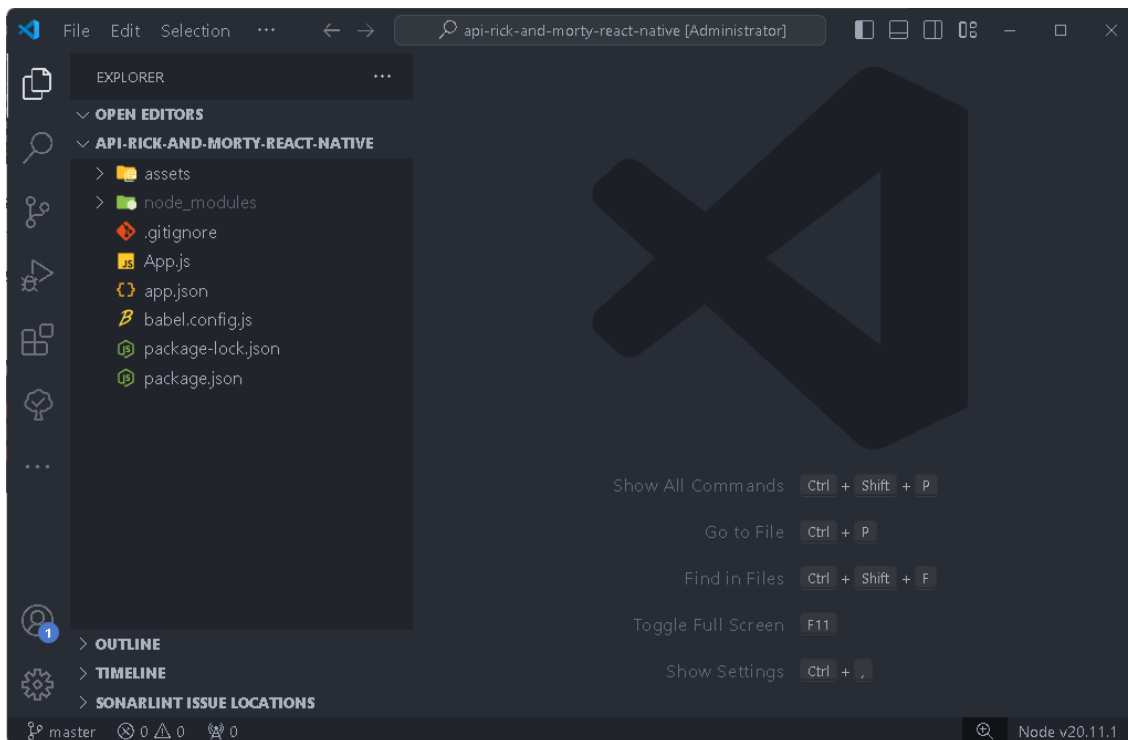
Con esto, habrás creado tu primer proyecto de React Native con Expo y estarás listo para comenzar a desarrollar tu aplicación móvil.

Empieza el desarrollo

Ahora puedes abrir el proyecto con un editor de código, por ejemplo [Visual Studio Code](#)

```
out a Mac
- npm run web
@keiner5212 → Desktop cd .\api-rick-and-morty-react-native\
@keiner5212 → api-rick-and-morty-react-native git(master) code .
@keiner5212 → api-rick-and-morty-react-native git(master) |
```

Ahora podemos ver los archivos que se nos han creado para el proyecto:



assets: Esta carpeta almacena los recursos estáticos de la aplicación, como imágenes, archivos de audio, fuentes de texto, etc.

node_modules: Esta carpeta contiene todas las dependencias del proyecto instaladas a través de npm. Aquí se encuentran todos los paquetes de terceros que la aplicación utiliza.

.gitignore: Este archivo especifica qué archivos y carpetas deben ignorarse por Git, el sistema de control de versiones. Esto suele incluir archivos generados automáticamente y dependencias instaladas.

App.js: Este archivo es el punto de entrada principal de la aplicación. Contiene el código principal de la aplicación, incluyendo la lógica de renderizado de componentes y el enrutamiento de la aplicación.

app.json: Este archivo es utilizado por Expo para configurar la aplicación. Contiene metadatos importantes como el nombre de la aplicación, la descripción, las versiones de SDK de Expo compatibles, las configuraciones de plataforma, entre otros.

babel.config.js: Este archivo contiene la configuración de Babel, una herramienta de compilación de JavaScript utilizada para transformar el código JavaScript moderno en una forma que pueda ser ejecutada por versiones más antiguas de los navegadores y entornos Node.js.

package-lock.json: Este archivo es generado automáticamente por npm y contiene una lista detallada de todas las dependencias instaladas en el proyecto, así como sus versiones específicas. Ayuda a garantizar la replicabilidad de las instalaciones de paquetes en diferentes entornos.

package.json: Este archivo es el archivo de configuración principal de npm para el proyecto. Contiene metadatos sobre el proyecto, como el nombre, la versión, las dependencias del proyecto, los scripts de ejecución, entre otros.

Ejecuta el proyecto

Para ejecutar el proyecto debes usar el comando

```
npm start
```

```
@keiner5212 → api-rick-and-morty-react-native git(master) npm start
> api-rick-and-morty-react-native@1.0.0 start
> expo start

Starting project at D:\Users\Administrator\Desktop\api-rick-and-morty-react-native
Starting Metro Bundler
|
```

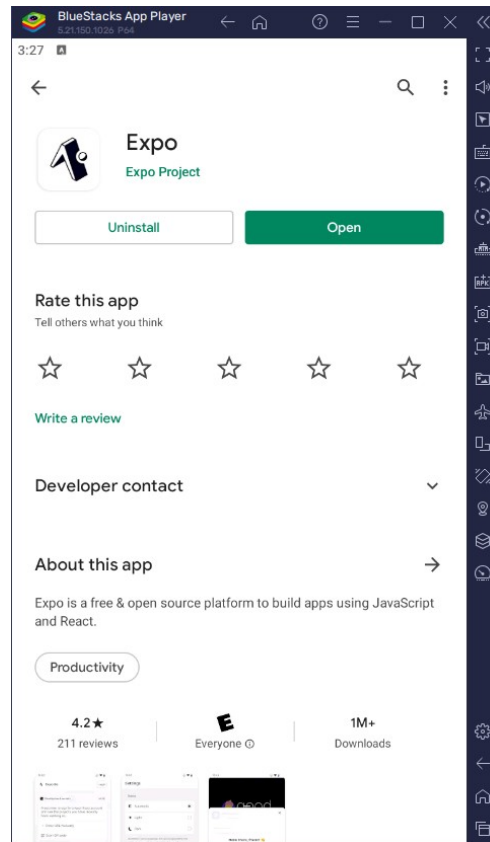
Este comando iniciará el servidor de desarrollo de React Native y Expo. Verás una salida similar a la siguiente:



Visualización del Proyecto en Ejecución

Al desarrollar una aplicación móvil con React Native y Expo, es esencial poder visualizar y probar la aplicación en un dispositivo móvil o en un emulador. Aunque hay varias opciones disponibles, me centraré en una de las formas más sencillas y rápidas de hacerlo: mediante la aplicación Expo Go en dispositivos móviles.

Expo Go en Dispositivos Móviles



Expo Go es una aplicación gratuita disponible en las tiendas de aplicaciones para Android e iOS. Permite cargar y ejecutar aplicaciones desarrolladas con Expo directamente en tu dispositivo móvil. Para visualizar tu proyecto en ejecución, sigue estos pasos:

Abre la aplicación Expo Go en tu dispositivo móvil.

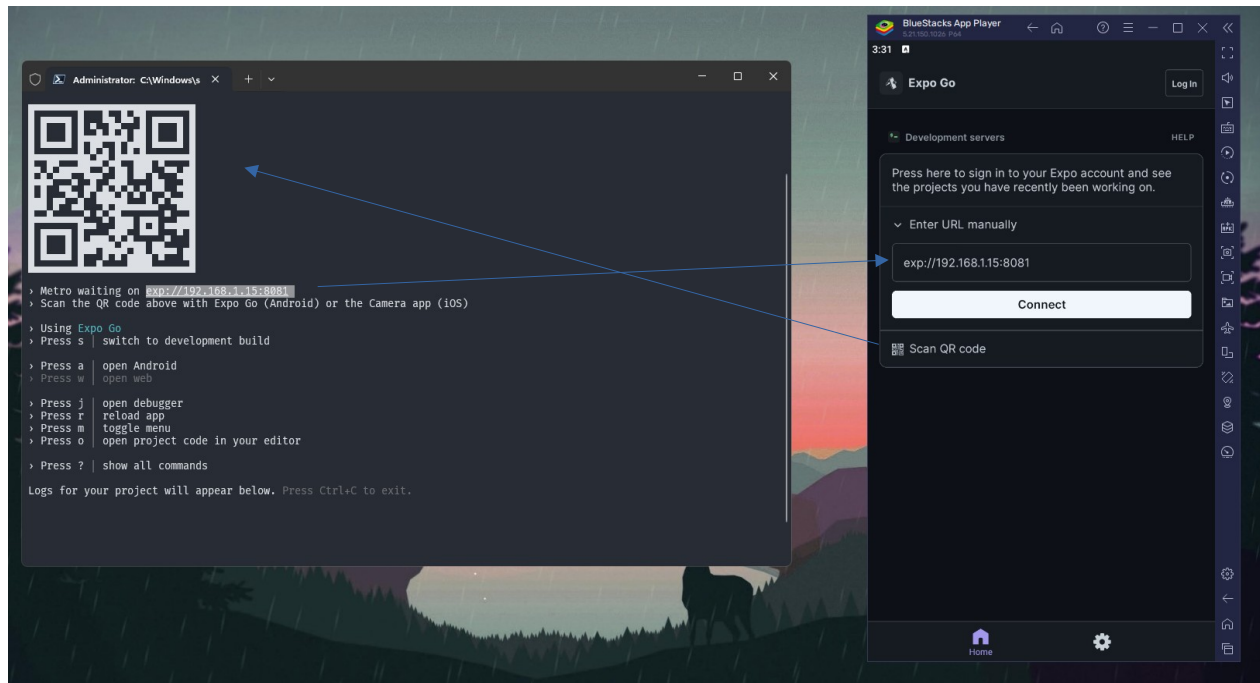
Escanea el código QR que se muestra en la terminal después de ejecutar el comando `npm start` o `expo start`. Esto cargará tu aplicación en Expo Go y comenzará a ejecutarla en tu dispositivo.

Una vez que la aplicación se haya cargado correctamente, podrás interactuar con ella y probar sus funcionalidades en tiempo real.

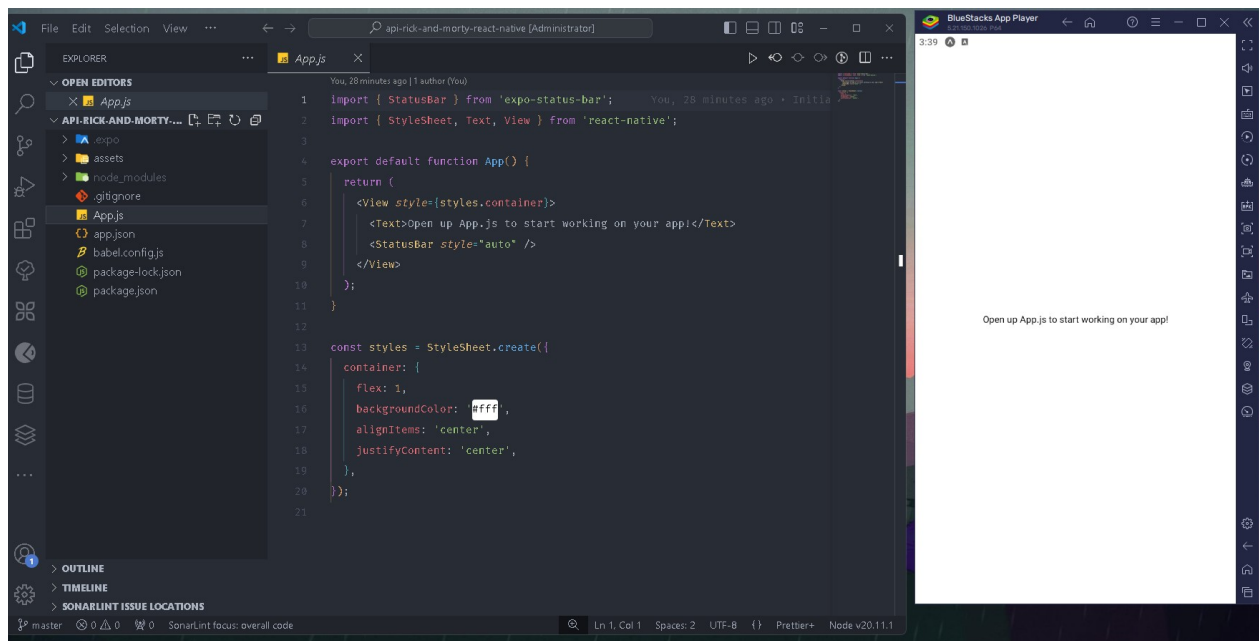
Comentarios sobre Configuraciones Adicionales

Es importante mencionar que, si bien Expo Go es una forma rápida y conveniente de visualizar tu aplicación en un dispositivo móvil, existen otras opciones disponibles para configurar emuladores de dispositivos móviles en tu computadora. Sin embargo, configurar emuladores puede ser un

proceso más complejo y extenso, que puede variar dependiendo del sistema operativo y las herramientas que estés utilizando así que no abordaremos ese tema aquí.



Una vez conectado podemos ver nuestra aplicación corriendo en tiempo real:



En el desarrollo web tradicional, HTML (HyperText Markup Language) se utiliza para definir la estructura y el contenido de una página web. Se hacen uso extensivo de etiquetas HTML como `<div>`, `<p>`, ``, entre otras, para definir los elementos de la interfaz de usuario.

Por otro lado, en el desarrollo de aplicaciones móviles con React Native, no se utilizan las mismas etiquetas HTML que en el desarrollo web. En su lugar, se utilizan componentes específicos de React Native que representan elementos de la interfaz de usuario móvil. Estos componentes están diseñados para renderizar de manera nativa en dispositivos móviles, lo que significa que se traducen directamente a elementos de la interfaz de usuario nativa de iOS y Android.

Paralelismos entre HTML y React Native

Aunque las etiquetas y componentes son diferentes entre HTML y React Native, existen paralelismos entre ambos que pueden ayudar a entender cómo se estructuran las aplicaciones móviles con React Native:

View y `<div>`: En HTML, utilizamos la etiqueta `<div>` para crear contenedores o divisiones en la página web. En React Native, utilizamos el componente View para un propósito similar, es decir, para agrupar otros elementos y aplicar estilos.

Text y `<p>`: La etiqueta `<p>` se utiliza en HTML para representar un párrafo de texto. En React Native, utilizamos el componente Text para renderizar texto en la pantalla de un dispositivo móvil.

Además, es importante destacar que al igual que en React, React Native también utiliza JSX (JavaScript XML).

Estilos en React Native

En React Native no se pueden importar archivos CSS como en el desarrollo web tradicional. Esto se debe a que React Native utiliza un modelo de estilo diferente al de las aplicaciones web. En lugar de CSS, en React Native se utilizan objetos de estilo de JavaScript para definir los estilos de los componentes.

En React Native, puedes definir estilos directamente en línea usando el atributo `style` de los componentes, o puedes crear objetos de estilo reutilizables que se pueden aplicar a múltiples componentes. Por ejemplo:

```
12  const styles = StyleSheet.create({
13    container: {
14      flex: 1,
15      backgroundColor: '#fff',
16      alignItems: 'center',
17      justifyContent: 'center',
18    },
19  });
20
```

Integración de la API de Rick and Morty en tu Proyecto de React Native

En el proyecto que he realizado como ejemplo, he integrado la API de Rick and Morty para acceder a datos relacionados con el universo de la serie de televisión "Rick and Morty". Esta API pública proporciona información detallada sobre personajes, episodios y ubicaciones presentes en la serie.

API: [The Rick and Morty API](https://rickandmortyapi.com/api)

Algunos aspectos clave a tener en cuenta sobre la API de Rick and Morty son los siguientes:

Recursos Disponibles: La API ofrece una variedad de recursos que se pueden acceder mediante endpoints específicos. Estos recursos incluyen personajes, episodios y ubicaciones, cada uno con su propio conjunto de datos asociados.

```
GET https://rickandmortyapi.com/api

{
  "characters": "https://rickandmortyapi.com/api/character",
  "locations": "https://rickandmortyapi.com/api/location",
  "episodes": "https://rickandmortyapi.com/api/episode"
}
```

Solicitudes HTTP: Para interactuar con la API, he utilizado solicitudes HTTP, aprovechando la biblioteca axios en mi aplicación de React Native. Esto me permite realizar solicitudes GET para obtener datos de la API y procesar las respuestas de manera eficiente.

Documentación Detallada: Antes de comenzar a trabajar con la API, revisé cuidadosamente la documentación oficial proporcionada por el equipo de Rick and Morty. Esta documentación detalla la estructura de los endpoints, los parámetros de consulta disponibles y los formatos de respuesta esperados, lo que facilita el proceso de integración.

Gestión de Datos: Una vez que obtengo los datos de la API, los proceso y los integro en mi aplicación de React Native según sea necesario. Utilizo estructuras de datos adecuadas y técnicas de gestión de estado para almacenar y manipular los datos de manera eficiente dentro de la aplicación.

Optimización y Rendimiento: Durante el desarrollo, he tenido en cuenta la optimización del rendimiento de la aplicación, minimizando el número de solicitudes a la API y optimizando la carga y renderizado de datos para mejorar la experiencia del usuario.

Puedes encontrar el proyecto en: [keiner5212/api-rick-and-morty-react-native \(github.com\)](https://github.com/keiner5212/api-rick-and-morty-react-native)