

REACT NATIVE

Antonio Manuel Cepeda, Francisco A. Lorente

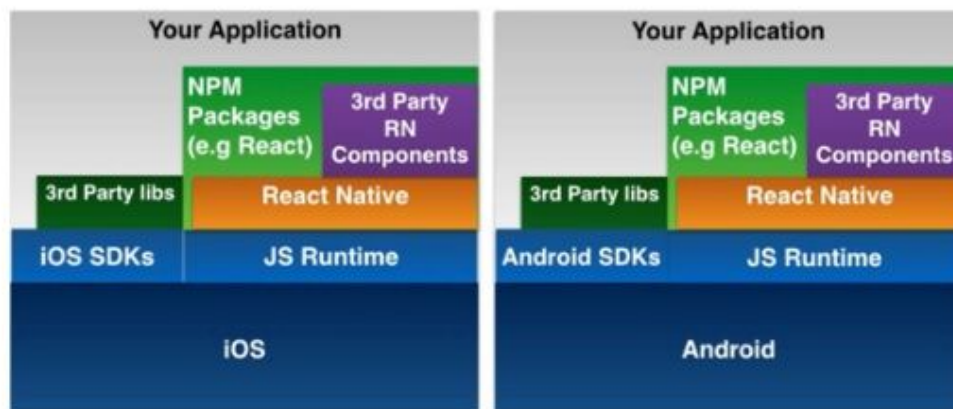
Arquitectura, estructura y características de la tecnología utilizada.

Primeramente, por introducir un poco a React Native, es un framework (algunos defienden que solamente, librería) para móviles, open-source, creado por Facebook.

La clave de la arquitectura de React Native es que permite que JavaScript se ejecute con el sistema operativo iOS o Android de la misma manera que lo hace el código nativo, facilitando el desarrollo por ende.

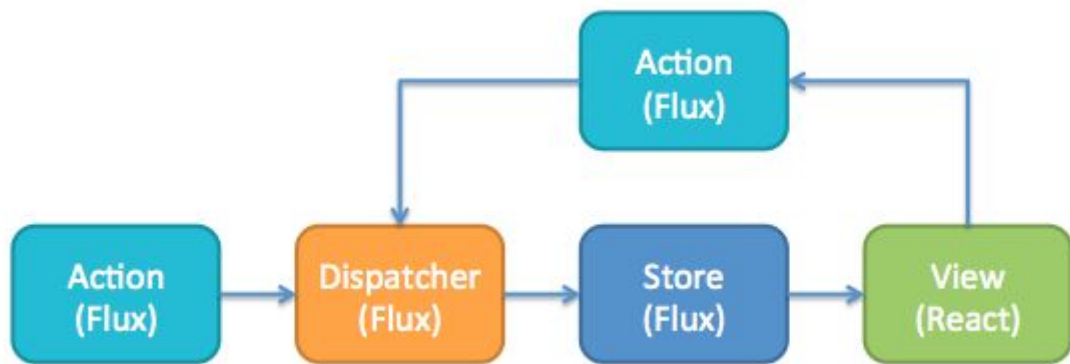
Debido a que JavaScript no es el lenguaje que se ejecuta de forma nativa en el teléfono, debemos usar una técnica llamada puenteo (bridging) para permitir que JavaScript se ejecute y se comuniqué con el procesador del teléfono.

React Native App Architecture



Ojo. El código no acaba siendo nunca código nativo, sino que se interpreta en tiempo de ejecución de la app (arquitectura Flux, de Facebook), y esto puede acabar perjudicando al rendimiento de la aplicación. Esto en cuanto a la lógica de negocio. En cuanto a las vistas, se aplica un metalenguaje que facilita la colocación y montaje de los distintos componentes visuales. Esto hace que las vistas sean idénticas tanto en Android como en iOS.

Volviendo al tema de Flux, ¿cómo funciona?



Flux es una arquitectura en la que el flujo de datos unidireccional. Los datos viajan desde la vista por medio de acciones y llegan a un Store desde el cual se vuelve a actualizar la vista de nuevo. Teniendo un único camino, y un sitio donde se almacena el estado de la aplicación, es más sencillo depurar errores y saber qué está pasando en cada momento.

Vista

La vista serían los componentes web.

Store

La *Store* sería lo más parecido al modelo de la aplicación. Guarda los datos/estado de la aplicación y en Flux puede haber varias. No hay métodos en la *Store* que permitan modificar los datos en ella, eso se hace a través de *dispatchers* y acciones.

Acciones

Un acción es simplemente un objeto JavaScript que indica una intención de realizar algo y que lleva datos asociados si es necesario. Por ejemplo si tenemos una aplicación tipo *Carrito de la compra*, y añadimos un ítem al carrito, la acción que representaría esto sería:

```
{
  type: 'ADD_ITEM',
  item: item
}
```

Dispatcher

Las acciones como la anterior son enviadas a un *dispatcher* que se encarga de *dispararla* o propagarla hasta la Store.

La vista es la que se encarga de enviar las acciones al *dispatcher*.

Un *dispatcher* no es más que un mediador entre la *Store* o *Stores* y las acciones.

En Resumen, el patrón FLUX sigue el siguiente recorrido:

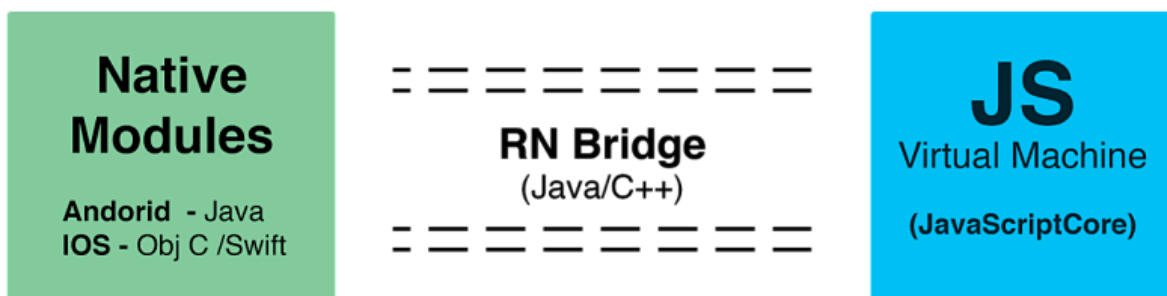
- La vista, mediante un evento envía una acción con la intención de realizar un cambio en el estado
- La acción contiene el tipo y los datos (si los hubiere) y es enviada al dispatcher.
- El dispatcher propaga la acción al Store y se procesa en orden de llegada.
- El Store recibe la acción y dependiendo del tipo recibido, actualiza el estado y notifica a las vistas de ese cambio.
- La vista recibe la notificación y se actualiza con los cambios.

Todo en un único sentido.

Proseguimos con algunas características, habiendo explicado el modelo Flux

El código de React Native, al igual que en React, se organiza en componentes, lo cual permite una alta reutilización de código y agiliza el proceso de desarrollo.

También permite añadir componentes creados por la comunidad los cuales disponen de una amplia documentación y mantenimiento, aparte de una gran cantidad de hilos en foros resolviendo problemas y dudas.



Características:

- **Compatibilidad Cross-Platform:** El mismo código puede ser ejecutado tanto en IOS como en Android, facilitando el desarrollo de aplicaciones simultáneas para los dos sistemas operativos.
- **Funcionalidad nativa:** Las aplicaciones creadas mediante React Native funcionan de la misma forma que una programada con lenguaje Nativo propio. Son indiferenciables.
- **Actualizaciones instantáneas (para desarrollo y/o test):** Con la extensión de JavaScript, los desarrolladores tienen la flexibilidad de subir los cambios contenidos en la actualización directamente al dispositivo del usuario sin tener que pasar por las tiendas de aplicaciones propias de cada sistema y sus tediosos ciclos de procesos previos. Esta característica está únicamente permitida para el desarrollo, ya que es ilegal para el entorno de producción y el castigo puede llegar a la retirada de la aplicación.
- **Sencilla curva de aprendizaje:** React Native es extremadamente fácil de leer y aprender, ya que se basa en los conceptos fundamentales del lenguaje JavaScript, siendo bastante intuitivo para expertos y novatos.
- **Experiencia positiva para el desarrollador:** Ofrece varias características que ayudarán al desarrollador en su tarea, como el Hot reloading, que nos refresca la app cada vez que guardamos un cambio, el uso del flexbox layout engine el cual nos permite abstraernos de los tediosos detalles de la generación de cada uno de los layouts correspondientes a IOS y Android, así como el debugger de las herramientas de desarrollador del navegador Google Chrome, facilitando la depuración del código.

Pros y Contras de React Native:

Pros de React Native

- React Native ofrece varios elementos listos para aplicar que pueden acelerar el tiempo de desarrollo.
- Recarga en caliente: podemos recargar nuestra aplicación rápidamente, sin volver a compilar.
- Uso directo de código nativo para optimizar nuestra aplicación a un gran nivel.
- Código compartido en ambas plataformas: iOS y Android.
- Desarrollo de aplicaciones más rápido con elementos preconstruidos.
- Acceso a funcionalidades nativas como cámara, acelerómetro, etc.
- UI móvil de alta calidad.

Contras de React Native

- La navegación de la aplicación móvil no sería equivalente a la navegación nativa. Una navegación diseñada en React Native no es tan suave.
- Al igual que Xamarin, React Native incluso nos brinda la oportunidad de desarrollar aplicaciones de calidad superior. Pero las aplicaciones creadas con React Native

son más lentas que las aplicaciones nativas de Android creadas con Java y las aplicaciones nativas de iOS creadas con Objective-C y Swift.

Fuentes:

<https://www.paradigmadigital.com/dev/desarrollando-aplicaciones-moviles-nativas-con-react-native/>

<https://www.bit.es/knowledge-center/react-native-una-nueva-arquitectura-para-la-generacion-de-apps-moviles/>

<https://www2.deloitte.com/es/es/pages/technology/articles/que-es-react-native.html>

<https://www.bit.es/knowledge-center/react-native-rendimiento-primeros-pasos-y-casos-de-uso/>

<https://medium.com/monoku/react-native-est%C3%A1-cambiando-la-forma-en-la-que-desarrollamos-aplicaciones-m%C3%B3viles-de7bbb742555>

Tipo de lenguajes de programación y estilos para la realización de una aplicación con alguna de estas tecnologías.

Cómo lenguaje, admite el uso de JavaScript, con etiquetas propias de React Native, parecidas a las etiquetas de HTML.

También admite el uso de una hoja de estilos SASS y el uso de Node para la gestión de paquetes.

No usa CSS propiamente dicho, sino que usamos un subconjunto de propiedades de CSS, las cuales podemos encontrar en su página web:

<https://facebook.github.io/react-native/docs/layout-props.html>

También es necesario tener instalado el Android / IOS SDK para poder montar la aplicación en un dispositivo IOS o Android o emularlo en su defecto.

<https://medium.com/monoku/react-native-est%C3%A1-cambiando-la-forma-en-la-que-desarrollamos-aplicaciones-m%C3%B3viles-de7bbb742555>

Características, instalación y configuración correcta de los distintos Entornos de Desarrollo utilizados para la realización de aplicaciones para estas plataformas.

Entornos utilizados:

Visual Studio Code, Android Studio

Instalación para poder realizar la aplicación:

Vamos a necesitar **Node**, **Python 2**, un **JDK** y **Android Studio**. Desde la web de React Native se nos aconseja (aunque no es obligatorio) usar **Chocolatey**, un gestor de paquete de Windows.

Con Chocolatey instalado, tendremos que introducir por cmd el siguiente comando:

```
choco install -y nodejs.install python2 jdk8
```

Tenemos que asegurarnos de que la versión del jdk sea la versión 8 o superior, y que la versión de Node sea la 8.3 o superior.

Una vez hecho este paso, introducimos por consola el siguiente comando para instalar React-Native

```
npm install -g react-native-cli
```

Con esto ya tenemos React-Native.

Como nosotros vamos a desarrollar una app para móvil para Android, tendremos ahora que instalar el entorno de Android para poder desarrollar para él. Primeramente, descargamos **Android Studio**. Escogemos una instalación personalizada, y nos aseguramos de que estén marcadas las siguientes opciones:

- Android SDK
- Android SDK Platform
- Performance (Intel ® HAXM)
- Android Virtual Device

Proseguimos con la instalación.

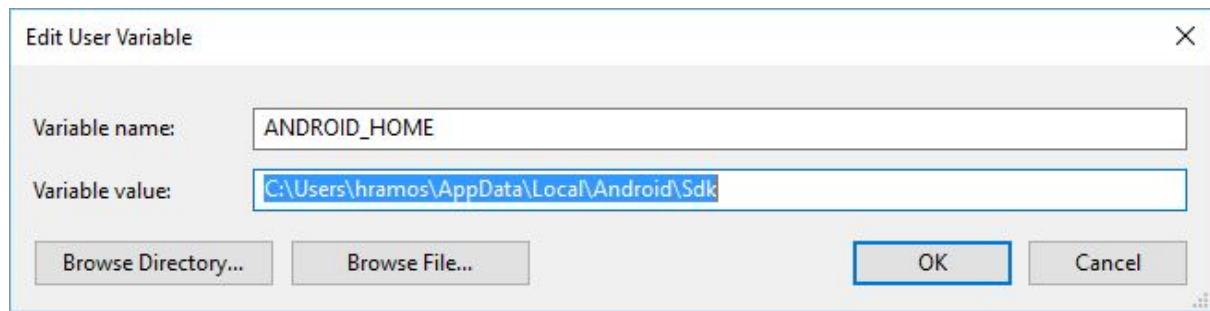
Instalamos el **SDK**, la **versión 9 (Pie)**. Tiene que ser ésta en particular, ni superior ni inferior. Versión 9 (Pie).

Seleccionamos el botón de 'Configuración', y le damos a 'SDK Manager'-'>'SDK Platforms', marcamos 'Show Package Details', expandimos donde pone **Versión 9 (Pie)** y nos aseguramos de que los siguiente elementos estén marcados:

- Android SDK Platform 28
- Intel x86 Atom_64 System Image **Ó** Google APIs Intel x86 Atom System Image

Tras eso, clicamos en la pestaña de 'SDK Tools'-'>'Android SKD Build Tools' y nos aseguramos de que la versión 28.0.3 está seleccionada. Le damos a aplicar y esperamos a que se complete el proceso.

Con esto hecho, debemos asegurarnos de que las siguientes variables de entorno estén creadas, o las creamos si no lo estuviesen.




Podemos crear un proyecto desde la terminal de Visual Studio (o desde cmd) para comprobar que está todo en orden:

```
react-native init AwesomeProject
```

Configuración correcta de los emuladores para poder probar aplicaciones en estas plataformas.

Pasamos ahora a configurar el emulador.

Usaremos el de **Android Studio**.

Buscamos un icono parecido a este (todo esto, con un proyecto abierto, da igual si no es el que hemos creado) , en la barra superior, por la parte derecha.

Le damos a '**Create Virtual Device**', escogemos el teléfono en el que queramos emular, y seleccionamos **Pie API 28 level image**. Con esto ya tendremos la herramienta de emulación preparada. Hacemos click en el botón de play verde que nos sale para 'encender' el dispositivo virtual.

Desde una terminal de Visual Studio (o el editor que desees), desde la carpeta de proyecto que hayamos creado, lanzamos el siguiente comando:

```
react-native run-android
```

Si todo ha ido correctamente, nuestra aplicación debe lanzarse correctamente en el emulador.

