React Native

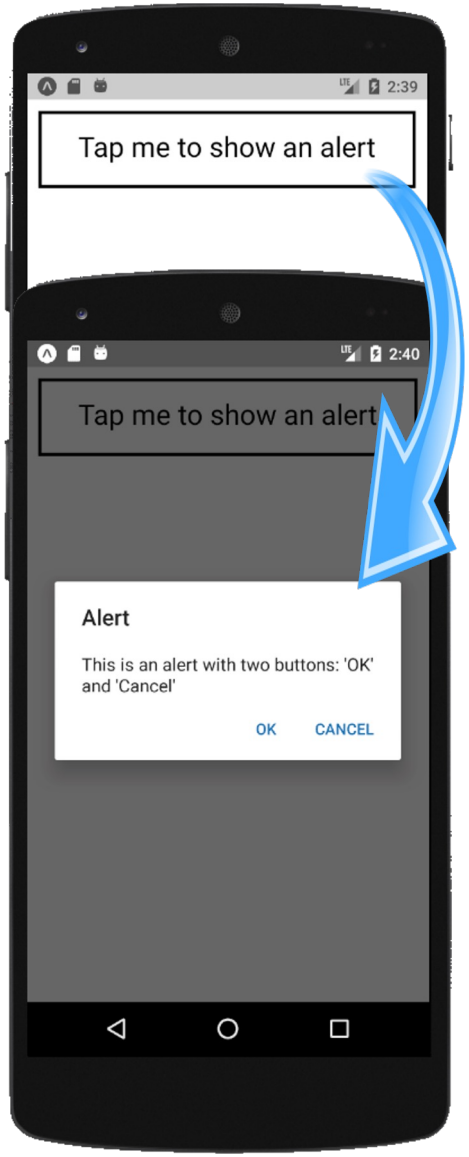
APIs avanzadas

Álvaro Alonso González
Enrique Barra Arias

Alert

- Permite mostrar **diálogos de alerta**
- Documentación:
  https://reactnative.dev/docs/alert

```
Alert.alert(
    "Alert",
    "This is an alert with two buttons: 'OK' and 'Cancel'",
    [
      {text: 'OK', onPress: () => console.log('OK pressed')},
      {text: 'Cancel', onPress: () => console.log('Cancel pressed')}
    ],
    { cancelable: false }
  )
```

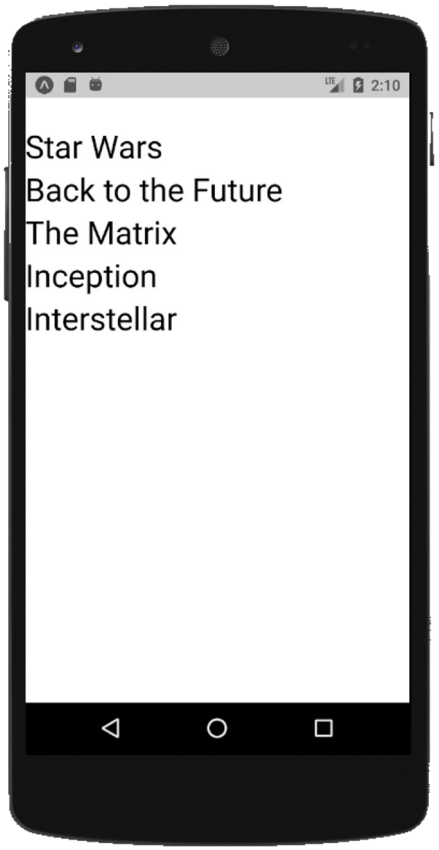


Fetch

- **API JavaScript** para cargar recursos de una red como Internet a partir de sus URLs
- Permite realizar peticiones **HTTP**
- Similar a la API **XMLHttpRequest** soportada en la mayoría de navegadores web
- Las operaciones son **asíncronas**: las funciones de la API Fetch devuelven **promesas**
- React Native también incorpora la API **XMLHttpRequest**, por lo que se puede usar tanto esta API directamente como bibliotecas que dependan de ella (ej: axios)
- Documentación:
  https://reactnative.dev/docs/network
  https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API
  https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest

Fetch: Ejemplo

```
async getMoviesFromAPI(){
  var url =  'https://facebook.github.io/react-native/movies.json';
  var response = await fetch(url,{
    method: 'GET',
    headers: {
      'Accept': 'application/json',
      'Content-Type': 'application/json',
    }
  });
  var responseJson = await response.json();
  return responseJson.movies;
}
```



AsyncStorage

- **API JavaScript** que proporciona a las aplicaciones React Native un sistema de **almacenamiento clave-valor** simple, asíncrono y persistente
- Misma función que proporciona la API HTML5 **LocalStorage** en aplicaciones web
- Todas las funciones de la API devuelven **promesas**
- Implementación
  - *Android*: SQLite o RocksDB *(http://rocksdb.org)*
  - *iOS*: diccionarios (valores pequeños) y ficheros (valores grandes)
- Documentación (mantenido por la comunidad): *https://react-native-async-storage.github.io/async-storage/*

AsyncStorage: Funciones

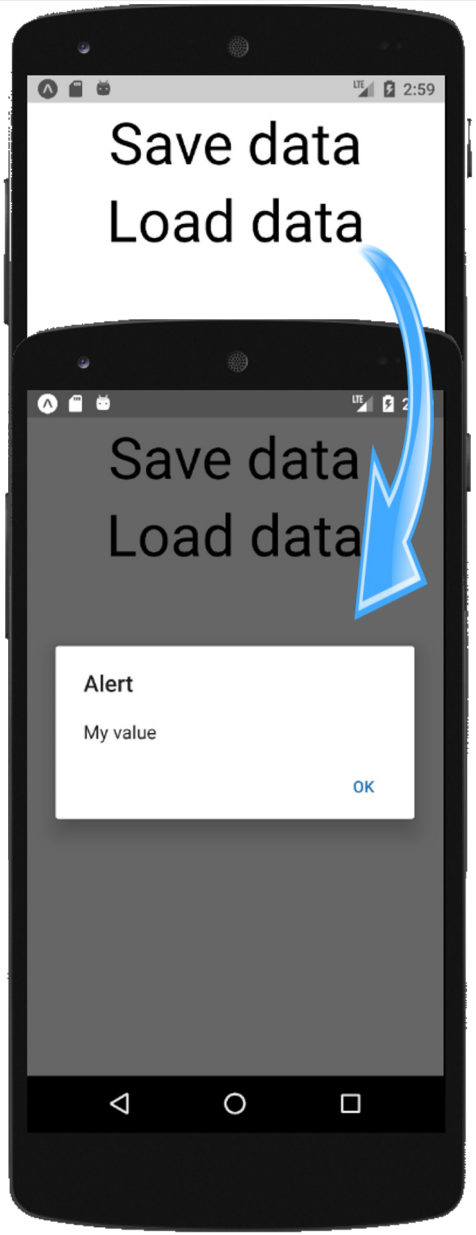
Función	Descripción
getItem (key, callback?)	Obtiene el **valor** para la clave ***key*** e invoca un ***callback*** al finalizar.
setItem (key, value, callback?)	Establece el valor ***value*** para la clave ***key*** e invoca un ***callback*** al finalizar.
removeItem (key, callback?)	Borra el **valor** para la clave ***key*** e invoca un ***callback*** al finalizar.
mergeItem (key, value, callback?)	Hace un merge del valor de la clave ***key*** y el valor ***value*** pasado como parámetro asumiendo que ambos son objetos convertidos a **cadenas JSON**. Guarda el valor resultante en la clave ***key*** e invoca un ***callback*** al finalizar.
Clear (callback?)	Elimina todos los pares clave-valor almacenados con AsyncStorage, incluso los de otras aplicaciones.
getAllKeys (callback?)	Obtiene todas las claves conocidas por la aplicación.

AsyncStorage: Funciones

Función	Descripción
multiGet (keys, callback?)	Obtiene los **valores** correspondientes al array de claves ***keys***. El ***callback*** será invocado con el array de los pares clave-valor encontrados.
multiSet (keyValuePairs, callback?)	Permite almacenar múltiples pares clave-valor proporcionados en el array ***keyValuePairs*** de la forma: *[['clave1', 'valor1'], ['clave2', 'valor2']]* Al finalizar, el ***callback*** se invoca con un array que contiene cualquier error encontrado durante la operación.
multiRemove (keys, callback?)	Borra los valores correspondientes a todas las claves contenidas en el array ***keys***. Al finalizar, el ***callback*** se invoca con un array que contiene cualquier error encontrado durante la operación.
multiMerge (keyValuePairs, callback?)	Permite realizar varios *merges* con una sola operación. Las claves y valores para realizar los merges se pasan en el array ***keyValuePairs***. Al finalizar, el ***callback*** se invoca con un array que contiene cualquier error encontrado durante la operación.

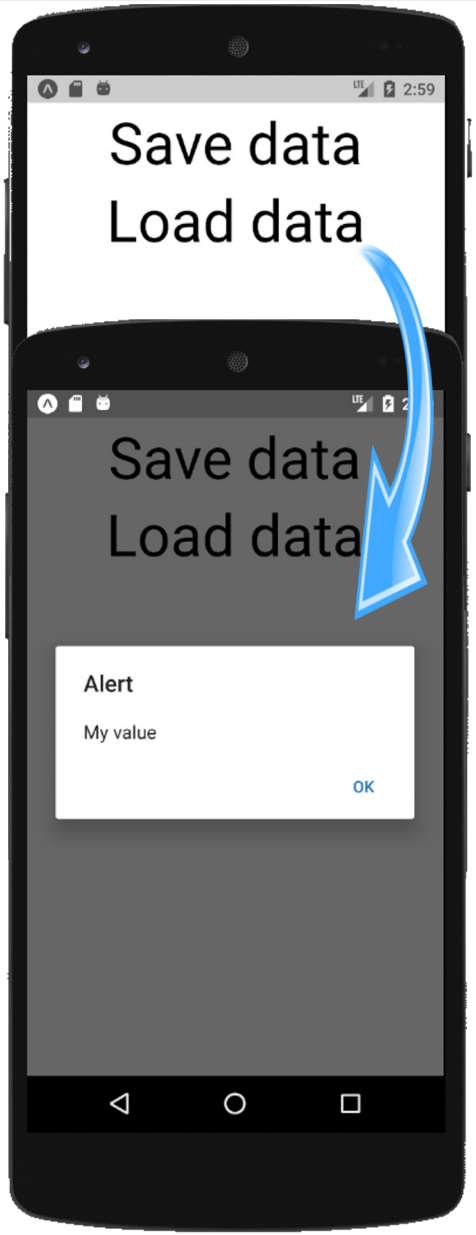
AsyncStorage: Ejemplo

```
import React from 'react';
import { TouchableOpacity, Text, View } from 'react-native';
import AsyncStorage from '@react-native-async-storage/async-storage';

export default function App () {
  async _saveData(){
    try {
      await AsyncStorage.setItem('@HelloWorld:myKey','My value');
    } catch (error) {  // Error saving data  }
  }
  async _loadData(){
    try {
      var value = await  AsyncStorage.getItem('@HelloWorld:myKey');
      if (value !== null){
        alert(value);
      }
    } catch (error) {  // Error retrieving data  }
  }
  [...]
}
```



AsyncStorage: Ejemplo

```
import React from 'react';
import { TouchableOpacity, Text, View } from 'react-native';
import AsyncStorage from '@react-native-async-storage/async-
storage';

export default function App () {
 [...]
  return (
    <View style={{ flex:1, alignItems:'center',
      justifyContent:'flex-start' }}>
     <TouchableOpacity  onPress={_saveData}>
      <Text style={{fontSize: 50}}>Save data</Text>
     </TouchableOpacity>
     <TouchableOpacity  onPress={_loadData}>
      <Text style={{fontSize: 50}}>Load data</Text>
     </TouchableOpacity>
    </View>
  )
}
```



Vibration

- **API JavaScript** que permite activar la vibración de los dispositivos
- Tiene dos funciones:
  - **vibrate**: activa la vibración
  - **cancel**: detiene la vibración
- Las funciones no tienen ningún efecto en aquellos dispositivos que no soporten vibración (ej: emuladores)
- Android permite configurar el tiempo durante el cual vibrará el dispositivo mientras que en iOS el dispositivo vibrará durante una cantidad de tiempo fija
- Documentación:
  https://reactnative.dev/docs/vibration

CameraRoll

- **API JavaScript** que proporciona acceso a la **galería de fotos de la cámara**
- Tiene dos funciones:
  - **getPhotos**: permite obtener fotos y vídeos existentes en la galería
  - **saveToCameraRoll**: permite guardar nuevas fotos y vídeos en la galería
- Las funciones de la API devuelven **promesas**
- Es necesario solicitar **permiso** al usuario
- Documentación:
  https://reactnative.dev/docs/cameraroll

React Native

APIs avanzadas

Álvaro Alonso González
Enrique Barra Arias