

React Native:

React Native es un marco de JavaScript para crear aplicaciones móviles nativas. Utiliza el marco React y ofrece una gran cantidad de componentes y API integrados.

React Native te permite crear aplicaciones móviles usando solo JavaScript. Utiliza el mismo diseño que React, lo que le permite componer una interfaz de usuario móvil enriquecida a partir de componentes declarativos. Con React Native, no creas una aplicación web móvil, una aplicación HTML5 o una aplicación híbrida; usted crea una aplicación móvil real que es indistinguible de una aplicación creada con Objective-C o Java. React Native utiliza los mismos componentes fundamentales de la interfaz de usuario que las aplicaciones normales de iOS y Android. Simplemente junte esos bloques de construcción usando JavaScript y React.

Reaccionar características nativas

Las siguientes son las características de React Native:

- **React** : este es un marco para crear aplicaciones web y móviles utilizando JavaScript.
- **Nativo** : puede utilizar componentes nativos controlados por JavaScript.
- **Plataformas** : React Native es compatible con las plataformas iOS y Android.

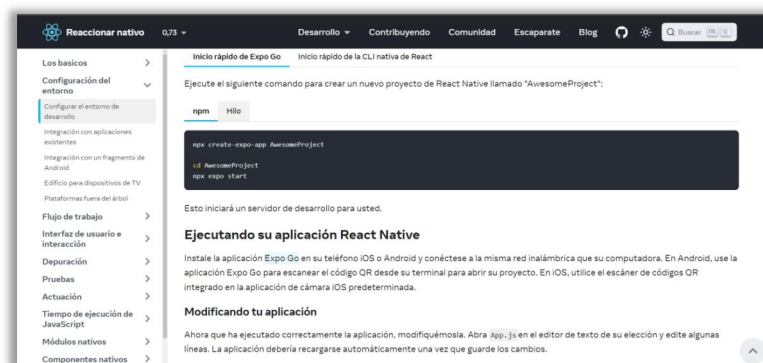
Configurar el entorno de desarrollo

Esta página lo ayudará a instalar y crear su primera aplicación React Native.

<https://reactnative.dev/docs/0.73/environment-setup?os=windows&platform=android&guide=native#jdk-studio>.

Si es nuevo en el desarrollo móvil, la forma más sencilla de comenzar es con Expo Go. Expo es un conjunto de herramientas y servicios creados en torno a React Native y, si bien tiene muchas funciones, la característica más relevante para nosotros en este momento es que puede permitirle escribir una aplicación React Native en cuestión de minutos. Sólo necesitarás una versión reciente de Node.js y un teléfono o emulador. Si desea probar React Native directamente en su navegador web antes de instalar cualquier herramienta, puede probar Snack .

Si ya está familiarizado con el desarrollo móvil, es posible que desee utilizar React Native CLI. Requiere Xcode o Android Studio para comenzar. Si ya tiene una de estas herramientas instalada, debería poder ponerla en funcionamiento en unos minutos. Si no están instalados, debería pasar aproximadamente una hora instalándolos y configurándolos.



Siga estas instrucciones si necesita crear código nativo en su proyecto. Por ejemplo, si está integrando React Native en una aplicación existente, o si ejecutó la "compilación previa" desde Expo para generar el código nativo de su proyecto, necesitará esta sección.

Las instrucciones son un poco diferentes según su sistema operativo de desarrollo y si desea comenzar a desarrollar para iOS o Android. Si quieres desarrollar tanto para Android como para iOS, está bien; puedes elegir uno para empezar, ya que la configuración es un poco diferente.



Instalando dependencias

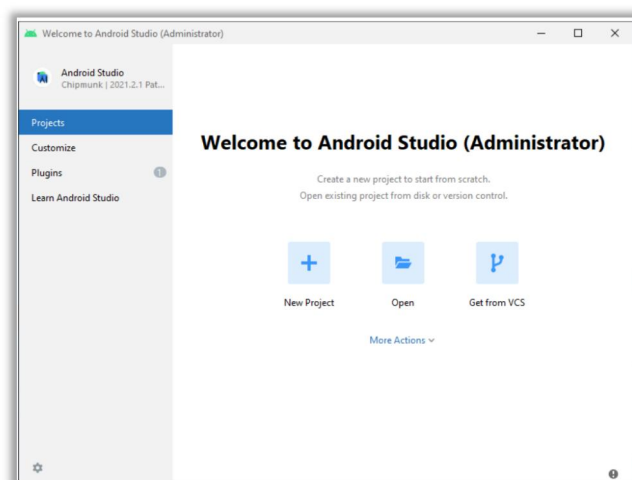
Necesitará Node, la interfaz de línea de comandos de React Native, un JDK y Android Studio.

Si ya instaló Node en su sistema, asegúrese de que sea Node 18 o posterior. Si ya tiene un JDK en su sistema, le recomendamos JDK17. Puede encontrar problemas al utilizar versiones superiores de JDK.

Instale Android Estudio

Descargue e instale Android Studio. Mientras esté en el asistente de instalación de Android Studio, asegúrese de que las casillas junto a todos los siguientes elementos estén marcadas:

- Android SDK
- Android SDK Platform
- Android Virtual Device



Seleccione la pestaña "Plataformas SDK" desde el Administrador de SDK, luego marque la casilla junto a "Mostrar detalles del paquete" en la esquina inferior derecha. Busque y expanda la Android 14 (UpsideDownCake) entrada, luego asegúrese de que los siguientes elementos estén marcados:

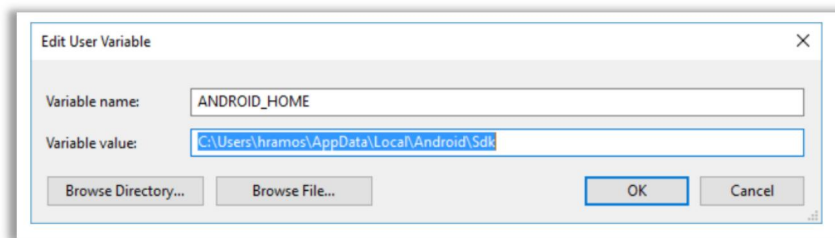
Android SDK Platform 34

Intel x86 Atom_64 System Image Google APIs Intel x86 Atom System Image

A continuación, seleccione la pestaña "Herramientas SDK" y marque la casilla junto a "Mostrar detalles del paquete" aquí también. Busque y expanda la Android SDK Build-Tools entrada, luego asegúrese de que 34.0.0 esté seleccionada.

Finalmente, haga clic en "Aplicar" para descargar e instalar el SDK de Android y las herramientas de compilación relacionadas.

Configure la variable de entorno ANDROID_HOME



Nota: En cuenta de usuario

Agregue herramientas de plataforma a Path

```
%LOCALAPPDATA%\Android\Sdk\platform-tools
```

Reaccionar interfaz de línea de comando nativa

React Native tiene una interfaz de línea de comandos incorporada. En lugar de instalar y administrar una versión específica de la CLI globalmente, le recomendamos acceder a la versión actual en tiempo de ejecución usando `npx`, que se incluye con Node.js. Con `npx react-native <command>`, la versión estable actual de la CLI se descargará y ejecutará en el momento en que se ejecute el comando.

Creando una nueva aplicación

Si instaló anteriormente un react-native-cli paquete global, elimínelo ya que puede causar problemas inesperados:

```
npm uninstall -g react-native-cli @react-native-community/cli
```

React Native tiene una interfaz de línea de comandos incorporada, que puedes usar para generar un nuevo proyecto. Puede acceder a él sin instalar nada globalmente usando npx, que se incluye con Node.js. Creemos un nuevo proyecto React Native llamado "AwesomeProject":

```
npx react-native@latest init AwesomeProject
```

Esto no es necesario si está integrando React Native en una aplicación existente, o si ha instalado Expo en su proyecto, o si está agregando compatibilidad con Android a un proyecto React Native existente (consulte Integración con aplicaciones existentes). También puede utilizar una CLI de terceros para iniciar su aplicación React Native, como Ignite CLI.

[Opcional] Usar una versión o plantilla específica

Si desea iniciar un nuevo proyecto con una versión específica de React Native, puede usar el **`--version`** argumento:

```
npx react-native@X.XX.X init AwesomeProject --version X.XX.X
```

También puedes iniciar un proyecto con una plantilla React Native personalizada con el **`--template`** argumento.

Usando un dispositivo físico

Si tiene un dispositivo Android físico, puede usarlo para el desarrollo en lugar de un AVD conectándolo a su computadora mediante un cable USB y siguiendo las instrucciones <https://reactnative.dev/docs/running-on-device>

Usando un dispositivo virtual

Si usa Android Studio para abrir `./AwesomeProject/android`, puede ver la lista de dispositivos virtuales Android (AVD) disponibles abriendo el "Administrador AVD" desde Android Studio

Si instaló Android Studio recientemente, es probable que necesite crear un nuevo AVD . Seleccione "Crear dispositivo virtual...", luego elija cualquier teléfono de la lista y haga clic en "Siguiente", luego seleccione la imagen UpsideDownCake API Nivel 34.

Haga clic en "Siguiente" y luego en "Finalizar" para crear su AVD. En este punto, debería poder hacer clic en el botón triangular verde al lado de su AVD para iniciarlo y luego continuar con el siguiente paso.

Ejecutando su aplicación React Native

Paso 1: inicia Metro

Metro es la herramienta de compilación de JavaScript para React Native. Para iniciar el servidor de desarrollo de Metro, ejecute lo siguiente desde la carpeta de su proyecto:

```
npm start
```

Paso 2: Inicie su aplicación

Deje que Metro Bundler funcione en su propia terminal. Abra una nueva terminal dentro de la carpeta de su proyecto React Native. Ejecute lo siguiente:

```
npm run android
```

Crear un nuevo proyecto

Crear carpetas y actualizar react-native-cli

```
$\backslash$Tutorialspoint>cd Desktop
```

```
$\Tutorials\point\Desktop>
```

- Actualizar react-native-cli

```

C:\Administrador Símbolo del sistema>
Microsoft Windows [Versión 10.0.19045.4046]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\WINDOWS\system32>

D:\>cd D:\A_CURSOS\2024_V_Native\TutorialPoint\Desktop

D:\A_CURSOS\2024_V_Native\TutorialPoint\Desktop>npm uninstall -g react-native-cli @react-native-community/cli
up to date in 728ms

D:\A_CURSOS\2024_V_Native\TutorialPoint\Desktop>

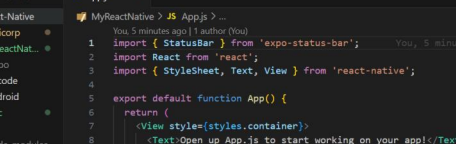
```

- Crear Proyecto MyReactNative

```
npx react-native@latest init MyReactNative
```

[illegible]

- Desplegar en vscode



The screenshot shows a code editor with a file explorer on the left. The file explorer lists the following files and folders: React-Native, credcorp, MyReactNative, expo, vscode, android, doc, ios, node_modules, gitattributes, gitignore, JS App.js (selected), app.json, babel.config.js, index.js, metro.config.js, package.json, and yarn.lock. The main editor area displays the content of JS App.js, which is a React Native app component. The code is as follows:

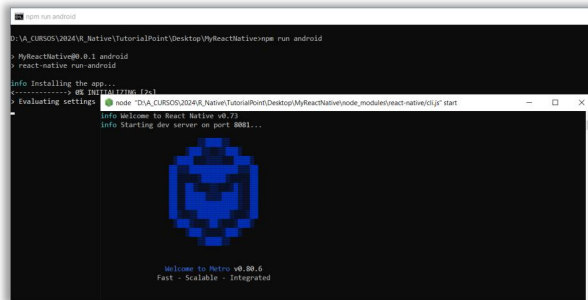
```

1  import { StatusBar } from 'expo-status-bar';
2  import React from 'react';
3  import { StyleSheet, Text, View } from 'react-native';
4
5  export default function App() {
6    return (
7      <View style={styles.container}>
8        <Text>Open up App.js to start working on your app!</Text>
9        <StatusBar style="auto" />
10      </View>
11    );
12  }
13
14  const styles = StyleSheet.create({
15    container: {
16      flex: 1,
17      backgroundColor: '#fff',
18      alignItems: 'center',
19      justifyContent: 'center',
20    },
21  });
22

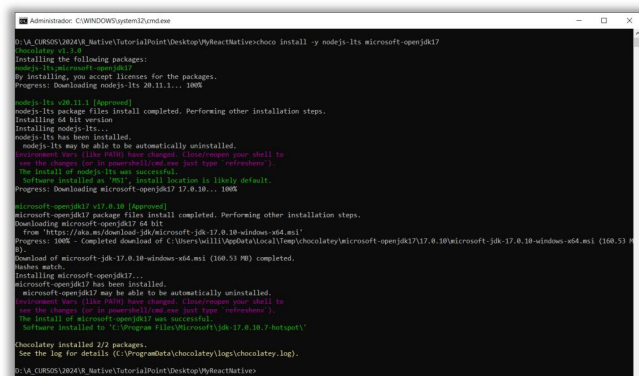
```

Correr el proyecto

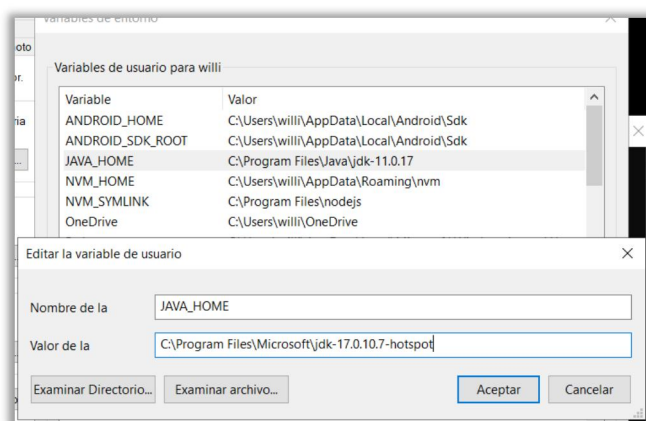
npm run android

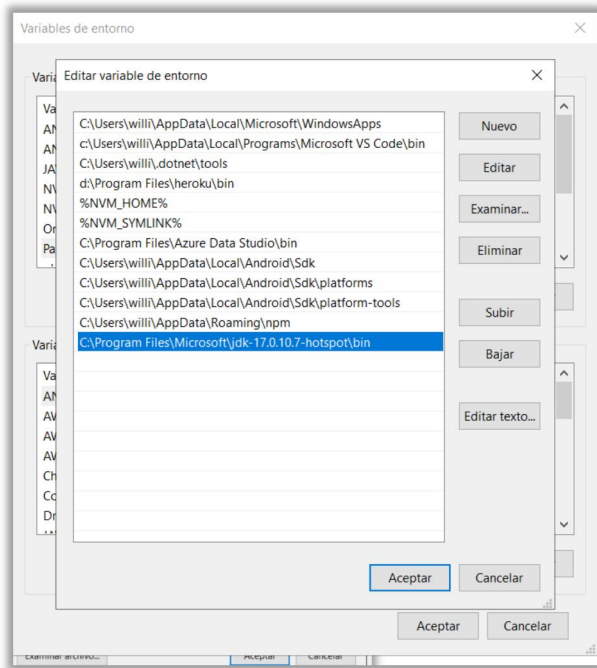


- Da error por JDK. Se probó con yarn igual da el mismo error
- Se procede a instalar la versión de java kit de desarrollo Java SE (JDK):
choco install -y nodejs-lts microsoft-openjdk17



- Actualizar parámetros de Java en Variables de ambiente de usuario y del sistema
C:\Program Files\Java\jdk-11.0.17





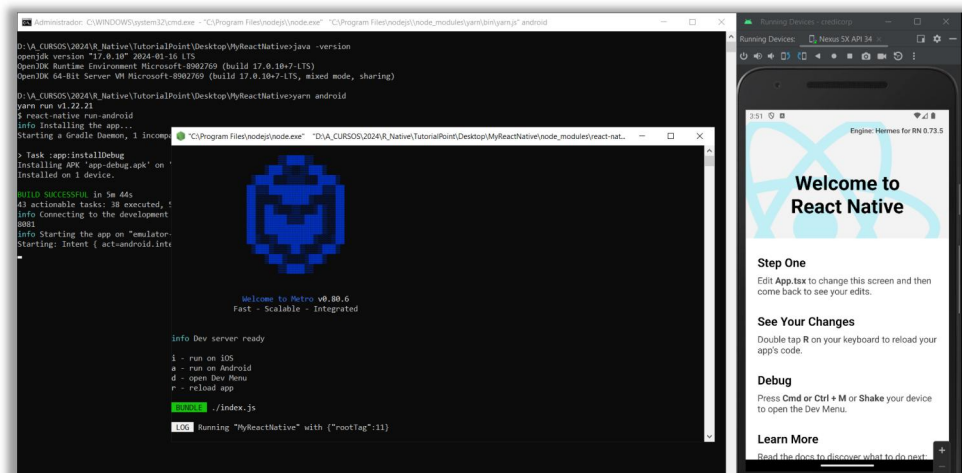
- Abrir otro CMD y probar de nuevo

```
Administrador: Símbolo del sistema

D:\A_CURSOS\2024\R_Native\TutorialPoint\Desktop\MyReactNative>java -version
openjdk version "17.0.10" 2024-01-16 LTS
OpenJDK Runtime Environment Microsoft-8902769 (build 17.0.10+7-LTS)
OpenJDK 64-Bit Server VM Microsoft-8902769 (build 17.0.10+7-LTS, mixed mode, sharing)

D:\A_CURSOS\2024\R_Native\TutorialPoint\Desktop\MyReactNative>
```

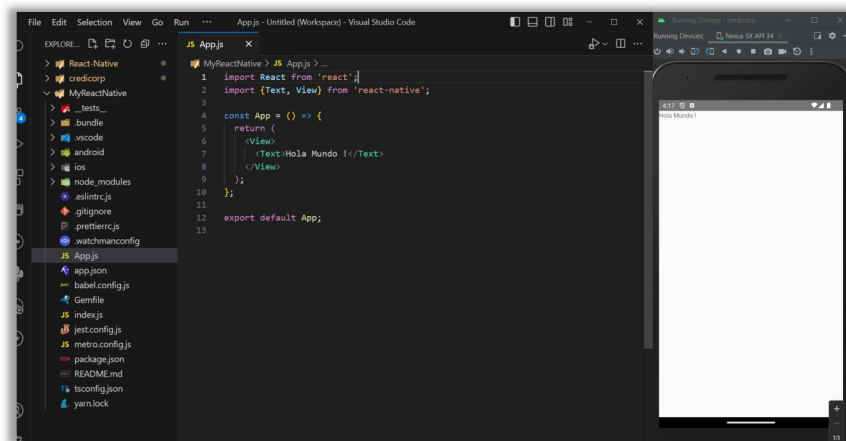
yarn android



- Puede requerir estos comandos
- ✓ ..\TutorialPoint\Desktop\MyReactNative\> npm install
- ✓ ..\TutorialPoint\Desktop\MyReactNative\> yarn install
- ✓ ..\TutorialPoint\Desktop\MyReactNative\android > gradlew clean (limpiar)

- ✓ `..\TutorialPoint\Desktop\MyReactNative\> npm run android`
- ✓ `..\TutorialPoint\Desktop\MyReactNative\> yarn android`
- ✓ `..\TutorialPoint\Desktop\MyReactNative\>npx react-native run-android`
- ✓ `..\TutorialPoint\Desktop\MyReactNative\>npx react-native start --reset-cache (Limpiar cache:)`

React Native – Hola Mundo



React Native – State

- Los datos dentro de React Componentes se administran por estado y accesorios.
- El **estado es mutable** mientras que los **accesorios son inmutables**. Esto significa que el estado se puede actualizar en el futuro, mientras que los accesorios no se pueden actualizar.

Usando el estado (state)

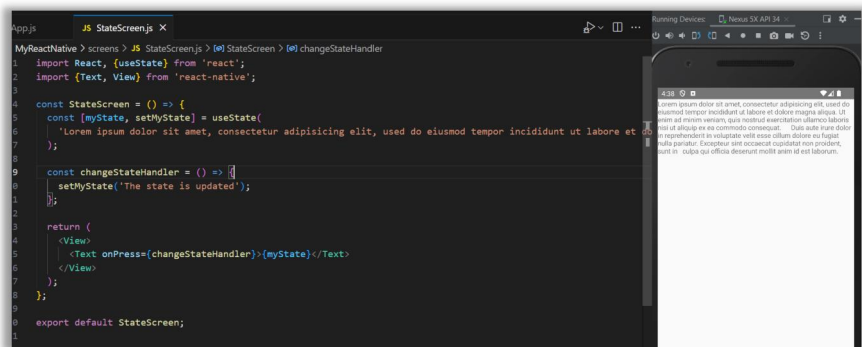
```
import React, {useState} from 'react';
import {Text, View} from 'react-native';

const StateScreen = () => {
  const [myState, setMyState] = useState(
    'Lorem ipsum dolor sit amet, consectetur adipisicing elit, used do eiusmod tempor
    incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud
    exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.      Duis aute irure
    dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.
    Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit
    anim id est laborum.',
  );

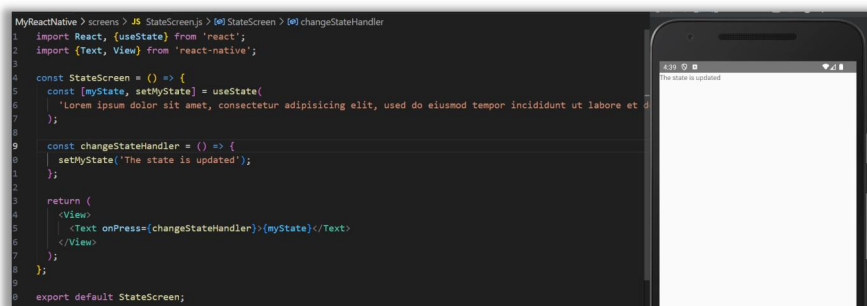
  const changeStateHandler = () => {
    setMyState('The state is updated');
  };

  return (
    <View>
      <Text>React Native - Estado</Text>
      <Text onPress={changeStateHandler}>{myState}</Text>
    </View>
  );
};

export default StateScreen;
```



- Click sobre el texto



React Native - Props

- El **estado es mutable** mientras que los **accesorios son inmutables**. Esto significa que el estado se puede actualizar en el futuro, mientras que los accesorios no se pueden actualizar.
- Los componentes de presentación deben obtener todos los datos **pasando accesorios**. Sólo los **componentes del contenedor** deben tener **estado**.

Componente del contenedor

El componente contenedor solo se utiliza para manejar el estado. Toda la funcionalidad relacionada con la vista (estilo, etc.) se manejará en el componente de presentación.

Ahora actualizaremos nuestro componente contenedor. Este componente manejará el estado y pasará los accesorios al componente de presentación.

```
import React from 'react';
//import StateScreen from './screens/StateScreen';
import PropsScreenContainer from './screens/PropsScreen/PropsScreenContainer';
import {StyleSheet, View} from 'react-native';

const App = () => {
  //return <StateScreen />;
  return (
    <View style={styles.container}>
      <PropsScreenContainer />
    </View>
  );
};

export default App;

const styles = StyleSheet.create({
  container: {
    flex: 1,
    flexDirection: 'column',
    margin: 15,
    backgroundColor: '#c7f69e',
  },
});
```

```
import React, {useState} from 'react';
import PropsScreenView from './PropsScreenView';

/**
 * Componente Contenedor. Este componente maneja el estado y pasa los accesorios
 * al componente de presentación
 */
```

```

const initText =
  'No hay nadie que ame el dolor mismo, que lo busque y lo quiera tener, simplemente porque es el dolor...';

const PropsScreenContainer = () => {
  const [myState, setMyState] = useState(initText);

  const updateStateHandler = newText => {
    setMyState(
      newText === 'Estado actualizado..' ? initText : 'Estado actualizado..',
    );
  };

  return <PropsScreenView value={myState} onUpdate={updateStateHandler} />;
};

export default PropsScreenContainer;

```

Componente de presentación

Los componentes de presentación deben usarse únicamente para presentar la vista a los usuarios. Estos componentes no tienen estado. Reciben todos los datos y funciones como accesorios.

La funcionalidad relacionada con la vista (estilo, etc.) se manejará en el componente de presentación.

- La mejor práctica es utilizar tantos componentes de presentación como sea posible.

En la parte inferior del archivo, crearemos nuestra hoja de estilos y la asignaremos a la constante de estilos. Tenga en cuenta que nuestros estilos están en camelCase y no usamos px ni % para diseñar. Para aplicar estilos a nuestro texto, necesitamos agregar la propiedad `style = {styles.text}` al elemento `Text`.

```

import {View, Text, StyleSheet} from 'react-native';
import React from 'react';

/**
 * Componente Contenedor. Este componente maneja el estado y pasa los accesorios
 * al componente de presentación
 */
const PropsScreen = ({value, onUpdate}) => {
  return (
    <View style={styles.container}>
      <View style={styles.titleView}>
        <Text style={styles.title}>React Native - Props</Text>
      </View>
      <View style={styles.textView}>
        <Text
          style={styles.text}
          onPress={() => {

```

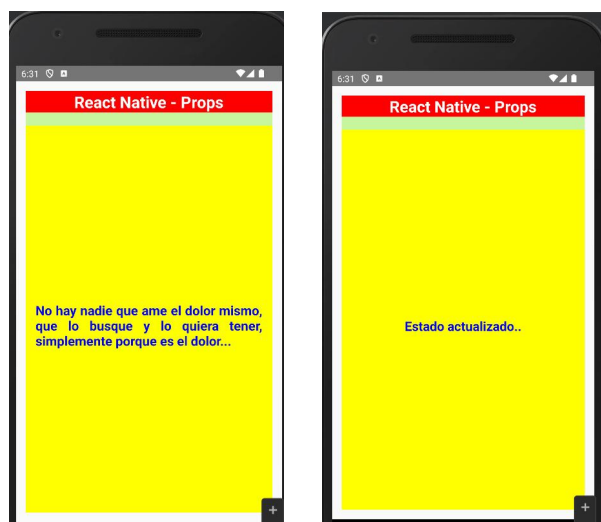
```

        onUpdate(value);
      }}>
      {value}
    </Text>
  </View>
</View>
);
});

export default PropsScreen;

const styles = StyleSheet.create({
  container: {
    flex: 1,
    flexDirection: 'column',
  },
  titleView: {backgroundColor: 'red', alignItems: 'center'},
  title: {color: 'white', fontSize: 25, fontWeight: 'bold'},
  textView: {
    flex: 1,
    marginTop: 20,
    backgroundColor: 'yellow',
    alignItems: 'center',
    justifyContent: 'center',
  },
  text: {
    marginTop: 20,
    textAlign: 'justify',
    color: 'blue',
    fontWeight: 'bold',
    fontSize: 20,
  },
});

```



React Native - Flexbox

Para adaptarse a diferentes tamaños de pantalla, React Native ofrece compatibilidad con **Flexbox**.

Disposición

Para lograr el diseño deseado, flexbox ofrece tres propiedades principales:

- `flexDirection`
- `justifyContent`
- `alignItems`.

La siguiente tabla muestra las opciones posibles.

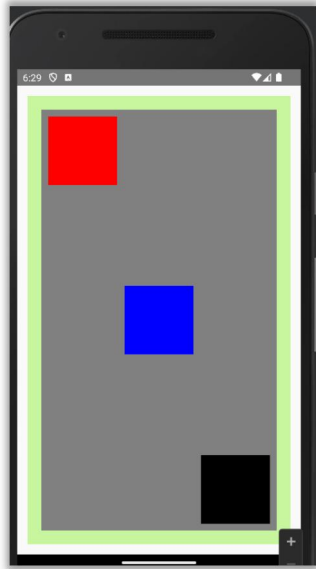
Propiedad	Valores	Descripción
flexDirection	'column', 'row'	Se utiliza para especificar si los elementos se alinearán vertical u horizontalmente.
justifyContent	'center', 'flex-start', 'flex-end', 'space-around', 'space-between'	Se utiliza para determinar cómo se deben distribuir los elementos dentro del contenedor.
alignItems	'center', 'flex-start', 'flex-end', 'stretched'	Se utiliza para determinar cómo se deben distribuir los elementos dentro del contenedor a lo largo del eje secundario (opuesto a <code>flexDirection</code>)

```
...
const App = () => {
  return (
    <View style={styles.container}>
      {/* return <StateScreen />; */}
      {/* <PropsScreenContainer /> */}
      <FlexBoxScreen />
    </View>
  );
};
...
```

```
import React from 'react';
import {View, StyleSheet} from 'react-native';

const FlexBoxScreen = () => {
  return (
    <View style={styles.container}>
      <View style={styles.redbox} />
      <View style={styles.bluebox} />
      <View style={styles.blackbox} />
    </View>
  );
};
```

```
    );  
  });  
  
export default FlexBoxScreen;  
  
const styles = StyleSheet.create({  
  container: {  
    flex: 1,  
    flexDirection: 'row',  
    backgroundColor: 'grey',  
    margin: 20,  
    justifyContent: 'space-between',  
    alignItems: 'flex-start',  
  },  
  redbox: {  
    width: 100,  
    height: 100,  
    backgroundColor: 'red',  
    marginTop: 10,  
    marginLeft: 10,  
  },  
  bluebox: {  
    width: 100,  
    height: 100,  
    backgroundColor: 'blue',  
    alignSelf: 'center',  
  },  
  blackbox: {  
    width: 100,  
    height: 100,  
    backgroundColor: 'black',  
    alignSelf: 'flex-end',  
    marginRight: 10,  
    marginBottom: 10,  
  },  
});
```

React Native - ListView

Crear una lista en React Native. Importaremos la Lista en nuestro componente Inicio y la mostraremos en la pantalla.

```
const App = () => {  
  return (  
    <View style={styles.container}>  
      {/* return <StateScreen />; */}  
      {/* <PropsScreenContainer /> */}  
      {/* <FlexBoxScreen /> */}  
      <ListViewScreen />  
    </View>  
  );  
};
```

Para crear una lista, usaremos el método **map()** . Esto iterará sobre una serie de elementos y representará cada uno de ellos.

```
import {View, Text, StyleSheet, TouchableOpacity} from 'react-native';  
import React, {useState} from 'react';  
  
const data = [  
  {  
    id: 0,  
    name: 'Carlos',  
  },  
  {  
    id: 1,  
    name: 'Daniel',  
  },  
  {  
    id: 2,
```

```

      name: 'Andrés',
    },
    {
      id: 3,
      name: 'Camila',
    },
    {
      id: 4,
      name: 'William',
    },
  ],
];

const ListViewScreen = () => {
  const [myList, setMyList] = useState({names: data});

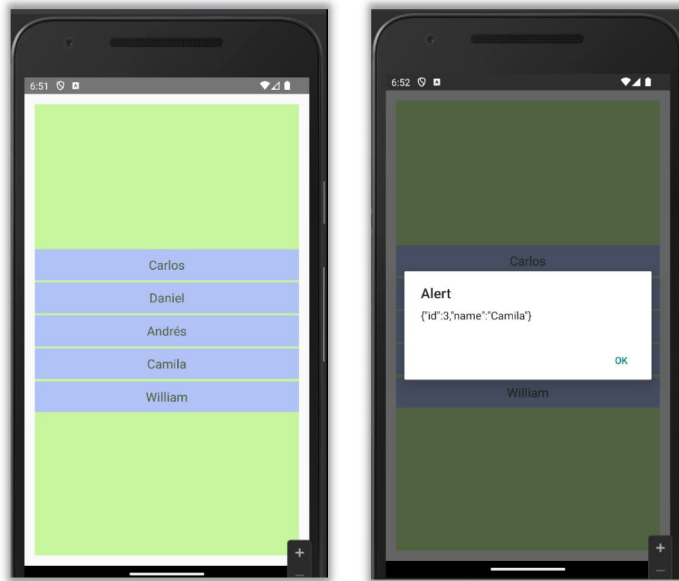
  const showItem = item => {
    alert(JSON.stringify(item));
  };

  return (
    <View style={styles.container}>
      {myList.names.map((item, index) => (
        <TouchableOpacity
          key={item.id}
          style={styles.item}
          onPress={() => showItem(item)}>
            <Text style={styles.text}>{item.name}</Text>
          </TouchableOpacity>
        )))}
    </View>
  );
};

export default ListViewScreen;

const styles = StyleSheet.create({
  container: {flex: 1, justifyContent: 'center'},
  item: {
    padding: 10,
    marginTop: 3,
    backgroundColor: '#b1c2f7',
    alignItems: 'center',
  },
  text: {color: '#4f603c', fontSize: 18},
});

```



React Native - Text Input

En este capítulo, le mostraremos cómo trabajar con elementos **TextInput** en React Native.

El componente Inicio importará y renderizará las entradas.

Definiremos el estado inicial.

Después de definir el estado inicial, crearemos las funciones **handleEmail** y **handlePassword**. Estas funciones se utilizan para actualizar el estado.

A través de un botón la función **login()** simplemente alertará el valor actual del estado.

También agregaremos algunas otras propiedades a las entradas de texto para deshabilitar el uso automático de mayúsculas, eliminar el borde inferior en dispositivos Android y establecer un marcador de posición.

Cambiar JDK de la versión 17 a la 11 en React Native:

> Android Gradle plugin requires Java 17 to run. You are currently using Java 11.

Your current JDK is located in C:\Program Files\Java\jdk-11.0.17

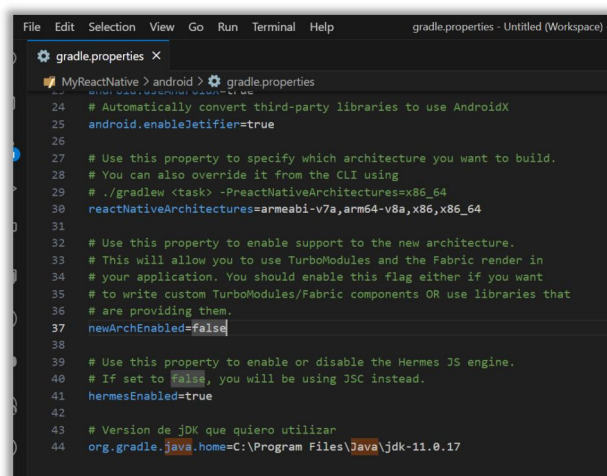
You can try some of the following options:

- changing the IDE settings.
- changing the JAVA_HOME environment variable.
- changing `org.gradle.java.home` in `gradle.properties`.

* Try:

- > Run with --stacktrace option to get the stack trace.
- > Run with --info or --debug option to get more log output.
- > Run with --scan to get full insights.
- > Get more help at <https://help.gradle.org>.

Cambio mi archivo de gradle.properties y recompilo el proyecto



```
File Edit Selection View Go Run Terminal Help gradle.properties - Untitled (Workspace)
gradle.properties
MyReactNative > android > gradle.properties
24 # Automatically convert third-party libraries to use AndroidX
25 android.enableJetifier=true
26
27 # Use this property to specify which architecture you want to build.
28 # You can also override it from the CLI using
29 # ./gradlew <task> -PreactNativeArchitectures=x86_64
30 reactNativeArchitectures=armeabi-v7a,arm64-v8a,x86,x86_64
31
32 # Use this property to enable support to the new architecture.
33 # This will allow you to use TurboModules and the Fabric render in
34 # your application. You should enable this flag either if you want
35 # to write custom TurboModules/Fabric components OR use libraries that
36 # are providing them.
37 newArchEnabled=false
38
39 # Use this property to enable or disable the Hermes JS engine.
40 # If set to false, you will be using JSC instead.
41 hermesEnabled=true
42
43 # Version de jdk que quiero utilizar
44 org.gradle.java.home=C:\Program Files\Java\jdk-11.0.17
```

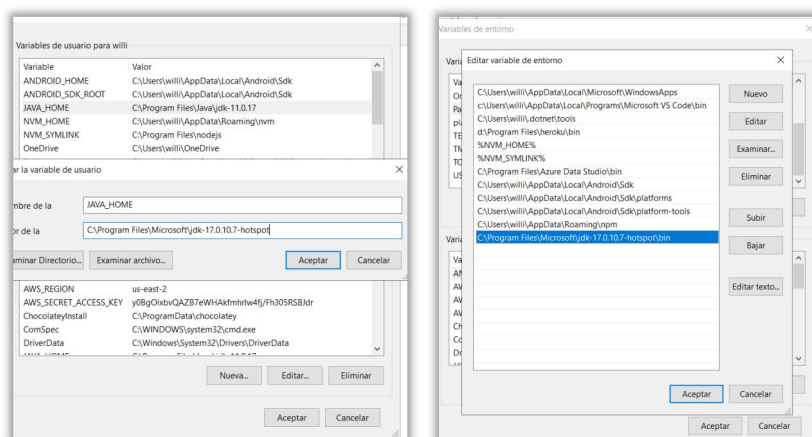
No me funciona.

Voy a cambiar las variables de ambiente.

JDK

11: C:\Program Files\Java\jdk-11.0.17

17: C:\Program Files\Microsoft\jdk-17.0.10.7-hotspot



Nota: Para usuario y sistema

- Si se quiere se limpia la Android y se compila todo de nuevo, sino solo se levanta la APP.
- ✓ `..\TutorialPoint\Desktop\MyReactNative\android > gradlew clean (limpiar)`
- ✓ `..\TutorialPoint\Desktop\MyReactNative\> npm run android`
- ✓ `..\TutorialPoint\Desktop\MyReactNative\> yarn android`