

DESARROLLO DE APLICACIONES PARA DISPOSITIVOS MÓVILES

Universidad de Sonsonate

Diseño y Composición de Interfaces con Componentes Básicos y Flexbox



Objetivos

- Ejecutar primera aplicación “Hola Mundo” en un emulador
 - Comprender la estructura y uso de los componentes básicos de React Native para construir interfaces móviles iniciales.
 - Aplicar principios de diseño con Flexbox y lógica básica en JavaScript para organizar visualmente los elementos en pantalla.
-

Instalación y configuración de Android Studio

Objetivos específicos:

- Comprender **qué es Android Studio** y por qué es útil para probar aplicaciones móviles.
- Instalar **Android Studio correctamente** en las computadoras.
- Configurar un **emulador Android** compatible con Expo Go.
- Aprender a **vincular el emulador con Expo** para correr aplicaciones sin un celular físico.

¿Qué es Android Studio y por qué lo usamos?

- Android Studio es el entorno oficial de desarrollo para apps Android.
- Nos permite **crear emuladores**, que son simulaciones de teléfonos.
- Aunque no programemos en Java/Kotlin, Expo puede usar Android Studio para **correr apps sin necesitar un dispositivo físico**.

Instalación paso a paso de Android Studio

Requerimientos de hardware: Asegurarse de tener mínimo **8 GB de RAM** y **20 GB de espacio libre**.

Paso 1: Descargar

- Ir a <https://developer.android.com/studio>
- Elegir el sistema operativo (Windows, Mac, Linux)
- Descargar y ejecutar el instalador.

Instalación paso a paso de Android Studio

Paso 2: Instalación guiada

- Abrir el instalador y seguir los pasos por defecto.
- Instalar los componentes por defecto: Android Studio, SDK, Emulator, etc.
- Dejar las configuraciones por defecto en la mayoría de pasos.
- Instalar el componente extra llamado: **Android Virtual Device (AVD)**.
- Al finalizar, Android Studio se abrirá con el "Welcome Screen".

Crear y configurar un emulador

Objetivo: tener un emulador Android con **Play Store** (requisito para usar Expo Go).

Paso 1: Abrir el AVD Manager

- En Android Studio, clic en "More Actions" > **Virtual Device Manager**
- O desde la pantalla principal: *Tools* > *Device Manager*

Crear y configurar un emulador

Paso 2: Crear un nuevo dispositivo

En Device Manager

- Clic en **Create Virtual Device**
- Elegir: Pixel 7 o similar (buen rendimiento)
- Seleccionar una imagen del sistema con Google Play Store (x86_64) — idealmente API 35 o API 36.
- Esperar descarga e instalación.

Paso 3: Configurar

En Device

- Nombre: “ExpoEmulator”.

En Additional Settings

- RAM: puede quedarse por defecto (2–3 GB).

Finalizar y lanzar el emulador.

Probar una app Expo en el emulador

Paso 1: Abrir un proyecto creado con npx : **create-expo-app@latest miApp**

Paso 2: Iniciar con

- **npx expo start**

Nota: Recordar modificar el punto de entrada en package.json:

- **"main": "node_modules/expo/AppEntry.js"**
 - Crear (si no existe) en la raíz del proyecto el archivo principal **App.js**
-

Código de ejemplo

JS App.js U X

JS App.js > App

```
1  import { Text, View } from 'react-native'
2
3  export default function App() {
4    return(
5      <View style={{marginTop: 300, marginLeft: 70}}>
6        <Text style={{fontSize: 50}}>Hola, mundo!</Text>
7      </View>
8    )
9  }
```

Probar una app Expo en el emulador

Paso 3: En la terminal, presionar la tecla **a**
Esto abre **Expo Go en el emulador**.

Si no funciona, revisar que: El emulador esté encendido.

Expo esté reconociendo el emulador (mensaje: Opening on Android emulator...).

Agregar ADB al PATH en Windows para automatizarlo con Expo

¿Qué es ADB?

ADB (Android Debug Bridge) es una herramienta que permite comunicarse con dispositivos Android o emuladores desde la terminal.

Expo usa ADB para encontrar y lanzar el emulador automáticamente.

Paso 1: Ubicar la ruta del SDK de Android

- Abrir **Android Studio**
- Ir a:

File > Settings > Appearance & Behavior > System Settings > Android SDK

Se verá una ruta como:

C:\Users\TuUsuario\AppData\Local\Android\Sdk

Dentro de esa carpeta, abrir **platform-tools**

Allí está el archivo:

adb.exe

Paso 2: Copiar la ruta completa de platform-tools

Por ejemplo:

```
C:\Users\TuUsuario\AppData\Local\Android\Sdk\platform-tools
```

Paso 3: Agregar la ruta a las variables de entorno

Presionar Windows + S y escribir:

- **variables de entorno**

Abrir:

- **Editar las variables de entorno del sistema**

En la ventana, hacer clic en

- **“Variables de entorno...”**

En la sección “Variables del sistema”, buscar y seleccionar la variable llamada:

- **Path**

Hacer clic en "**Editar...**"

Luego en "**Nuevo**" y pegar la ruta:

C:\Users\TuUsuario\AppData\Local\Android\Sdk\platform-tools

Aceptar todo: “**Aceptar**” > “**Aceptar**” > “**Aceptar**”.

Paso 4: Agregar emulador al PATH

Volver a las variables de entorno (**Path**)

Agregar esta ruta también:

- C:\Users\TuUsuario\AppData\Local\Android\Sdk\emulator

Guardar los cambios y cerrar todas las terminales abiertas.

Paso 5: Verificar que todo funcione

Abrir una nueva terminal (CMD, PowerShell o terminal de VS Code).

Escribir: **adb devices**

Si sale algo como “List of devices attached”, ADB está listo.

Si devuelve el nombre del emulador (Pixel7..), también está todo bien.

Paso 6: Probar en Expo

Abrir proyecto creado con Expo

En la terminal ejecutar:

- **npx expo start --Android**
- O solo **npx expo start** y elegir la opción **a**

Proceso exitoso:

- Expo lanzará automáticamente el emulador.
- Se abrirá **Expo Go** dentro del emulador.
- Y cargará la app sin tocar Android Studio.

Estructura de un Proyecto en React Native (con Expo)

Objetivos:









- Entender **la estructura base de un proyecto React Native creado con Expo.**
- Identificar el rol de cada carpeta y archivo principal.
- Diferenciar qué archivos son editables por el desarrollador y cuáles no.
- Preparar el terreno para **crear componentes y organizar la app profesionalmente.**

PARTE 1: ¿Qué pasa al hacer **npx create-expo-app**?

- Obtenemos un proyecto con una Estructura básica típica similar a la siguiente:

```
mi-proyecto/  
|  
├─ assets/  
├─ node_modules/  
├─ .gitignore  
├─ App.js  
├─ app.json  
├─ babel.config.js  
├─ package.json  
└─ README.md
```




PARTE 2: Explicación de cada parte

Elemento	¿Qué es y para qué sirve?
App.js	 Punto de entrada de la app. Aquí comienza todo. Contiene el componente raíz.
assets/	 Carpeta donde colocamos imágenes, fuentes, sonidos , etc.
node_modules/	 Aquí se instalan automáticamente todas las dependencias (librerías externas).
package.json	 Define el nombre de la app, scripts y qué paquetes usa. Es el corazón del proyecto .
app.json	 Configuración de Expo: nombre de la app, ícono, splash screen, permisos, etc.
.gitignore	 Archivos que no se deben subir a Git (como node_modules, claves, etc).
babel.config.js	 Configuración de Babel , el compilador de JavaScript moderno.
README.md	 Archivo de documentación inicial del proyecto.

PARTE 3: Estructura profesional sugerida

```
mi-proyecto/  
|  
├─ assets/  
├─ src/  
|   ├─ components/  
|   ├─ screens/  
|   ├─ navigation/  
|   ├─ context/  
|   └─ utils/  
├─ App.js  
├─ app.json  
├─ package.json  
...
```

Explicación de la nueva estructura

Carpeta / Archivo	¿Para qué sirve?
components/	 Componentes reutilizables (botones, tarjetas, inputs...).
screens/	 Cada pantalla principal de la app (Home, Login, Perfil...).
navigation/	 Configuración de la navegación con react-navigation.
context/	 Estado global de la app usando React Context o Redux.
utils/	 Funciones reutilizables, validaciones, formateos, etc.

Primeros Pasos Visuales: Componentes Básicos, Flexbox y un vistazo a la lógica con JavaScript

Objetivos específicos

- Comprender la estructura y uso de los componentes básicos de React Native para construir interfaces móviles iniciales.
- Aplicar principios de diseño con Flexbox y lógica básica en JavaScript para organizar visualmente los elementos en pantalla.

Contenidos a Desarrollar

- **Componentes básicos de React Native:**

View, Text, Image, Button, TouchableOpacity

- **Introducción a Flexbox:**

flex, flexDirection, justifyContent, alignItems, alignContent

- **JavaScript básico embebido:**

- Declaración de variables (const, let)
- Insertar variables en JSX: {nombre}

Componentes Básicos de React Native

Componente	¿Qué es?	¿Para qué se usa?
<View>	Contenedor general de otros componentes.	Para agrupar y organizar elementos en la pantalla.
<Text>	Muestra texto en pantalla.	Para títulos, párrafos, etiquetas, etc.
<Image>	Muestra una imagen (local o desde internet).	Para íconos, portadas, fotos de perfil, etc.
<Button>	Botón básico con acción (nativo del sistema operativo).	Para ejecutar acciones al hacer clic o tap.

Flexbox en React Native

Propiedad

¿Para qué sirve?

Tipo de valor que acepta

flexDirection

Decide si los elementos van **de arriba a abajo** (column) o **de lado a lado** (row).

"column" – "row"

justifyContent

Alinea los elementos **en la dirección principal** (vertical u horizontal).

"flex-start" – "center" – "space-between" – etc.

alignItems

Alinea los elementos **en la otra dirección** (por ejemplo, **izquierda/derecha** si están en columna).

"flex-start" – "center" – "stretch" – etc.

Flex

Hace que un elemento **ocupe más o menos espacio** relativo a otros.

Número (1, 2, etc.)

Ejemplo visual



Ejemplo Inicial

```
JS App.js U X
JS App.js > App

1  import { Button, Image, Text, View } from 'react-native';
2
3  export default function App() {
4    return (
5      <View>
6        <Image
7          source={{ uri: 'https://cdn-icons-png.flaticon.com/512/9187/9187604.png' }}
8          style={{ width: 100, height: 100 }}
9        />
10       <Text style={{ fontSize: 50 }}>
11         juanPer05
12       </Text>
13       <Button title="Editar" onPress={() => alert("Editando...")} />
14     </View>
15   )
16 }
```

Gracias por su atención