

# DESARROLLO DE APLICACIONES PARA DISPOSITIVOS MÓVILES

Universidad de Sonsonate

## **Instalación y Configuración del Entorno de Desarrollo con React Native**



# Objetivos

- Comprender el propósito de cada herramienta en la cadena de desarrollo móvil con React Native.
  - Configurar el entorno básico de trabajo por fases.
  - Fomentar la lectura técnica y la autonomía guiada.
  - Ejecutar un primer proyecto Expo del tipo “Hola Mundo”
-

# FASES DEL ENTORNO DESARROLLO

## **FASE 1: Instalación de Herramientas básicas**

- Visual Studio Code
- Node.JS
- Expo Go

# FASES DEL ENTORNO DE DESARROLLO

## **FASE 2: Primer Proyecto “Hola Mundo”**

- Crear primer proyecto
- Explorar herramientas
- Inicio del desarrollo

## **FASE 3: Instalación de Android Studio para uso de Emuladores**

- Descarga Android Studio
- Configurar y preparar un Emulador

# Visual Studio Code

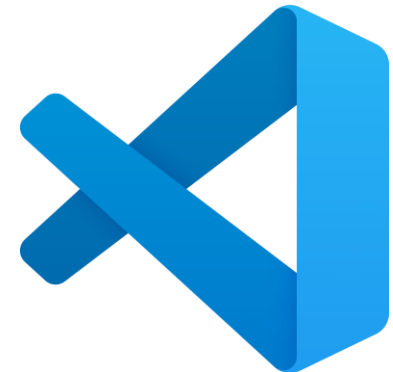
- **Qué es:** Editor de código multiplataforma, liviano y extensible.
- **Razones de uso:**
  - Integración con Git, terminal integrada, extensiones como React Native Tools.
  - Buen soporte para JavaScript, TypeScript y JSX.



# VS-Code para React Native

## Herramientas esenciales que lo hacen ideal:

- **IntelliSense:** Sugiere código automáticamente mientras escribimos.
- **ESLint:** Es un verificador de errores en el código JS.
- **Prettier:** Formatea el código automáticamente para que se vea ordenado.
- **React Native Tools** Extensión oficial que permite:
  - Ejecutar la app desde VS Code.
  - Ver errores y logs en el mismo editor.
  - Conectarse al emulador o teléfono.
  - Depurar tu código paso a paso.



# Iniciando proceso de Instalación de VS-Code

Visitar Sitio Web Oficial:

- <https://code.visualstudio.com/>

Leer documentación y proceder la instalación...



# Node.JS



**Qué es:** Es el motor que permite ejecutar herramientas modernas de desarrollo JavaScript **desde la terminal**. React Native (especialmente con Expo) depende de Node.js para instalar paquetes, compilar código y correr el servidor de desarrollo.

## **En resumen:**

- Permite usar npm o npx (administradores de paquetes)
- Ejecuta el **Metro Bundler** (servidor que sirve y compila la app)
- Hace posible usar herramientas como Expo CLI



# Iniciando proceso de Instalación de Node.JS

Visitar Sitio Web Oficial:

- <https://nodejs.org/>

Leer documentación y proceder la instalación...



Recomendaciones

- Instalar la versión pre compilada de Node.js® para Windows
- Elegir la opción de **Windows Installer**

# Proceso de Instalación de Node.JS

Versión descargada: **LTS** (Long Term Support) — es más estable para desarrollo.

## Al instalar Node.js se incluye:

- node (el motor)
- npm (Node Package Manager)
- npx (Node Package eXecute)

## Verificación en terminal:

*node -v*

*npm -v*

*npx -v*

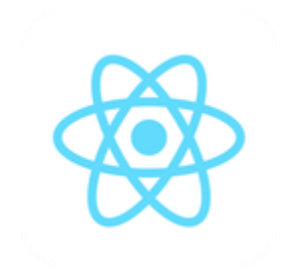


# Investigar

Gestor de paquetes alternativo

- **Yarn**

# Primer Proyecto Con Expo RN



## Objetivo:

Aprender a instalar Expo CLI (tanto global como por proyecto), y correr una app de React Native con Expo en dispositivos físicos o emuladores.

## Prerrequisitos

- Antes de empezar asegurarse de tener:
- VS Code
- Node.js
- Terminal o consola (PowerShell, Terminal de VSCode, etc.)
- Cuenta en Expo.dev (opcional, pero útil para sincronizar proyectos)

# Parte 1: ¿Qué es Expo?

- Expo es una **plataforma y conjunto de herramientas** que facilita el desarrollo de apps con React Native. Permite correr una app sin necesidad de configurar Android Studio ni Xcode al inicio.
- **Tiene dos formas principales de uso**

Forma	¿Qué hace?	¿Cómo se instala?
<b>Global</b>	Instala Expo CLI en todo el sistema	<i><code>npm install -g expo-cli</code></i>
<b>Por proyecto (recomendada)</b>	Usa Expo sin instalarlo globalmente	<i><code>npx create-expo-app</code></i>

# Parte 2: Forma Recomendada – Uso Local (por proyecto)

Paso 1: Crear un nuevo proyecto con npx

- **npx create-expo-app@latest MiPrimeraApp**

Paso 2: Navegar al proyecto

- **cd MiPrimeraApp**

- Paso 3: Ejecutar el proyecto

- **npx expo start**

# Ajustes importantes

Cambio en package.json

De: "main": "**expo-router/entry**"

A: "main": "**node\_modules/expo/AppEntry.js**"

¿Por qué?

- Usamos navegación manual (react-navigation)
- **Aprendemos desde la base**
- Evitamos confusión con rutas automáticas
- No veremos web en este curso

Esto permite **iniciar desde App.js**, como antes.

## ¿Cómo visualizar la app?

### Método



Expo Go App



Emulador  
Android



Emulador iOS

### Requisitos

Android/iOS

Android  
Studio

Xcode

### Notas

Escanea el QR con la app  
Expo Go

Requiere configuración  
previa

Solo en macOS



# Parte 3: Forma Alternativa – Uso Global de Expo CLI

- Nota: No es obligatoria, pero puede ser útil si se usa mucho Expo.

Paso 1: Instalar globalmente

- **npm install -g expo-cli**

Paso 2: Crear proyecto

- **expo init MiAppGlobal**

Paso 3: Ejecutar

- **cd MiAppGlobal expo start**

## Comparación final

### Criterio

### Expo Global

### Expo por proyecto (npx)

Instalación previa

Necesaria

No necesaria

Versión de Expo

Puede quedar  
desactualizada

Siempre usa la  
última

Recomendado  
para clases



Sí

Ideal para  
principiantes



Sí

# Android Studio

## Objetivos de la fase:

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

# PARTE 1 ¿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**.

# PARTE 2: 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".

# PARTE 3: 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" > **AVD Manager**
- O desde la pantalla principal: *Tools* > *Device Manager*

# Crear y configurar un emulador

## Paso 2: Crear un nuevo dispositivo

- Clic en **More Action**
- Clic en **Virtual device Manager**
- Clic en **Create Virtual Device**
- Elegir: Pixel 7 o similar (buen rendimiento)
- Seleccionar una imagen del sistema con Play Store (x86\_64) — idealmente API 33 o API 34.
- Esperar descarga e instalación.

## Paso 3: Configurar

- Nombre: “ExpoEmulator”.
- RAM: puede quedarse por defecto (2–3 GB).
- Finalizar y lanzar el emulador.





# PARTE 4: Probar una app Expo en el emulador

Paso 1: Abrir un proyecto creado con npx

- **create-expo-app**

Paso 2: Iniciar con

- **npx expo start**
-

# 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...).

# Pasos 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

Si no, buscar la ruta en el explorador de archivos y entro de la carpeta **Sdk**, 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

**Nota:** Emulador debe estar cerrado para comprobar que se abra automáticamente

En la terminal ejecutar:

- **npx expo start --Android**

Proceso exitoso:

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

Gracias por su atención