GUÍA PASO A PASO

Migrar a la arquitectura nueva de React Native

La checklist de migración desde React Native 0.81 o el SDK 54 de Expo hasta la versión actual, dónde activar la arquitectura nueva, un script en Node que separa tus librerías nativas y una tabla para auditarlas una por una.

PARA	Apps en React Native o Expo que todavía usan la arquitectura vieja o no saben cuál usan
PROBADO CON	Node 22.22.3 · script de auditoría corrido sobre una carpeta de prueba · sin build en dispositivo
ACTUALIZADA	19 de noviembre de 2026
DOCUMENTACIÓN	reactnative.dev/blog/2025/10/08/react-native-0.82 · docs.expo.dev/guides/new-architecture
ARTÍCULO	ecomlabs.dev/g/arquitectura
AUTOR	Luis Morales · @luismoralesdev

Índice

Antes de empezar	Dónde estás y a dónde vas
Paso 01	Sube a la 0.81 o al SDK 54
Paso 02	Activa la arquitectura nueva
Paso 03	Audita tus librerías nativas
Paso 04	Si una librería no es compatible
Paso 05	Sube hasta la versión actual
Al terminar	Checklist

ANTES DE EMPEZAR

Dónde estás y a dónde vas

Desde React Native 0.82 (8 de octubre de 2025) la arquitectura nueva es la única: `newArchEnabled=false` en Android y `RCT_NEW_ARCH_ENABLED=0` en iOS se ignoran. La 0.81 y el SDK 54 de Expo son las últimas versiones que permiten la arquitectura vieja. Hoy, 19 de noviembre de 2026, la versión actual es la 0.88 y el SDK 58 de Expo la incluye. Desde la 0.76 la nueva viene activada por defecto: si nadie la apagó (paso 02), ya la usas.

Si estás en la 0.80 o menos, empieza en el paso 01. Si estás en la 0.81 o el SDK 54 con la vieja, en el paso 02. Si ya usas la nueva, ve al paso 05, y vuelve al paso 03 cuando algo falle.

> **CUIDADO**
> Antes de tocar nada, decide cómo vuelves: guarda el último build que funciona, anota su versión y ten la rama lista. Una app publicada en las tiendas no se revierte con un clic.

PASO 01

Sube a la 0.81 o al SDK 54

Haz este paso sin cambiar de arquitectura. Expo recomienda subir los SDK de uno en uno; si estás en el SDK 52, pasa primero al 53 y después al 54.

```
# Proyecto con Expo: repite por cada SDK hasta el 54
npm install expo@^54.0.0
npx expo install --fix
npx expo-doctor
```

El primero sube `expo`, el segundo alinea las demás dependencias con ese SDK y el tercero revisa problemas comunes: corrígelos antes de seguir. Sin Expo, usa el Upgrade Helper (`react-native-community.github.io/upgrade-helper`) para ver los cambios de archivos entre tu versión y la 0.81, e instala `react-native` y `react` con las versiones que indica. Prueba la app completa en esta versión. Si algo falla aquí, el problema es la versión, no la arquitectura.

PASO 02

Activa la arquitectura nueva

La guía de Expo documenta tres lugares donde se apaga. Búscalos desde la raíz del proyecto:

```
grep -rn "newArchEnabled" app.json android/gradle.properties ios/Podfile.properties.json
grep -rn "RCT_NEW_ARCH_ENABLED" ios/Podfile package.json
```

ARCHIVO	APAGADA	ACTIVADA
`app.json` (SDK 54)	`"newArchEnabled": false`	`"newArchEnabled": true` o sin la línea
`android/gradle.properties`	`newArchEnabled=false`	`newArchEnabled=true`
`ios/Podfile.properties.json`	`"newArchEnabled": "false"`	`"newArchEnabled": "true"`
Scripts o `Podfile`	`RCT_NEW_ARCH_ENABLED=0`	Quita la variable y reinstala los pods

Si usas `app.config.ts`, busca ahí también. Después crea un build de desarrollo nuevo (`npx expo run:android` y `npx expo run:ios`, o tu build en la nube). Expo Go no sirve para probar tus propias librerías nativas.

PASO 03

Audita tus librerías nativas

Las librerías que solo tienen JavaScript no se ven afectadas. Las que traen `ios/`, `android/` o un `.podspec` sí. Este script las separa leyendo tu `node_modules`, sin conexión. Córrelo después de `npm install`:

audit-native-deps.mjs

```
// audit-native-deps.mjs: list the dependencies that ship native code and what they declare.
// Usage: node audit-native-deps.mjs [path/to/app]   (run after npm install)
import { readFileSync, existsSync, readdirSync } from "node:fs";
import { join } from "node:path";

const root = process.argv[2] ?? ".";
const app = JSON.parse(readFileSync(join(root, "package.json"), "utf8"));
const names = Object.keys({ ...app.dependencies, ...app.devDependencies }).sort();

const rows = [];
for (const name of names) {
  const dir = join(root, "node_modules", name);
  if (!existsSync(join(dir, "package.json"))) {
    rows.push([name, "?", "not installed"]);
    continue;
  }
  const pkg = JSON.parse(readFileSync(join(dir, "package.json"), "utf8"));
  const files = readdirSync(dir);
  const native = files.includes("ios") || files.includes("android") || files.some((f) =>
f.endsWith(".podspec"));
  if (!native) continue; // JavaScript only: the architecture does not affect it
  const verdict = pkg.codegenConfig
    ? "codegenConfig: ships New Architecture specs"
    : files.includes("expo-module.config.json")
      ? "Expo module: supports the New Architecture"
      : "CHECK: native code, no codegenConfig";
  rows.push([name, pkg.version, verdict]);
}

for (const [name, version, verdict] of rows) console.log(`${name.padEnd(34)} ${String(version).padEnd(9)}
```

```
${verdict}`);
const check = rows.filter((r) => r[2].startsWith("CHECK") || r[1] === "?").length;
console.log(`\n${rows.length} native or missing, ${check} to check by hand`);
process.exit(check ? 1 : 0);
```

`codegenConfig` significa que trae especificaciones para TurboModules o Fabric; `Expo module`, que es un módulo de Expo, y esos soportan la arquitectura nueva. Pruébalas igual en el build. `CHECK` es código nativo sin especificación: puede correr por el interop layer o fallar, y va a tu tabla de auditoría. El script sale con código 1 si hay algo que revisar, así que puedes ponerlo en CI. Lo corrimos con Node 22 sobre una carpeta de prueba con paquetes inventados; no es un veredicto, es un filtro.

Dos fuentes más para la lista CHECK

En React Native Directory (`reactnative.directory`) cada librería tiene un campo `newArchitecture`, y `npx expo-doctor` compara tus dependencias con esos datos. Si ya revisaste una que el directorio no conoce, exclúyela en `package.json` con `expo.doctor.reactNativeDirectoryCheck.exclude`.

Tabla de auditoría

Lleva una tabla así en tu repositorio, una fila por librería CHECK. Las filas son ilustrativas:

LIBRERÍA	VERSIÓN	DIRECTORIO	PANTALLA PROBADA	RESULTADO	ACCIÓN
old-native-lib	0.9.3	sin dato	Perfil, subir foto	Falla al abrir	Reemplazar
@acme/maps	1.4.0	sí	Mapa de tiendas	Funciona	Actualizar a la última

Añade una fila por librería. Solo se cierra cuando abriste en el build de desarrollo la pantalla que la usa, en iOS y en Android.

PASO 04

Si una librería no es compatible

En este orden, según la guía de Expo:

1. **Actualízala** a su última versión. Muchas ya añadieron soporte.
2. **Aíslala**: crea una rama, quita las librerías que fallan hasta que la app arranque y mide cuánto dependes de cada una.
3. **Reemplázala** por una compatible. Expo sugiere, por ejemplo, `expo-location` en lugar de `react-native-geolocation-service`.
4. **Repórtala** con un issue o un pull request en su repositorio, con un ejemplo mínimo.
5. **Si es tuya, mígrala**: escribe su especificación en TypeScript con el prefijo `Native` (por ejemplo, `NativeLocalStorage.ts`), añade `codegenConfig` a su `package.json` y regenera.

PASO 05

Sube hasta la versión actual

Quedarte en el SDK 54 con la arquitectura vieja es una pausa, no un plan. Con la app estable en la nueva, sube un SDK a la vez (55, 56, 57, 58): repite los comandos del paso 01 con el número nuevo y vuelve a correr el script del paso 03. El SDK 55 trae React Native 0.83 y quita `newArchEnabled` de `app.json`; el SDK 58 trae la 0.88. Sin Expo, usa el Upgrade Helper de versión en versión.

AL TERMINAR

Checklist

- [ ] Hay un build anterior guardado y una forma de volver a él.
- [ ] La app corre estable en React Native 0.81 o en el SDK 54.
- [ ] Ningún archivo deja `newArchEnabled` en `false` ni `RCT_NEW_ARCH_ENABLED=0`.
- [ ] Hay un build de desarrollo nuevo con la arquitectura nueva activada.
- [ ] El script de auditoría corrió y cada fila CHECK está en la tabla.
- [ ] Cada pantalla que usa una librería nativa se abrió en iOS y en Android.
- [ ] `npx expo-doctor` pasa, o cada exclusión tiene motivo.
- [ ] Cada librería incompatible está actualizada, reemplazada, reportada o migrada.
- [ ] La actualización sigue de SDK en SDK hasta la versión actual.