Builds de VeriX

Builds de VeriX (`scripts\build.ps1`)

Cómo generar un build de VeriX con versión y nombre de archivo automáticos, para `dev`, `test` o `prod`.

Qué hace este script

`build.ps1` es un wrapper alrededor de `eas build`. Dispara el build **en la nube de Expo** (los servidores de Expo compilan, no tu máquina — es exactamente lo mismo que correr `eas build -p android --profile test` a mano), espera a que termine, y descarga el `.apk`/`.aab` ya renombrado. No reemplaza nada de tu flujo normal, solo lo automatiza.

> `eas build --local` **NO se usa aquí.** Si en algún lado ves referencias a builds "locales" en commits viejos de este repo, ya no aplica — el flujo actual es 100% en la nube.

Uso

Desde la raíz del repo (`nexus33-app-verix`), en PowerShell:

```
.\scripts\build.ps1 -Env test
.\scripts\build.ps1 -Env dev
.\scripts\build.ps1 -Env prod
```

Por defecto compila Android. Para iOS:

```
.\scripts\build.ps1 -Env test -Platform ios
```

El build en la nube tarda normalmente **15-20+ minutos** — el script se queda esperando, no hace falta hacer nada mientras tanto.

Prerrequisitos

- Sesión iniciada en EAS: `npx eas-cli login` (si nunca lo has hecho en esta máquina).
- Conexión a internet (el build corre en los servidores de Expo, no localmente).
- Git instalado y este repo clonado con al menos un commit (el script usa el hash del commit actual).

No hace falta Android SDK, JDK, ni Xcode instalados — eso lo tiene Expo del lado de sus servidores.

Troubleshooting

`An Expo user account is required to proceed` / **`eas build exited with code 1`** No hay sesión iniciada en EAS en esta máquina (el script corre con `--non-interactive`, así que no te va a pedir login solo — falla directo). Soluciónalo con:

```
npx eas-cli login
```

y confirma con `npx eas-cli whoami`. Después vuelve a correr el script normalmente. Ver también la nota sobre `app.json` más abajo si el build falló después de que la versión ya se bumpeó.

`eas : term not recognized` Estás intentando correr `eas` directo en vez de `npx eas-cli`. El script usa `npx eas-cli` internamente, así que no necesitas tener `eas-cli` instalado global — pero si quieres el comando corto disponible en tu terminal: `npm install -g eas-cli`.

Qué hace el script, paso a paso

1. Lee la versión actual de `app.json` (`expo.version`).
2. Calcula la siguiente versión — ver "Versionado" abajo.
3. Escribe la nueva versión en `app.json` (**en disco, sin commit/push** — eso lo sigues haciendo tú manualmente, como siempre).
4. Obtiene el hash corto del commit actual (`git rev-parse --short HEAD`).
5. Genera el timestamp del build.
6. Mapea `-Env` al perfil real de `eas.json`: `dev`→`development`, `test`→`test`, `prod`→`production`.
7. Corre `eas build -p <platform> --profile <perfil> --wait` (equivalente a hacerlo a mano) y espera a que termine.
8. Cuando termina, toma el link de descarga que devuelve Expo y baja el archivo a `builds\<nombre-generado>`.

Si algo falla en cualquier paso, el script se detiene y muestra el error — no deja archivos de build a medias con nombres incorrectos.

> ⚠ **Ojo con fallos después del paso 3.** El bump de versión en `app.json` ocurre *antes* de disparar el build (paso 3 vs. paso 7). Si el build falla después de eso — por ejemplo, por no tener sesión iniciada en EAS — `app.json` queda con la versión ya incrementada en disco, sin build real asociado. Antes de reintentar, revisa `git diff app.json`: puedes revertirlo (`git checkout -- app.json`) para no "quemar" un número de versión, o simplemente dejarlo y correr el script de nuevo (no pasa nada grave si el contador salta un número).

Formato del nombre de archivo

```
VeriX-{env}-{version}-{yyyyMMddHHmm}+{commitHash}.{apk|aab|ipa}
```

Ejemplo:

```
VeriX-test-1.0.1-202607241005+7bc21f0.apk
```

Parte	Significado	Origen
`env`	`dev`, `test` o `prod`	Parámetro `-Env`
`version`	Versión semántica	`app.json` (`expo.version`), auto-bumpeada por el script
`yyyyMMddHHmm`	Fecha/hora del build	Generado al momento de compilar
`commitHash`	Hash corto del commit actual	`git rev-parse --short HEAD`
Extensión	`.apk`, `.aab`, `.ipa` (o `.tar.gz` en builds de simulador iOS)	Tomada del link real que devuelve Expo, no adivinada

`prod` en Android genera `.aab` (Android App Bundle, lo que exige Google Play para publicar). `dev` / `test` generan `.apk` (instalable directo en un dispositivo). Esto ya estaba definido en `eas.json`, el script no lo cambia.

Versionado (`app.json` ? `expo.version`)

Formato `major.minor.patch`:

- **`major`** — siempre manual. Solo tú lo cambias, para indicar un cambio grande/breaking.

- `minor` y `patch` — semi-automáticos, con un contador **global** compartido entre `dev`, `test` y `prod` (no es un conteo separado por ambiente):
  - Cada build (sin importar el ambiente) suma 1 al `patch`.
  - Al llegar a 20 builds, el `patch` vuelve a 0 y el `minor` sube en 1.

Build #	Versión
1 (manual, punto de partida)	`1.0.0`
2	`1.0.1`
…	…
20	`1.0.19`
21 (rollover)	`1.1.0`

`versionCode` (Android) / `buildNumber` (iOS) — campos aparte

Estos **no** son lo mismo que `expo.version` de arriba. Son 2 campos separados (uno por plataforma) que las tiendas usan para exigir que cada build subido sea estrictamente más nuevo que el anterior:

- `android.versionCode` (entero: 1, 2, 3…)
- `ios.buildNumber` (string: "1", "2", "3"…)

Hoy están configurados para auto-incrementarse **solo en el perfil** `production` (`eas.json` → `"autoIncrement": true`). Cada build de `production` los sube en 1 automáticamente — EAS los escribe en `app.json` solo, sin que el script tenga que hacer nada. `test`/`development` no los tocan (no lo necesitan: no se suben a las tiendas, y reinstalar un `.apk` con el mismo `versionCode` sobre uno viejo funciona bien en Android).

Salida

Los builds quedan en `nexus33-app-verix\builds\` (carpeta ignorada por git, ver `.gitignore` — no se commitean).

Commit del bump de versión

El script deja `app.json` modificado en disco pero **no commitea ni pushea nada**. Cuando quieras dejar ese cambio de versión en git, hazlo tú manualmente como siempre (`git add app.json`, `git commit`, `git push`).

Pendiente / no cubierto todavía por este flujo

- **`eas submit`** — subir el `.aab`/`.ipa` a Play Store / App Store todavía no está automatizado (`eas.json` → `submit.production` está vacío). Hoy la subida a las tiendas se hace manual con el archivo que este script genera. Los pasos completos para instalar/distribuir cada build (pruebas y producción) están documentados aparte: ver `distribucion-android-eas.md` y `distribucion-ios-eas.md`.
- **Credenciales de firma** (keystore Android / certificados iOS) — no verificadas todavía. EAS las pide de forma interactiva la primera vez que corras `eas build --profile production`; no requiere trabajo previo, solo tenerlo en cuenta la primera vez.
- **Fichas de la app en Play Console / App Store Connect** — no es código, hay que crearlas manualmente (screenshots, descripción, política de privacidad, data safety / privacy nutrition label) antes de poder subir el primer build.
- **OTA updates (`expo-updates`)** — decisión consciente de dejarlo pendiente (2026-07-23). Para el primer lanzamiento no aporta lo suficiente frente a la complejidad que agrega, y al ser una app de cumplimiento regulatorio (DOT) hay valor en que cada cambio pase por el review formal de la tienda. Revisar de nuevo en 1-2 meses con datos reales de cadencia de updates/hotfixes. No requiere ningún cambio previo — es 100% aditivo cuando se decida hacerlo.

Revision #1
Created 2026-08-04 12:40:05 UTC by Admin
Updated 2026-08-04 12:40:15 UTC by Admin