Manual de puesta en marcha

Jolt está completo y probado, pero deliberadamente sin publicar: no tiene cuentas de tienda, ni hosting, ni dominio contratados. Este manual es el camino de una sola pasada desde el código que recibís hasta la app publicada en las dos tiendas. No hay nada de código funcional por escribir: todo lo que sigue es configuración. Cada valor que aparece acá está tomado del código real, no es un ejemplo genérico.

1. Qué estás recibiendo

- **App iOS** — Expo SDK 53, React Native 0.79, React 19, TypeScript. Incluye el proyecto nativo Xcode y un módulo nativo propio para las compras con StoreKit.
- **App Android** — la misma base de producto, con las compras vía Google Play Billing.
- **API** — NestJS + MongoDB, con validación de compras contra Apple y Google, webhooks de las dos tiendas, límites por plan aplicados en el servidor y permisos por rol y por vínculo.
- **Landing y política de privacidad** — estáticas, en cuatro idiomas, listas para subir a cualquier hosting.
- **La cuenta de correo operativa del producto** — trainingjolt@gmail.com — se transfiere al comprador junto con la app. Es la que se usa para las cuentas de tienda y la que figura como contacto en la política de privacidad. La garantía de fallas, en cambio, se reclama a la dirección personal del vendedor: ederfornero.ef@gmail.com.
- **Este manual** — lo único que queda entre el código y producción.

2. Cómo está organizado el código

El producto vive en dos ramas que nunca se mergean entre sí:

Rama	Plataforma	Se publica en
`ios`	iPhone, iPad	Apple Developer / App Store Connect
`android`	Teléfonos y tablets Android	Google Play Console

No es una preferencia de estilo: iOS necesita CocoaPods, un proyecto nativo y un módulo propio para StoreKit, y Android no usa nada de eso. Cada rama tiene su propio package.json, y como node_modules no está trackeado, al cambiar de rama en el mismo directorio queda el node_modules de la otra. Lo práctico es clonar el repo dos veces, una por rama.

```
# iOS
git clone <repo> jolt-ios && cd jolt-ios && git checkout ios
npm install              # el .npmrc ya fija legacy-peer-deps
cd ios && pod install && cd ..

# Android
git clone <repo> jolt-android && cd jolt-android && git checkout android
npm install
```

3. Lo que hay que crear o contratar

Nada de esto existe hoy. Los costos son de referencia a la fecha de este manual.

Qué	Dónde	Costo
`Cuenta Apple Developer`	developer.apple.com/programs	USD 99 / año
`Cuenta Google Play Console`	play.google.com/console/signup	USD 25, pago único
`Base de datos`	MongoDB Atlas — cloud.mongodb.com	M0 gratis para probar; M10 ~USD 57/mes en producción
`Hosting de la API`	Render — dashboard.render.com	Starter USD 7/mes
`Storage de imágenes y videos`	Firebase Storage — console.firebase.google.com	Gratis en el plan inicial
`Envío de correos`	Resend — resend.com	Gratis hasta 3.000 mails/mes
`Dominio`	Cualquier registrador	~USD 10-15 / año

El plazo más largo no es técnico: una cuenta personal de Google Play creada después de noviembre de 2023 necesita un closed test con 12 testers durante 14 días corridos antes de poder publicar en producción. Abrí esa cuenta primero y dejá el test corriendo mientras hacés todo lo demás.

4. Paso 1 — Desplegar la API

Creá el cluster en Atlas y el servicio en Render apuntando al repo de la API. Render detecta NestJS; el health check es /health. Después cargá estas variables de entorno, que son exactamente las que el código lee:

Variable	Ejemplo del valor	De dónde sale
`NODE_ENV`	production	Escrito así, exacto: es lo que desactiva los bypass de compra de desarrollo
`PORT`	3000	Render lo inyecta solo; dejalo si no hay conflicto
`MONGO_URI`	mongodb+srv://jolt:PASS@cluster0.ab12c.mongodb.net/GymApp	Atlas → Connect → Drivers
`JWT_SECRET`	64 caracteres hex	Generalo con: openssl rand -hex 32. No reutilices el de desarrollo
`JWT_EXPIRES_IN`	30d	Vida del token de sesión; 30d es el default
`RESEND_API_KEY`	re_8Kd2mVx…	Resend → API Keys
`MAIL_FROM`	Jolt <no-reply@tudominio.com>	Tiene que usar un dominio verificado en Resend
`APPLE_BUNDLE_ID`	com.jolt.app	El bundle de la app iOS (paso 4)
`APPLE_APP_APPLE_ID`	6478123456	App Store Connect → tu app → App Information → Apple ID. Es numérico, no el bundle
`GOOGLE_ANDROID_PACKAGE_NAME`	com.jolt_training.app	Ya definido en el repo; sólo cambialo si cambiás el package
`GOOGLE_SERVICE_ACCOUNT_EMAIL`	jolt-play@tu-proyecto.iam.gserviceaccount.com	Google Cloud → IAM → Service Accounts (paso 5)

Variable	Ejemplo del valor	De dónde sale
`GOOGLE_SERVICE_ACCOUNT_PRIVATE_KEY`	-----BEGIN PRIVATE KEY-----\nMIIEvQ…	Campo private_key del JSON de esa service account. En Render se pega con los \n literales

No definas APPLE_VERIFY_BYPASS ni GOOGLE_PLAY_VERIFY_BYPASS en producción. Son de desarrollo y el servidor tira error si aparecen con NODE_ENV=production.

5. Paso 2 — Firebase Storage

La app sube las fotos de perfil y los medios de los ejercicios directamente a Firebase Storage. Creá un proyecto propio, activá Storage y pasá las credenciales web del proyecto al .env de las dos apps (las claves FIREBASE_* que ya están declaradas).

Revisá las reglas de seguridad del bucket antes de publicar. Un bucket abierto a escritura sin autenticar es cómodo para desarrollar y un problema en producción.

6. Paso 3 — Correo

La API usa Resend para el correo de bienvenida, la recuperación de contraseña y las invitaciones. Creá la cuenta, verificá tu dominio con los registros DNS que te indica el panel, y cargá RESEND_API_KEY y MAIL_FROM.

Sin un dominio verificado, Resend no entrega. Es el único punto donde el dominio es imprescindible del lado del servidor.

7. Paso 4 — Apple

1. Abrí la cuenta en el Apple Developer Program. Si la abrís como empresa te van a pedir número D-U-N-S y demora semanas; como individuo es inmediato.
2. Registrá el bundle identifier. El código ya viene con com.jolt.app, en app.json y en el proyecto nativo. Si querés otro, cambialo en los dos lugares antes de la primera subida: después no se puede cambiar.
3. Creá la app en App Store Connect y anotá su Apple ID numérico para APPLE_APP_APPLE_ID.
4. Creá un Subscription Group y dentro los tres productos, con estos identificadores exactos.
5. Generá una APNs Auth Key (Certificates → Keys) para las notificaciones push y subila con eas credentials.
6. Configurá las App Store Server Notifications V2 con la URL de tu API.

Product ID	Precio	Límite de deportistas
`jolt_starter`	USD 19 / mes	10
`jolt_pro`	USD 39 / mes	40
`jolt_elite`	USD 79 / mes	ilimitado

```
App Store Connect → tu app → App Information → App Store Server Notifications
Production Server URL (versión 2):
https://TU-API/api/payments/apple/server-notifications
```

8. Paso 5 — Google

1. Abrí la cuenta en Play Console y arrancá el closed test cuanto antes: son 12 testers y 14 días corridos.
2. Creá la app con el package com.jolt_training.app, o cambialo en app.json antes del primer build.
3. Creá las tres suscripciones en Monetize → Products → Subscriptions, con los mismos identificadores y precios de la tabla del paso anterior.
4. Creá una service account en Google Cloud → IAM, descargá su JSON, y en Play Console → Users and permissions invitala con permiso para ver datos financieros. De ahí salen GOOGLE_SERVICE_ACCOUNT_EMAIL y GOOGLE_SERVICE_ACCOUNT_PRIVATE_KEY.
5. Creá un topic de Pub/Sub para las notificaciones de renovación y dale permiso de publicar a la cuenta de sistema de Google Play.
6. Pegá el nombre del topic en Play Console → Monetization setup → Real-time developer notifications.

```
Topic de Pub/Sub, por ejemplo:
projects/tu-proyecto/topics/jolt-rtdn

La suscripción push de ese topic apunta a:
https://TU-API/api/subscriptions/google/server-notifications

Cuenta que necesita permiso de publicar en el topic:
google-play-developer-notifications@system.gserviceaccount.com
```

Sin los webhooks de los pasos 4 y 5, la app cobra la primera vez pero nunca se entera de renovaciones, cancelaciones, vencimientos ni reembolsos. Es el paso que más se saltea y el que más caro sale.

9. Paso 6 — Apuntar las apps al backend y buildear

La URL de la API está en tres lugares por rama, con un placeholder explícito. Reemplazá los tres en cada rama por la URL de tu despliegue:

```
dependencies/api.fetch.ts     → BASE_URL / baseURL
utils/SocketService.ts        → serverUrl
eas.json                      → build.production.env.SERVER_ADDRESS

eas build --platform ios      # desde la rama ios
eas build --platform android  # desde la rama android
```

Cada rama se buildea y se sube por separado a su tienda. No hay un build único para las dos.

10. Paso 7 — Declaraciones de privacidad en las fichas

Las dos tiendas te van a pedir declarar qué datos maneja la app, y tiene que coincidir con la política de privacidad o rebotan la revisión. Según lo que la app realmente hace:

- **Apple (App Privacy)** — vinculado a la identidad del usuario: datos de contacto (nombre, email, teléfono), identificadores de usuario, datos de salud y fitness, contenido del usuario (mensajes, fotos) y compras. Marcá «no usado para seguimiento»: no hay publicidad ni tracking entre apps.
- **Google Play (Data safety)** — datos personales, datos de salud y fitness, mensajes, fotos e información de compras. Declarás cifrado en tránsito y que el usuario puede pedir la eliminación.
- **Clasificación por edad** — 16+, consistente con la política de privacidad.
- **URL de la política** — la misma en las dos tiendas y en el campo privacyPolicyUrl de app.json en las dos ramas. Hoy apunta a la copia publicada; cambiala por la tuya.

La política de privacidad viene escrita, en cuatro idiomas y ya publicada en https://landing-jolt.vercel.app/privacy.html, que es el valor que hoy tiene el campo privacyPolicyUrl de app.json en las dos ramas. Antes de publicar en las tiendas, hospedala en tu propio dominio y reemplazá el correo de contacto por el tuyo: desde la transferencia el responsable del tratamiento sos vos.

11. Cosas que conviene no cambiar

- **Los identificadores de los planes** — jolt_starter, jolt_pro y jolt_elite tienen que coincidir exactamente en las dos tiendas y en el backend. Renombrarlos implica tocar código en tres lugares y romper las compras existentes.
- **El catálogo de planes del servidor** — los límites de cada plan se calculan en el servidor y se aplican al crear cada recurso. Cambiar los números es una línea de configuración, pero cambiar la estructura toca la lógica de suscripciones.
- **Los proveedores externos** — el producto está construido sobre Atlas, Render, Firebase Storage y Resend. Reemplazar cualquiera es posible pero es trabajo, no configuración.

12. Qué queda sin probar hasta que esto esté

Todo lo demás está probado end-to-end en simulador y emulador, con las compras validadas contra el servidor mediante el bypass de desarrollo. Tres cosas sólo se pueden verificar con las cuentas reales:

- Una compra real en la App Store y en Google Play, con el diálogo nativo de pago.
- El correo de recuperación de contraseña llegando a una casilla real.
- Las notificaciones push en un dispositivo físico, con APNs y FCM configurados.

13. Orden recomendado

1. Abrir la cuenta de Google Play y arrancar el closed test de 14 días. Todo lo demás corre en paralelo.
2. Decidir el bundle identifier de iOS y el package de Android, si querés cambiar los que vienen.
3. Contratar Atlas y Render, y desplegar la API con las variables reales.
4. Firebase Storage y Resend con el dominio verificado.
5. Crear los productos de suscripción en las dos tiendas.
6. Configurar los dos webhooks.
7. Apuntar las apps al backend, buildear y subir.
8. Completar las declaraciones de privacidad y publicar.

14. Soporte y garantía posventa

La venta incluye la totalidad del código —las dos apps y la API— y su transferencia definitiva. Desde la entrega el vendedor se desvincula del producto: no opera cuentas, no toca el código y no interviene en su evolución. Lo que sigue es lo único que se mantiene.

- **Fallas** — sin cargo. Si algo que debería funcionar no funciona, se corrige gratis. Para hacerlo el vendedor necesita acceso temporal a los repositorios, que ya son del comprador, y ese acceso lo concede el comprador. Lo que rompe un flujo se atiende con prioridad; lo que no impide usar la app, en un plazo más holgado.
- **Cambios** — no son fallas. Agregar, modificar o extender funcionalidad no entra en la garantía. Se evalúa caso por caso: si el vendedor considera que el cambio es viable y conveniente, se acuerda por separado, con precio cerrado y una seña previa según el alcance.
- **Puesta en marcha** — guía sin costo, siempre. Cuentas de Apple y Google, servidor, dominio, webhooks: instrucciones claras y, si hace falta, una demostración en video o una llamada. El vendedor no ejecuta la configuración por el comprador; las credenciales son del comprador y quedan a su nombre.
- **Código modificado por terceros** — fuera de alcance. Desde que el producto se modifica, se extiende o se integra con algo nuevo, esa parte deja de estar cubierta: no se puede garantizar código ajeno.
- **Mantenimiento, funciones nuevas y evolución del producto** — no incluidos, igual que el costo de las cuentas y los servicios.
- **Plazo de la garantía de fallas** — 90 días desde la entrega, escribiendo a ederfornero.ef@gmail.com.

> En una línea: si algo está roto y no debería estarlo, se arregla sin cobrar. Lo que se cobra aparte es el trabajo nuevo, y se acuerda antes de empezar.