Prueba técnica Full-Stack Engineer — Choppi

Objetivo

Implementar un **MVP end-to-end** pequeño que refleje el stack de Choppi y tu criterio técnico. Incluye **auth básica**, **catálogo** (Stores, Products, StoreProducts) y **vistas** para operarlo.

Stack

- **Backend:** TypeScript, **NestJS**, PostgreSQL, ORM (TypeORM), **Auth JWT**, **Swagger**.
- **Frontend (elige uno; el otro es bonus):**
  - **Flutter** (Bloc/Cubit + Dio), o
  - **Next.js** (TypeScript + App Router + Tailwind).
- **Infra:** Deploy simple accesible públicamente (Vercel/Render/Railway/Fly/etc.). *(Opcional: Docker).*

Requisitos mínimos

Backend (obligatorio)

- **Auth:** Login por email/contraseña; emitir y verificar **JWT**; proteger **POST/PUT/DELETE**.
- **Stores:**
  - GET /stores (paginado; ?q= para búsqueda por nombre)
  - GET /stores/:id
  - POST /stores, PUT /stores/:id, DELETE /stores/:id *(soft-delete opcional)*
- **Products (globales):**
  - GET /products/:id
  - POST /products
- **StoreProducts (producto por tienda):**
  - POST /stores/:id/products → { productId, price, stock }
  - GET /stores/:id/products (paginado; filtros ?q=, ?inStock=true)
  - PUT /stores/:id/products/:storeProductId (price, stock)
  - DELETE /stores/:id/products/:storeProductId

- **Swagger** público; **migraciones y seeds** (2–3 stores, 10–20 products y asociaciones).

Frontend (puedes hacer las 2)

Opción A (Esta opción pesa más que la B) — Flutter - Pantallas: **Login**; **Stores** (lista + buscador); **Store Detail** (productos paginados + filtro inStock); **Product Detail**. - **Bonus:** carrito local (agregar/quitar; subtotal en cliente). - Estados de carga/errores; **1 animación simple**.

Opción B — Next.js - Páginas: **/login**; **/stores** (lista + buscador); **/stores/[id]** (productos paginados + inStock); **/products/[id]**. - Protección: sólo páginas de **crear/editar** requieren login. - **Bonus:** página **/admin** para crear/editar Store y Product.

(Opcional) Cálculo de carrito

- Endpoint: POST /cart/quote con [{ storeProductId, quantity }] → devuelve **subtotal** (precio*qty). *No se requiere persistir pedidos.*

Entregables

- **URLs públicas** de **API** (Swagger) y **Frontend** (o **APK** si usas Flutter).
- **Repositorio(s)** con **README**: setup local, variables de entorno, migraciones/seeds, cómo desplegar.
- **Usuario demo** para login (email y clave).

Criterios de evaluación

- **Correctitud de requisitos (40%)**.
- **Calidad de código y arquitectura (25%)**.
- **UX básica**: cargas/errores, paginación y filtros (20%).
- **Entrega**: README claro, seeds, deploy accesible (15%).
- **Bonus** (hasta +10%): segunda UI, Docker, búsqueda tolerante a typos/acentos, endpoint de carrito, etc.

Pautas

- Tipado estricto (**TS**), **class-validator**, errores consistentes.
- **Paginación obligatoria** en listados (>20 ítems).

- **Búsqueda mínima:** ILIKE por nombre. *(Si agregas fuzzy/acentos, cuenta como bonus).*
- Commits con mensajes claros.

Plazo

5 días corridos desde el envío de esta prueba.

Envío

Comparte links a **repo(s)**, **deploy(s)** y **Swagger**; incluye **APK** si usaste Flutter.