Prueba Técnica - Desarrollador Full Stack

DISCLAIMER IMPORTANTE

Esta es una prueba técnica con fines exclusivamente evaluativos. Los datos presentados son completamente ficticios y generados únicamente para propósitos de evaluación. Los resultados y análisis producidos durante esta prueba NO serán utilizados por la empresa para fines comerciales reales. Esta evaluación no conlleva compensación económica y su único objetivo es medir las competencias técnicas del aspirante.

Tiempo: 48 horas.

Debes construir una **mini-app de gestión de solicitudes ("Tickets")** para un equipo interno:

- Usuarios pueden **crear**, **listar**, **filtrar**, **editar** y **cerrar** tickets.
- Cada ticket tiene: `id`, `title`, `description`, `priority` (Low/Med/High), `status` (Open/In Progress/Closed), `assignee`, `createdAt`, `updatedAt`, `tags[]`.
- Debe haber **auth básica** (login) y **roles** (Admin puede eliminar; User no).
- Se espera **UI responsive**, accesible y con diseño limpio.

Entregables

Backend (API REST)

- Endpoints:
  - `POST /auth/login` → devuelve JWT o similar.
  - `GET /tickets?status=&priority=&q=&page=&limit=` con **paginación** y **búsqueda** por texto (en `title`/`description`) + filtros por `status`/`priority`.
  - `POST /tickets` (crear), `GET /tickets/:id`, `PATCH /tickets/:id`, `DELETE /tickets/:id` (solo Admin).
- **Validaciones** (p.ej., `title` requerido, `priority` en conjunto permitido, etc.).
- **Persistencia**: libre (SQLite/Postgres en docker). Semilla inicial (10–20 tickets).

- **Tests**: al menos unit tests de servicios y 1–2 de integración (auth + CRUD).
- **Performance**:
    - Lista de tickets < 200ms (p99 < 500ms) con 1k registros.
    - Búsqueda paginada eficiente (índices / estrategia).
- **Seguridad**:
    - JWT en `Authorization: Bearer`.
    - Sanitizar input, evitar inyección, CORS bien configurado.
    - Rate limit básico (p.ej., 60 req/min/ip) opcional.

Frontend (SPA)

- Tech sugerida: **React + TypeScript** (libre si prefieres otra).
- Vistas:
    - **Login**.
    - **Listado**: tabla con filtros (status, priority), búsqueda, paginación, badges de estado, orden por fecha, loader skeleton.
    - **Detalle/Edición**: formulario validado (client-side), tags con chips, cambio de estado.
    - **Creación** de ticket.
- **UX/UI**:
    - La empresa es IQ Outsourcing ten en cuenta la gama de colores corporativos para la estetica
    - Diseño claro y consistente (spacing, tipografía, jerarquía visual).
    - Accesibilidad: labels, focus states, contraste, navegación teclado.
    - Empty states y mensajes de error/éxito.
- **Estado**:
    - Manejo de cache y error en llamadas.

    - Loading states diferenciados (global vs. por componente).

- **Tests**: 3–5 pruebas (render, interacción de filtros, validaciones).

DevOps / Calidad

- **Docker** para back y front (y DB si aplica).
- **Makefile o npm scripts**: `dev`, `test`, `lint`, `build`, `start`.
- **Lint + Formatter** (ESLint/Prettier o equivalentes).
- **README** con:
    - Arquitectura breve (diagrama simple o bullets).
    - Pasos de ejecución local con un solo comando (`docker compose up` recomendado).
    - Decisiones técnicas y trade-offs.
    - Tiempo invertido y pendientes.
- (Opcional) **CI básico** (GitHub Actions) corriendo `lint` + `test`.