v.1.0. – 23 octubre 2025

Prueba técnica – Full Stack Junior

1. Instrucciones generales

- Tiempo estimado de trabajo para la prueba obligatoria: **3-5 horas**.
- La entrega debe incluir: código fuente (preferiblemente con control de versiones como Git), instrucciones de instalación/ejecución, breve README con decisiones tomadas, posibles mejoras.
- El candidato puede usar frameworks/librerías a su elección (por ejemplo: React, Vue, Svelte, Angular, o en backend Express, FastAPI, Django, etc.).
- Se valorará: claridad del código, estructura modular, separación frontend/backend, tests básicos, manejo de errores, facilidad para extender.
- Obligatorio: frontend + backend mínimos funcionales.
- Opcional: funcionalidades extras.

2. Contexto del ejercicio

Imagina que Moradai quiere lanzar un pequeño módulo de "referidos" para su plataforma. El concepto de referido se basa en que, a un usuario ya existente, se le genera un código que puede compartir con un conocido. Si este conocido compra el producto, a la persona que refirió se le abona un cierto importe. Simplificadamente, desarrollar:

- Backend: API para obtener lista de códigos de referidos existente, y para crear un código de referido aleatorio (se deja libertad para decidir el formato del código).
- Frontend: en una página intermedia de este estilo https://www.moradai.io/el-testigo-tu-vecino-virtual introducir una funcionalidad de validación del código de referido.
- Persistencia: puede usarse base de datos ligera (SQLite, PostgreSQL, Mongo, o incluso archivo JSON para simplificar) o en memoria si lo prefieres pero documenta la decisión.
- Se deja a liberta la autenticación de la API

3. PARTE A – Preguntas obligatorias

Backend: API básica

Crea un pequeño servidor que exponga dos endpoints:

- GET /code→ devuelve lista de códigos de referidos (cada elemento tiene al menos: id, código, id usuario que ha generado el código, fecha de creación)
- POST /code→ recibe una llamada (con el id del usuario que solicita generar el código) y lo añade, devolviendo el código generado con id + fecha.

Manejo de errores: por ejemplo, si en el POST falta el id del usuario, devuelve código 400 con mensaje.

Asegúrate de que los datos persisten (temporalmente está bien con memoria, pero idealmente base de datos ligera).

Incluye una prueba o dos (por ejemplo test unitario o test de integración) para cada endpoint.

Frontend: interfaz mínima

Crea una interfaz sencilla donde:

- Se pueda introducir el código de referido
- Se valide que el código es válido
- Se avance a otra página de confirmación de compra (esta página no hay que diseñarla)

Documentación & despliegue local

Incluye un archivo README que explique cómo instalar (dependencias, configuración, entorno), cómo ejecutar (backend + frontend).

Explica qué decisiones tomaste (por ejemplo: qué framework, por qué, limitaciones, cómo escalarías).

Preguntas teóricas (respuestas en README o archivo aparte)

a) El código de referido no deja de ser una promoción para captar clientes. Existen diferentes tipos de promociones como afiliaciones, promos de descuento, etc ¿Cómo modularizarías el código para que en el futuro fuera escalable y los mismos servicios aceptasen otro tipo de promociones?

b) ¿Qué aspectos de seguridad te preocuparían si esta API estuviera en producción (aunque sin autenticación)?

c) ¿Cómo implementarías paginación para GET /items si la lista pudiera tener miles de registros?

d) ¿Qué ventajas/desventajas ves en usar una base de datos relacional vs NoSQL para este módulo de “reservas/incidencias”?

4. PARTE B – Preguntas Opcionales

- Implementar edición (PUT /code/:id) y borrado (DELETE /code/:id) de códigos de referidos en el backend + reflejarlo en el frontend (en la validación).
- Usar autenticación básica para proteger los endpoints de creación/borrado.
- Desplegar en un entorno gratuito (por ejemplo Heroku, Vercel para frontend, o similar) y documentar la URL pública.
- ¿Qué mejorarías de esta página: https://www.moradai.io/el-testigo-tu-vecino-virtual?