■ Prueba Técnica – Fullstack JavaScript (Junior)

■ Consignas de la Prueba Técnica

Construir una aplicación fullstack que permita gestionar productos de una tienda.

■■ *Frontend (React)*

- Pantalla principal donde se listen todos los productos.
- Formulario para agregar un nuevo producto (nombre, precio y stock).
- Opción para editar un producto existente.
- Opción para eliminar un producto.
- Validaciones: nombre no vacío, precio > 0, stock entero >= 0.

■ *Backend (Node.js + Express + MongoDB)*

- GET /products → Devuelve todos los productos.
- POST /products → Agrega un producto nuevo.
- PUT /products/:id → Edita un producto.
- DELETE /products/:id → Elimina un producto.
- Validaciones en el backend con manejo de errores.

■ Requerimientos Técnicos

- Frontend: React con hooks (useState, useEffect).
- Backend: Node.js + Express.
- Base de datos: MongoDB (local o Atlas).
- Comunicación: el frontend debe conectarse al backend con fetch o axios.
- Código limpio y organizado.

■ Entregables

- Repositorio en GitHub con dos carpetas: frontend/ y backend/.
- README.md con instrucciones de ejecución.
- (Opcional) Capturas o GIF demostrativo.

■ Preguntas adicionales para entrevista

- ¿Cómo estructuraste tu backend y por qué?
- ¿Cómo manejaste el estado en React?
- ¿Qué harías para mejorar la seguridad de la API?
- ¿Qué cambiarías si el proyecto creciera a gran escala?

■ Preguntas típicas de JavaScript (con respuestas)

■ *¿Cuál es la diferencia entre var, let y const?*

■ var tiene scope de función o global, let y const tienen scope de bloque. const no permite reasignación.

■ *¿Qué es una Promesa en JavaScript y cómo funciona async/await?*

■ Una Promesa representa el resultado eventual de una operación asíncrona. async/await facilita su manejo.

■ *¿Cuál es la diferencia entre == y ===?*

■ == compara con coerción de tipo, === compara valor y tipo estrictamente.

■ *¿Qué es el Event Loop en JavaScript?*

■ Es el mecanismo que permite que JS maneje operaciones asíncronas aunque sea single-threaded.

■ *¿Qué son los Hooks en React? Explicá useState y useEffect.*

■ Hooks son funciones especiales para usar estado y efectos. useState maneja estado, useEffect efectos secundarios.

■ *Ejercicio: dado un array de números, devolvé los pares.*

■ Ejemplo: arr.filter(num => num % 2 === 0).

■ *¿Cómo funciona try/catch?*

■ Permite atrapar errores de ejecución. En async/await se combina para manejar promesas rechazadas.

■ Mini Test Práctico de JavaScript

1. Escribí una función que reciba un string y devuelva cuántas vocales tiene.
2. Dado un array de números, devolvé el número más grande sin usar Math.max.
3. Escribí una función que invierta un string (ej: 'hola' -> 'aloh').
4. Dado un array de objetos con {nombre, edad}, devolvé el nombre de la persona más joven.
5. Implementá una función que reciba dos arrays y devuelva los elementos en común.