TECNOLOGÍAS DE LA INFORMACIÓN

PRUEBA TÉCNICA – PROGRAMADOR/A FULLSTACK TYPESCRIPT

Instrucciones Generales

- La prueba técnica consiste en dos partes: una parte de backend y otra parte de frontend.
- Se puede utilizar cualquier recurso de documentación oficial.
- Entrega el código en un repositorio público (GitHub, Bitbucket, etc.) o como un archivo comprimido.
- Se valoran buenas prácticas de desarrollo, organización del código y documentación clara.

Criterios de evaluación

- Correcto uso de TypeScript.
- Uso de versiones recientes de NodeJs (>= v20), NestJs (>= v10) y React (>= v18).
- Arquitectura modular y limpia en NestJS y React.
- Implementación segura de autenticación y autorización.
- Uso adecuado de acceso a datos (ORM y base de datos).
- Buenas prácticas en React (gestión del estado, componentes reutilizables, etc.).
- Documentación clara (README con instrucciones de instalación y uso).

Requisitos no obligatorios pero valorables. (Opcional)

- Proporcionar ficheros de Docker para despliegue de la solución.
- Implementación de tests unitarios en frontend.
- Implementación de documentación de API backend con Swagger.

Backend - API REST con NestJS

- Implementar una API REST en NestJS con las siguientes entidades:
    - **Usuario**: id, nombre, email, password.
    - **Tarea**: id, título, descripción, estado (pendiente, en progreso, completada), usuario Id (FK).
- Implementar autenticación de usuarios con JWT:
    - Registro (POST /auth/register): Permitir a los usuarios registrarse.
    - Login de usuarios (POST /auth/login): Devolver un token JWT al usuario.
- Endpoints para la gestión de tareas:
    - POST /tareas: Crear una tarea.
    - GET /tareas: Obtiene todas las tareas del usuario autenticado.
    - GET /tareas/:id: Ver una tarea (solo si la tarea pertenece al usuario autenticado).
    - PUT /tareas/:id: Editar una tarea (solo si la tarea pertenece al usuario autenticado).
    - DELETE /tareas/:id: Eliminar una tarea (solo si la tarea pertenece al usuario autenticado).
- Implementar test unitarios como mínimo para las funcionalidades de login de usuario y creación y borrado de tareas.
- Usar una base de datos relacional (MySQL, PostgreSQL) con un ORM (TypeORM por ejemplo) para la persistencia de datos.
- Implementar gestión de errores adecuada y validaciones de tipos de datos en los endpoints.

Frontend - React

- Desarrollar una aplicación en React que consuma la API creada en la parte de backend.
- Implementar un sistema de autenticación con JWT en el frontend.
- Mostrar una lista con las tareas, permitiendo:
    - Crear nuevas tareas.
    - Editar tareas existentes.
    - Eliminar tareas.
- Utilizar una librería de componentes UI (Bootstrap, Material UI, NextUI, etc.).
- Gestionar el estado global de la aplicación con React Context.