Prueba Técnica para el Cargo de Desarrollador Web Junior

Duración máxima: 3 horas

Instrucciones Generales
- La prueba debe realizarse de manera individual.
- Crea un repositorio en GitHub y sube tu código al finalizar, compartiendo el enlace.
- Documenta brevemente tus decisiones en un archivo `README.md` dentro del repositorio.
- Asegúrate de seguir las mejores prácticas de código.

Parte 1: Fundamentos de JavaScript (30 minutos)

Responde las siguientes preguntas escribiendo código claro y funcional:

1. Manipulación de Arrays:
Escribe una función llamada “procesarArray” que reciba un array de números y devuelva un nuevo array que contenga:
   - Los números pares multiplicados por 2.
   - Los números impares multiplicados por 3.

   Ejemplo:
   procesarArray([1, 2, 3, 4]); // Resultado: [3, 4, 9, 8]

2. Promesas y manejo de asincronía:
Escribe una función “obtenerDatos” que simule la obtención de datos desde una API utilizando “Promise”. La función debe retornar los datos proporcionados después de 2 segundos.

 obtenerDatos().then((data) => console.log(data));
   // Resultado esperado: "Datos obtenidos"
   ```

3. Clases en JavaScript:
Crea una clase “Persona” con:

   - Propiedades: “nombre” y “edad”.
   - Un método “saludar” que imprima en consola “Hola, soy {nombre} y tengo {edad} años.”

Parte 2: Ejercicio práctico con React.js y librerías (2 horas)

Objetivo: Construir un formulario funcional utilizando React.js, Formik, Axios y Redux.

Requisitos del Proyecto:

1. Configuración inicial:
   ```

- Crea un proyecto con “create-react-app” o “Vite”.
- Configura Redux para manejar el estado global.

2. Formulario:
- Crea un formulario utilizando Formik que contenga los siguientes campos:
- Nombre (texto, requerido, mínimo 3 caracteres).
- Correo electrónico (formato de email, requerido).
- Mensaje (texto, requerido, mínimo 10 caracteres).
- Agrega validaciones con mensajes de error claros.

3. Consumo de API:
- Simula el envío del formulario haciendo una petición POST a una API ficticia utilizando Axios.
- Muestra un mensaje de éxito o error según corresponda.

4. Gestor de estado con Redux:
- Crea un slice en Redux para almacenar los datos del formulario enviado.
- Despliega una lista con los datos de todos los formularios enviados (almacenados en el estado global).

Estructura esperada del proyecto:

```
src/
|-- components/
|   |-- Formulario.js
|   |-- ListaFormularios.js
|-- redux/
|   |-- formSlice.js
|   |-- store.js
|-- App.js
|-- index.js
```

Criterios de Evaluación:
- Uso correcto de React.js y organización del proyecto.
- Implementación de Formik con validaciones claras.
- Uso adecuado de Axios para el manejo de peticiones.
- Configuración y manejo de Redux para el estado global.
- Buenas prácticas de código y claridad en la documentación.

Parte 3: Pregunta de reflexión (30 minutos)

En el archivo `README.md`, responde la siguiente pregunta:

Si tuvieras que optimizar tu solución para manejar un volumen masivo de formularios enviados, ¿qué cambios implementarías? Explica tus decisiones técnicas.