Descripción General:

Esta prueba técnica está orientada a conocer tus habilidades como **Desarrollador de Arquitectura Frontend**. Queremos ver cómo abordas la transformación de una maqueta HTML en una **WebApp en React o Next.js**, con especial enfoque en la organización del código, la separación de lógica de transformación de datos fuera del frontend, y la implementación de mejores prácticas. Además, si eliges **Next.js**, deberás justificar el uso de un **BFF (Back for Frontend)** y cómo separas las capas frontend y backend, explicando tus decisiones técnicas.En ambos casos debe hacer render SSR.

Requisitos

Opción 1: React desde cero

1. **Estructura del Proyecto**:
    - Inicia el proyecto desde cero (sin usar herramientas como **Create React App** u otros generadores automáticos) con React, configurando manualmente **Webpack** y **Babel** para el entorno de desarrollo.
    - Organiza el código dentro de una estructura clara bajo `/src`, dividiendo componentes y servicios de manera lógica.
    - Incluye configuraciones de ESLint y Prettier para mantener buenas prácticas de estilo y calidad del código.

Opción 2: Next.js con Back for Frontend (BFF)

1. **Separación de Capas**:
    - Si decides usar **Next.js**, debes estructurar el proyecto con una separación clara entre el frontend y el backend. La carpeta de backend debe llamarse `bff` y contener toda la lógica que se encargue de las interacciones con la API y la transformación de datos.
    - El frontend debe estar separado en una carpeta `front` y manejar las vistas, componentes y la lógica propia del renderizado.
    - La comunicación entre frontend y backend debe ser clara, utilizando una arquitectura de tipo **API Client**.

Estructura Sugerida (Opción 2 - Next.js + BFF):

Si eliges la **Opción 2 (Next.js + BFF)**, utiliza una estructura de proyecto similar a la siguiente, con una clara separación de responsabilidades entre el **frontend** y el **BFF**. Esta estructura ayuda a aislar la lógica de transformación de datos fuera del frontend y proporciona un entorno bien organizado para el desarrollo.

Ejemplo de Estructura Sugerida:

```
.
├── README.md
```

```
├── bff              # Lógica del Backend for Frontend
|   ├── Dockerfile
|   ├── src
|   |   ├── controllers
|   |   ├── repositories
|   |   ├── routes
|   |   ├── services
|   |   └── utils
|   └── tests
├── front            # Lógica de Frontend con Next.js
|   ├── Dockerfile
|   ├── src
|   |   ├── components
|   |   ├── hooks
|   |   ├── pages
|   |   ├── services
|   |   └── utils
└── docker-compose.yml
```

2. Esta estructura separa las responsabilidades de transformación de datos en el backend (BFF) y el renderizado en el frontend, facilitando la escalabilidad y mantenibilidad del proyecto.

Detalles Técnicos

Assets Proporcionados:

- **Maqueta HTML**: Maqueta HTML
- **Estilos CSS**: Estilos CSS
- **Endpoint**: Endpoint JSON (Método GET)

Objetivos Específicos:

1. **Componentización de la Maqueta HTML:**
    - Crear una aplicación en React/Next.js que transforme la maqueta HTML proporcionada en componentes reutilizables, con el objetivo de replicar y estructurar la maqueta provista

    - Cada parte del HTML (e.g., encabezado, lista de artículos, tags) debería ser un componente separado.

2. **Consumo del Endpoint:**
    - **Filtrado de Artículos:**
        - Obtén los 30 artículos desde el endpoint proporcionado.
        - Filtra los artículos para quedarte solo con aquellos cuyo atributo `"subtype"` sea igual a `"7"`.
    - **Agrupación y Totalización de Tags:**
        - Cada artículo tiene tags dentro del atributo `"taxonomy.tags"`.
        - Agrupa todos los tags de los 30 artículos filtrados.
        - Totaliza cuántas veces aparece cada tag, y ordénalos de mayor a menor.
        - Muestra los 10 tags más frecuentes en la parte debajo del título “Acumulado Grilla”.
        - Cada tag debe mostrarse como un enlace con la forma: `/tema/[tag.slug]`.
    - **Renderizado de los Artículos:**
        - Renderiza los 30 artículos en una grilla.
        - Usa los siguientes atributos para mostrar cada artículo:
            - Imagen: `promo_items.basics.url`
            - Título: `headline.basic`
            - Fecha: `display_date` (respetando el formato de la maqueta proporcionada).
3. **Manejo de Estado y Lógica de Negocio:**
    - Si usas React, maneja el estado con React Hooks y, si lo consideras necesario, implementa Context API.
    - Si usas Next.js, la lógica de negocio y la transformación de datos deberían estar en el backend (BFF). El frontend debería recibir los datos ya preparados para ser renderizados.

Configuración del Entorno

- **React desde cero**: Si trabajas con React desde cero, debes configurar manualmente **Webpack** y **Babel** para el bundling y la transpilación del proyecto. Además, debes asegurarte de configurar correctamente **ESLint** y **Prettier** para mantener la calidad y consistencia del código.

- **Next.js**: Si decides usar **Next.js**, este ya incluye configuraciones base para **ESLint** y **Prettier**, pero asegúrate de personalizarlas según las necesidades del proyecto. También deberás gestionar la lógica del backend en el **BFF (Back for Frontend)**.
- **Docker (Opcional)**: Aunque no es obligatorio, recomendamos incluir un **Dockerfile** para cada parte del proyecto (frontend y backend). El uso de **docker-compose** para levantar ambas capas será considerado un plus, pero queda a criterio del desarrollador implementarlo.

Test y Buenas Prácticas

- Implementa **unit tests** para componentes críticos.
- Sigue patrones de **programación declarativa**.
- Asegúrate de que el código sea consistente y fácil de mantener.

Puntos Extra (Opcionales)

- **State Management**: Implementa una solución para el manejo de estado global con **Context API** o **Redux** si es necesario.
- **Despliegue**: Configura el despliegue en un PaaS gratuito como **Vercel** o **Heroku,** o alguno que conozcas.

Entregables Esperados

1. **Repositorio GitHub**: Comparte el enlace a tu repositorio con el código completo de la aplicación.
2. **Documentación**: Incluye un **README** que explique la estructura del proyecto, las decisiones técnicas que tomaste, y cómo organizaste el frontend y, si corresponde, el backend. Si utilizas **Next.js**, asegúrate de detallar cómo implementaste la separación entre el **frontend** y el **BFF (Back for Frontend)**.

Criterios de Evaluación

1. **Componentización y Modularidad**: Se evaluará cómo organizas el proyecto y los componentes para facilitar el mantenimiento y la escalabilidad.
2. **Separación de Capas (para Next.js)**: Si decides usar **Next.js**, nos gustaría ver cómo separas claramente el **frontend** del **backend**, utilizando el **BFF** para manejar la lógica de negocio y preparar los datos antes de enviarlos al frontend. Además, sería ideal que expliques por qué elegiste Next.js y cómo gestionaste la lógica de transformación de datos fuera del frontend.
3. **Configuración del Entorno**:Evaluaremos la correcta configuración de Webpack, Babel, ESLint, Prettier y Docker(opcional), así como la capacidad de trabajar con un entorno bien estructurado.
4. **Tests**:Los tests deben estar bien definidos y cubrir los casos críticos de la aplicación.

5. **Rendimiento y Optimización**:Nos interesa ver cómo manejas el rendimiento en listas grandes y optimizas el renderizado, evaluando cómo implementas **SSR** para mejorar la carga inicial y la experiencia del usuario.