ENUNCIADO:

Una empresa de comercio electrónico, *TechWorld*, ha decidido renovar su tienda en línea para ofrecer una experiencia más dinámica e interactiva a sus clientes. El objetivo es mejorar el compromiso de los usuarios, optimizar el proceso de pago y asegurar una integración fluida con los sistemas de retroalimentación de clientes. Para ello, la empresa se ha centrado en seleccionar la mejor arquitectura y herramientas de desarrollo web para garantizar escalabilidad, rendimiento y una experiencia de usuario sin interrupciones en distintos navegadores.

1. **Selección de Arquitectura y Herramientas de Programación**

Arquitectura de Microservicios:

Dado que *TechWorld* espera un aumento significativo en el tráfico durante las temporadas de descuentos, optaron por una **arquitectura de microservicios**. Esto permite al equipo de desarrollo dividir la aplicación en pequeños servicios independientes. Por ejemplo, el catálogo de productos, el carrito de compras y los sistemas de pago están gestionados por microservicios diferentes, permitiendo que cada uno escale de manera independiente durante los momentos de mayor tráfico.

Framework JavaScript (React.js):

Para crear una interfaz de usuario interactiva, *TechWorld* eligió **React.js** para el frontend. La arquitectura basada en componentes de React asegura que el equipo de desarrollo pueda reutilizar código eficientemente, acelerando el desarrollo y mantenimiento. Este framework también soporta la creación de **Aplicaciones de Una Sola Página (SPA)**, asegurando una experiencia de navegación fluida y rápida sin necesidad de recargar la página por completo.

Backend con Node.js:

En el backend, la empresa implementó **Node.js**, que es ligero y maneja operaciones asincrónicas de manera eficiente. Esto permite a *TechWorld* gestionar múltiples solicitudes de clientes simultáneamente, mejorando el rendimiento durante los periodos de tráfico intenso.

1. **Mecanismos de Ejecución de Código en los Navegadores Web**

El equipo de desarrollo se aseguró de que todo el código ejecutado en el navegador (del lado del cliente) siguiera **mejores prácticas de optimización de rendimiento**. Esto incluyó:

- **Ejecución de JavaScript**: El equipo utilizó **JavaScript** para crear interacciones dinámicas, como la búsqueda de productos y la adición de artículos al carrito de compras. Con llamadas asincrónicas **AJAX**, los clientes pueden añadir productos al carrito sin recargar la página.

- **APIs del Navegador**: Para mejorar la experiencia del usuario, *TechWorld* integró APIs del navegador como **Geolocalización**, que proporciona a los clientes las ubicaciones de tiendas cercanas, y **localStorage**, que guarda los artículos del carrito incluso si el cliente abandona el sitio y regresa más tarde.

1. **Capacidades y Limitaciones de los Navegadores**

El equipo de desarrollo probó la plataforma en distintos navegadores (Chrome, Firefox, Safari, Edge) para garantizar una experiencia sin problemas.

Capacidades:

•Todos los navegadores modernos soportaron características de **HTML5, CSS3 y JavaScript**, asegurando una apariencia y funcionalidad coherente en todas las plataformas.

Limitaciones:

•Algunas animaciones avanzadas y características, como **WebAssembly**, no estaban completamente soportadas en versiones más antiguas de navegadores, por lo que se proporcionaron métodos alternativos.

Para superar estos desafíos, el equipo usó herramientas de **detección de características** para asegurar que la funcionalidad del sitio estuviera optimizada para las capacidades del navegador del usuario.

1.**Lenguajes de Programación y Tecnologías del Lado del Cliente**

El **lenguaje principal** utilizado fue **JavaScript**, con mejoras proporcionadas por **TypeScript** para mayor seguridad en los tipos de datos y mejor escalabilidad del código. Además, el equipo utilizó **HTML5** para estructurar el contenido y **CSS3** para gestionar el diseño visual y la capacidad de respuesta del sitio.

JSX en React fue clave para integrar JavaScript directamente en HTML, lo que permitió a los desarrolladores gestionar fácilmente contenido dinámico, como mostrar recomendaciones personalizadas según el historial de navegación.

1.**Tecnologías y Lenguajes Asociados**

El equipo también utilizó tecnologías como:

•**Sass**: Para gestionar el CSS de manera más organizada y modular en proyectos grandes.

•**Progressive Web Apps (PWA)**: Para permitir a los usuarios instalar la aplicación web en sus dispositivos y proporcionar una experiencia offline durante interrupciones de red.

Al utilizar **arquitectura de microservicios**, **React.js** y APIs modernas del navegador, *TechWorld* construyó una tienda en línea dinámica y escalable. Las pruebas en varios navegadores garantizaron la compatibilidad y una experiencia fluida para los usuarios, mientras que la integración de tecnologías como **PWA** y **AJAX** mejoró el compromiso de los usuarios.

Recomendaciones:

•Implementar más características de **WebAssembly** para herramientas de visualización de productos y ofrecer modelos 3D para productos electrónicos de alta gama.

•Enfocarse en pruebas continuas de navegadores para estar al tanto de nuevas características y optimizar aún más la experiencia del usuario.

PREGUNTAS DEL CASO

1.**Selección de Arquitectura y Herramientas**:

•¿Cuál crees que es la arquitectura más adecuada para un proyecto de comercio electrónico de gran escala, como una tienda online con muchos productos y usuarios? ¿Microservicios, monolítica o serverless? Justifica tu elección.

La arquitectura más adecuada parece ser la elegida en este caso que es la arquitectura de microservicios. La razón es que esta arquitectura separa la aplicación en pequeños servicios independientes. La modularidad hace mas fácil el mantenimiento, escabilidad y tiene una gran flexibilidad. Tanto la arquitectura monolitica como la s serverless serían mucho mas complejas y dificiles en cuanto a mantenimiento y despliegue al no tener diferenciadas las funciones.

•¿Qué framework frontend seleccionarías (React, Angular, Vue.js) para desarrollar la interfaz del usuario en este proyecto? Explica las razones detrás de tu elección.

Ahora mismo elegiría React por su curva de aprendizaje y uso, además permite crear aplicaciones web progresivas (PWA) y es más rápido que Vue y Angular. Angular tiene una sintaxis mas compleja y Vue tiene mas problemas en proyectos enormes de cara a la integración asi como una comunidad mas pequeña para la resolución de dudas y problemas.

2.**Mecanismos de Ejecución de Código en el Navegador**:

•Describe cómo se ejecuta el código JavaScript en el navegador y cómo puedes utilizar **AJAX** para mejorar la experiencia del usuario en una tienda en línea.

JavaScript ejecuta el código en el mismo navegador a tiempo real, ademas usando AJAX la página web funciona de manera asincrona, es decir las nuevas peticiones quedan en segundo plano y no hace falta recargar la página principal para que aparezca el nuevo contenido.

Por ejemplo, actualizar el carrito de compra sin tener que salir de esa página y a tiempo real.

•¿Cómo puedes aprovechar las APIs del navegador, como la **API de Geolocalización** o **localStorage**, para mejorar la experiencia del cliente en una tienda online?

Usos de la API de Geolocalización : sugerir tiendas cercanas, cambiar la divisa de moneda de pago o el idioma del navegador según la localización donde nos encontremos.

Usos de la API de localStorage: guardar el carrito de compra aun cuando hemos salido de la página web, quedando almacenado hasta que se limpie los datos de caché del navegador.

También se pueden guardar algunas preferencias de usuario como : idioma, tema oscuro/claro, incluso datos de inicio de sesión.

3.**Capacidades y Limitaciones de los Navegadores**:

•¿Cuáles son algunas de las limitaciones que puedes encontrar al desarrollar para diferentes navegadores? ¿Cómo puedes asegurar que tu sitio funcione correctamente en todos ellos?

Algunos de los problemas más comunes en las limitaciones con navegadores estan relacionadas con CSS3 y APIs avanzadas como WebAssembly en relacion con navegadores mas obsoletos o antiguos.

Formas de solucionar incompatibilidad con CSS:

Utilizar **CSS y HTML estándar.** Realizar pruebas en los navegadores más usados, como Chrome, Firefox, Safari y Edge. Emplear herramientas de **validación de código** para detectar y corregir errores como W3C. Usar frameworks o librerías que aseguren la compatibilidad entre navegadores. Aplicar técnicas de diseño responsive para una correcta visualización en distintos dispositivos.

WebAssembly (Wasm) es un formato de código binario que permite ejecutar código de manera eficiente en navegadores web. Proporciona un rendimiento cercano al nativo y permitiendo que lenguajes como C, C++, Rust y otros se ejecuten en la web. Exiten incompatibilidades conocidas con todo navegador anterior a 2017, Internet Explorer asi como versiones antiguas de Safari.

•¿Qué herramientas usarías para probar y optimizar el rendimiento de tu aplicación web en diferentes navegadores?

Lo mas fundamental para depurar, optimizar y mejorar el rendimiento en los navegadores web son las propias herramientas de desarrollo incorporadas en esos navegadores como : **Chrome DevTools**, **Firefox Developer Tools**, **Safari Web Inspector** y **Edge DevTools.**

4.**Lenguajes de Programación en el Entorno Cliente**:

•Explica la diferencia entre **JavaScript** y **TypeScript**. ¿En qué tipo de proyectos recomendarías utilizar TypeScript?

JavaScript: tipado dinámico. Interpretado directamente en el navegador. Detecta errores en tiempo de ejecución.

TypScript: tipado estático opcional. Debe compilarse a JavaScript. Detecta errores en tiempo de compilación.

TypeScript está recomendado para proyectos grandes y complejos. El tipado estático reduce errores. Ademas si los proyectos van a ser escalables mantiene un código mas claro y facil de refactorizar asi como el desarrollo en equipos.

•¿Cómo integrarías CSS y JavaScript en un proyecto React para garantizar que la interfaz sea atractiva y funcional en todos los dispositivos?

Usaria Styled Components que es una popular librería para inyectar CSS en JavaScript, es verdad que no es específica de React pero gracias a esta librería los estilos quedan encapsulados dentro de los componentes , no hay colisiones de nombre de clases y lo hace una buena opción para gestionar los estilos en grandes proyectos.

5.**Tecnologías y Lenguajes Asociados**:

•¿Qué ventaja ofrece el uso de preprocesadores como **Sass** en lugar de CSS puro en un proyecto de gran escala?

Sass es un preprocesador CSS que genera automaticamente hojas de estilo y le añade variables propias de un lenguaje de programación como variables, herencias , funciones, etc, esto reduce el tiempo para crear y mantener CSS asi como mantiene una modularidad en cuanto a la organización de estilos que lo hace perfecto para un proyecto de gran escala.

•¿Cómo puedes mejorar el rendimiento de un sitio web utilizando **Progressive Web Apps (PWA)** y qué beneficios aportan a los usuarios?

Las PWA intengran lo mejor de las aplicaciones moviles con los navegadores web, permiten almacenar caché haciendo la carga más rápida mediante los service workers , mediante estos service workers además pueden cargar sin conexión.

Uno de los benificios a los usuarios además de los menciones es el manejo de notificaciones push que pueden hacer recordar al usuario que mantiene la página web en segundo plano o incluso que hay nuevo contenido.

6.**Conclusión**:

•¿Qué medidas tomarías para asegurar que el sitio web sea escalable y pueda soportar un gran volumen de tráfico durante periodos de alta demanda?

Arquitecta escalable como microservicios. Utilizar tecnologias como Docker y Kurbenetes para el despliegue de multiples servicios.

Uso de cache para mejorar la carga y respuesta en los navegadores.

Optimizar código , que sea limpio y ordenado.

Optimizar base de datos , una buena indexación de base de datos mejora el rendimiento.

•¿Qué herramientas y prácticas implementarías para mejorar la seguridad y el rendimiento del sitio web?

Para mejorar la seguridad:

Implementación HTTPS con certificados SSL/TLS para cifrar la comunicación entre el servidor y el cliente, para proteger información delicada y sensible.

Autentificación de dos factores, añadiendo una capa extra de seguridad en el inicio de sesión.

Actualización de bibliotecas y dependencias de forma regular para protección contra vulneradidades conocidas.

Para mejorar el rendimiento:

Uso de cache para mejorar la carga y respuesta en los navegadores.

Optimizar código , que sea limpio y ordenado.

Optimizar base de datos , una buena indexación de base de datos mejora el rendimiento.

Optimización de CSS y JavaScript, uso de service workers.