POSIBLES PREGUNTAS PARA UN DESARROLLADOR ANGULAR

DESARROLLADOR JUNIOR

¿Qué es Angular?

Angular es un framework de desarrollo de aplicaciones web desarrollado y mantenido por Google. Se utiliza para construir aplicaciones web (SPA), librerías y web progresivas (PWA

¿Qué es Angular CLI y cuál es su propósito en el desarrollo de aplicaciones Angular?

Angular CLI (Command Line Interface) es una herramienta de línea de comandos que se utiliza para inicializar, desarrollar y mantener aplicaciones Angular de manera rápida y eficiente. Su propósito es automatizar tareas comunes de desarrollo, como la creación de componentes, servicios, módulos, y la construcción y ejecución de la aplicación.

¿Qué es un componente en Angular y cómo se crea uno nuevo?

Un componente en Angular es una pieza modular de la interfaz de usuario que encapsula la lógica y la presentación de una parte específica de la aplicación. Se crea utilizando el comando **ng generate component** seguido del nombre del componente. Por ejemplo:

ng generate component nombre-del-componente

¿Qué es la interpolación en Angular y cómo se utiliza en las plantillas HTML?

La interpolación en Angular es una forma de vincular datos del controlador (.ts) hacia el HTML. Se realiza utilizando dobles llaves **{{ }}**. Por ejemplo, para mostrar el valor de una propiedad nombre del componente en la plantilla, se haría así: **{{ nombre }}**

¿Qué es una directiva?

Una directiva en Angular es una instrucción que se utiliza para agregar comportamiento a elementos HTML existentes o para manipular la estructura del DOM.

¿Cuál es la diferencia entre las directivas estructurales y las directivas de atributo en Angular?

Las directivas estructurales alteran la estructura del DOM añadiendo, eliminando o reemplazando elementos, como ***ngIf** y ***ngFor.** Las directivas de atributo modifican la apariencia o el comportamiento de un elemento, como **ngClass** y **ngStyle**.

¿Cómo se manejan los eventos en Angular? Proporciona un ejemplo.

Haciendo uso de los eventos binding en el HTML utilizando la sintaxis **(evento)="función()"**
Por ejemplo:

Html:

```
<button (click)="onClick()">Click Me</button>
```

Y en el typescript:

```
onClick() {
  console.log('Button clicked');
}
```

¿Qué es el routing en Angular y cómo se configura?

El routing en Angular es el proceso de navegar entre diferentes vistas de una aplicación. En versiones menores a la 14, se configura utilizando el módulo **RouterModule** y la directiva **router-outlet**. Además, se definen rutas utilizando las clases **Routes** y **Route**, e informamos a Angular sobre estas rutas con el servicio **RouterModule.forRoot()** en el módulo principal de la aplicación. Si estas usando Angular en su versión 14 o superior solo basta con crear las rutas usando las clases Routes y Routes y definirlas en el archivo main.ts usando la función **provideRouter()**

¿Qué son los módulos en Angular y cuál es su propósito?

Los módulos en Angular son contenedores que agrupan componentes, directivas, pipes y servicios relacionados. Su propósito es organizar y modularizar la aplicación para mejorar la reutilización y el mantenimiento del código.

¿Cómo se realiza la comunicación entre componentes en Angular?

La comunicación entre componentes en Angular se puede realizar a través de los decoradores @input y @output, mediante servicios compartidos o utilizando la gestión de estado con herramientas como RxJS.

¿Qué son los servicios en Angular y para qué se utilizan?

Los servicios en Angular son clases que se utilizan para organizar y compartir lógica y datos entre diferentes partes de la aplicación. Se utilizan para realizar operaciones como llamadas a APIs, manipulación de datos, etc.

¿Qué es la inyección de dependencias y cómo se implementa en Angular?

La inyección de dependencias en Angular es un patrón de diseño que se utiliza para proporcionar instancias de objetos a una clase en tiempo de ejecución. Se implementa utilizando el mecanismo de inyección de dependencias de Angular, que se basa en el concepto de proveedores y la anotación **@Injectable().**

¿Qué son los observables en Angular y cuál es su utilidad?

Los observables en Angular son una forma de manejar secuencias de eventos asíncronos. Son muy útiles para trabajar con operaciones asincrónicas como peticiones HTTP, eventos del DOM, y comunicación entre componentes.

¿Qué es el lazy loading y cómo se implementa en Angular?

El lazy loading en Angular es una técnica que consiste en cargar módulos de manera diferida, es decir, solo cuando son necesarios. Se implementa definiendo rutas en el archivo de enrutamiento con la función **loadChildren**. A partir de la versión 17 tenemos lazy loading aplicado a componentes, usando el decorador **@defer**

¿Qué es un Interceptor y en caso lo debo usar?

Un interceptor en Angular es una clase que se utiliza para interceptar y modificar peticiones HTTP o respuestas antes de que sean enviadas o recibidas por el servidor. Se utiliza para añadir lógica de manejo de errores, autenticación, logueo, agregar headers, modificar url, etc.

¿Cuáles son las mejores prácticas para la gestión del estado en una aplicación Angular?

Algunas de las mejores prácticas para la gestión del estado en una aplicación Angular incluyen el uso de servicios para compartir datos entre componentes, la implementación de patrones como Redux o la gestión de estado local utilizando el sistema de cambio de detección de Angular. También es recomendable utilizar herramientas como RxJS para el manejo de flujos de datos asíncronos.

OTRAS PREGUNTAS

¿Usas alguna metodología para el css? BEM

¿Has trabajado con alguna metodología ágil de desarrollo? SCRUM

¿Has trabajado con GIT? Si

¿Tienes experiencia en algún framework o librería de componentes para angular?

PrimeNG, Angular Material, NgBoostrap, etc

DESARROLLADOR SEMI SENIOR

Gestión del estado:

¿Cómo gestionarías el estado de una aplicación Angular de tamaño medio?

Para una aplicación de tamaño medio en Angular, optaría por mantener un estado centralizado utilizando servicios para compartir datos entre componentes relacionados. Mantener una estructura clara y coherente de los datos es esencial para evitar confusiones y errores. Además, haría un uso inteligente de los observables para manejar eventos asíncronos y mantener una comunicación fluida entre los diferentes componentes.

¿Cuándo considerarías la implementación de NgRx en lugar de usar servicios y observables para la gestión del estado?

La implementación de NgRx se vuelve más relevante a medida que la aplicación crece en complejidad. Cuando el manejo del estado se vuelve complicado con servicios y observables, NgRx puede ser la solución adecuada. Si la aplicación necesita características avanzadas como viajes en el tiempo del estado o acciones asíncronas, o si simplemente quieres una gestión de estado más robusta y estructurada, es un buen momento para considerar NgRx.

Optimización y rendimiento:

¿Qué estrategias emplearías para mejorar el rendimiento de una aplicación Angular?

Para mejorar el rendimiento de una aplicación Angular, primero identificaría y optimizaría las operaciones que consumen más recursos, como las llamadas a la API o el procesamiento de datos. Luego, implementaría la carga diferida (lazy loading) para cargar recursos solo cuando sean necesarios, lo que reduce el tiempo de carga inicial. También utilizaría técnicas de optimización de imágenes y archivos estáticos, así como el almacenamiento en caché para reducir las solicitudes al servidor. También revisaría si existe codificación o componentes redundantes y así reducir la cantidad de archivos en el proyecto.

¿Cómo abordarías la optimización del tiempo de carga de una aplicación Angular?

Para optimizar el tiempo de carga, utilizaría la técnica de preloading para cargar módulos secundarios en segundo plano mientras el usuario interactúa con la aplicación. Además, minificaría y comprimiría los archivos CSS y JavaScript para reducir el tamaño de los archivos descargados. También aprovecharía las herramientas de análisis de rendimiento (por ejemplo , **google speed test o Lighthouse**) para identificar cuellos de botella y áreas de mejora en el proceso de carga.

Testing y calidad del código:

¿Cuál es tu enfoque para escribir pruebas unitarias y de integración en Angular?

Mi enfoque sería escribir pruebas unitarias para probar individualmente las partes más pequeñas y críticas de la aplicación, como los componentes y servicios. Para las pruebas de integración, me aseguraría de probar la interacción entre diferentes componentes y servicios para garantizar su funcionamiento conjunto. Utilizaría herramientas como **Jasmine y Karma** o **Jest** para las pruebas unitarias y herramientas como **Cypress** para las pruebas de integración.

¿Qué herramientas y técnicas utilizarías para garantizar la calidad del código en un proyecto Angular?

Utilizaría herramientas de análisis estático de código como **ESLint** para identificar y corregir problemas de estilo y errores de sintaxis. **Prettier** para asegurar que todos los desarrolladores usamos el mismo formateo. **Husky** para activar los Linter y Formateadores y asi eviar un commit o push deficiente. Además, seguiría las mejores prácticas de codificación y aplicaría patrones de diseño como el patrón Presenter para un código limpio y modular. Realizaría revisiones de código regularmente y fomentaría la colaboración entre el equipo para asegurar una alta calidad del código.

Arquitectura y diseño:

¿Qué principios de diseño y arquitectura aplicarías en un proyecto Angular de tamaño medio?

En un proyecto de tamaño medio, aplicaría principios de diseño como la separación de preocupaciones y la modularidad para mantener un código limpio y organizado. Utilizaría el patrón de arquitectura de componentes para descomponer la aplicación en partes más pequeñas y reutilizables, lo que facilita el mantenimiento y la escalabilidad.

¿Cómo estructurarías los módulos y componentes en una aplicación Angular para facilitar la escalabilidad y el mantenimiento?

Para facilitar la escalabilidad y el mantenimiento, estructuraría la aplicación en módulos funcionales y reutilizables que agrupen componentes relacionados. Utilizaría módulos principales para cargar módulos secundarios de manera lazy loading, lo que mejora el rendimiento y la experiencia del usuario. Además, aplicaría el principio de la separación de preocupaciones para garantizar que cada componente tenga una responsabilidad clara y definida.

PROGRAMADOR SENIOR

Seguridad:

¿Qué medidas tomarías para garantizar la seguridad en una aplicación Angular a gran escala?

- Autenticación y autorización: Implementar un sistema de autenticación y autorización robusto, como Firebase Auth, Okta o Auth0.
- Validación de entrada: Validar todas las entradas de usuario para evitar ataques de inyección de código (XSS) y SQL (SQLi).
- Sanitización de salida: Sanitizar todas las salidas antes de mostrarlas al usuario para evitar ataques de Cross-Site Scripting (XSS).
- Almacenamiento seguro de datos: Almacenar las contraseñas y otros datos confidenciales de forma segura, utilizando técnicas como el hash y el cifrado.
- Protección contra ataques CSRF: Implementar medidas de protección contra ataques CSRF, como tokens anti-CSRF.
- Mantener el software actualizado: Mantener actualizado el framework Angular y las bibliotecas de terceros para corregir vulnerabilidades de seguridad.

¿Cómo implementarías la autenticación y autorización de manera segura en una aplicación Angular?

Para implementar la autenticación y autorización de manera segura en una aplicación Angular, utilizaría técnicas como JSON Web Tokens (JWT) para manejar la autenticación de usuarios. Esto implicaría un flujo de inicio de sesión seguro, que incluya la verificación de credenciales del usuario y la generación de un token JWT firmado. Además, implementaría un sistema de roles y permisos para la autorización, asegurándome de que cada acción dentro de la aplicación esté protegida según los roles asignados al usuario, por ejemplo:

- **Firebase Auth:** Usar Firebase Auth para la autenticación y autorización, aprovechando su integración con Google Cloud Platform.
- **Okta o Auth0:** Implementar Okta o Auth0 como proveedores de identidad externos, con Single Sign-On (SSO) y soporte multifactor.

Arquitectura y diseño:

¿Cómo diseñarías una arquitectura robusta y escalable para una aplicación Angular compleja?

Utilizaría el patrón de arquitectura de **MicroFrontEnd**. Esto implica dividir la aplicación en proyectos independientes y desacoplados que puedan escalar de forma independiente. Además, cada MicroFrontEd implementaría una clean arquitectura, por ejemplo, la arquitectura hexagonal. Tambien usaría la arquitectura mono repo o multi-repo para la gestions de cambios y dependencias de las librerías.

Para crear MicroFrontEnd puedo usar **Single SPA** o **ModuleFederation de WebPack**, son los más populares

¿Cuáles son las mejores prácticas para estructurar y organizar un proyecto Angular grande y complejo?

Carpeta por módulo: Organizar el código en carpetas por módulo, con subcarpetas para componentes, servicios, estilos y pruebas.

Nomenclatura clara: Utilizar una nomenclatura clara y consistente para los nombres de archivos, clases, variables y funciones.

Documentación: Documentar el código de la aplicación de forma clara y concisa.

Además debo guiarme a la arquitectura de proyecto, líneas arriba respondí que usaría la arquitectura hexagonal, entonces organizaría las carpetas principales, por “infraestructura”,” aplicación” y “dominio”.

¿Cómo integrarías Angular con tecnologías como WebSockets, GraphQL o WebRTC en un proyecto a gran escala?

WebSockets: Usar la biblioteca **@stomp/ng2-stompjs** para la comunicación en tiempo real con WebSockets.

GraphQL: Implementar **Apollo GraphQL** para realizar consultas y mutaciones en un servidor GraphQL.

WebRTC: Usar la biblioteca **peerjs** para la comunicación de audio y video en tiempo real.

¿Qué estrategias utilizarías para gestionar la integración con una API RESTful o GraphQL en una aplicación Angular compleja?

Servicio HTTP: Usar el servicio HttpClient de Angular para realizar peticiones HTTP a la API.

Encapsulamiento en un servicio: Encapsular la lógica de la API en un servicio dedicado, para facilitar su reutilización y mantenimiento.

Normalización de datos: Normalizar los datos de la API para que sean consistentes con el modelo de datos de la aplicación.