I. Evaluación General de Expertise

• ¿Cuántos años de experiencia tienes programando profesionalmente?

7 años

• ¿Cuál ha sido el proyecto más desafiante en el que has trabajado y cuál fue tu rol?

Desarrollo de Backend y Frontend de wizard para marcas, tiendas, productos y manejo de órdenes, admin para aprobaciones de tiendas y visualización y actualización de ordines, y el rol que tuve fue el de líder de equipo

• ¿Cómo te mantienes actualizado sobre nuevas tecnologías?

Suscribiéndome a los newslleters de las tecnologías, siguiendo influencers dedicados a tech, y viendo cursos en Udemy o Youtube.

• ¿Has trabajado en equipos multidisciplinarios? ¿Cómo manejas la colaboración?

Si, siempre trato de manejar con cordialidad y respeto, tratando de aportar y apoyar en lo que pueda a mis compañeros, siempre asegurándome de tener clara cuales son las expectativas, y tambien solicitando ayuda cuando la necesito

• ¿Cómo manejas situaciones en las que no tienes experiencia previa con una herramienta

requerida?

Leo la documentación oficial de la herramienta, que me sirve con una guía inicial, luego pregunto si hay alguien del equipo con experiencia en la herramienta, y trato de que me de una guía para saber que hacer, y luego también trato de buscar desarrollos ya hechos en la herramienta, de ser el primero trato de ver cursos en youtube o leer foros como stackOverflow entre otros.

II. Frontend: React, Next.js, TypeScript, JavaScript + Redux

• ¿Qué ventajas ofrece Next.js respecto a CRA (Create React App)?

Next.js te da ya todo un esquema con las herramientas necesarias para facilitar el desarrollo de una app en React, en cambio CRA te crear el template con React puro lo que hace que si quieres algún comportamiento que no está dentro del core inicial de React lo tengas que instalar y configurar.

• ¿Cómo organizas tu estructura de carpetas en una aplicación Next.js?

- App
- Pages
  - Alguna pagina
  - Otra pagina
- Components
- Hooks
- Lib
- Styles
- Types
- Tsconfig

• ¿Cómo tiparías correctamente los props en un componente funcional?

```
interface someBtnProps<T> {
    label: string
    onClick: () => Promise<T>
    disabled: boolean
}
```

• ¿Cómo manejarías el estado global de una app sin Redux?

Dependiendo del tamaño de la app si es pequeña o mediana React Context sino usaría una librería como zustand o jotai

• ¿En qué casos usarías JavaScript puro en lugar de React?

Cuando se tiene una app pequeña o estática que no requiera tanta complejidad a nivel de reactividad del virtual dom o cuando se quiera priorizar al máximo rendimiento y compatibilidad con los navegadores más antiguos

• ¿Que es un manejador de estado y cómo funciona?

Es una herramienta en la cual esta centralizado los datos que afectan al UI, donde para que cada una de las partes puedan estar escuchando los cambios se tiene que suscribir, y para actualizarlo se hace por medio de funciones, esto se almacena en un solo lugar para hacer las eficiente y que a la hora que se suscriban el UI reaccione o tenga los comportamientos que queremos automáticamente

III. Firebase

• ¿Cómo configurarías reglas de seguridad en Firestore para usuarios con diferentes

roles?

No he trabajado directamente con Firebase, pero entiendo que Firestore permite de alguna forma definir la reglas a seguir en cuanto a la seguridad se refiere basados en la autenticación del usuario y en los datos que nosotros definamos como relevantes para la autenticación como por ejemplo el rol, donde se pueden definir acceso a acciones dependiendo del rol que se tenga.

• ¿Cuál es tu experiencia utilizando Firebase Authentication?

No he usado Firebase Authentication, pero si he implementado autenticación con JWT y OAuth2 en otras aplicaciones

• ¿Has utilizado Firebase Storage y Functions? Describe un caso real.

No he utilizado Firebase Storage y Functions, pero entiendo que es parecido a AWS S3, en este caso yo si he utilizado AWS S3 para el almacenamiento de imágenes de perfil y de productos en proyecto anterior donde se configuro que cada vez que se suba el archivo se revisar en el backend la metadata de la imagen para ver si es una repetida, si ya existe no se hace nada, sino existe se inserta, también se agregó una opción para eliminar imágenes y así mantener limpio el bucket de S3.

• ¿Cómo manejarías una estructura de base de datos escalable para una app con usuarios,

productos y órdenes?

En mi caso lo que haría seria crear 3 entidades, Users, products y orders, donde lo que haría para no depender de joins o data de otras tablas es que guardaría los datos relevantes e indispensables por ejemplo lo que haría seria cuando se cree una orden en lugar de solo guardar el id del usuario o los productos comprados, lo que haría seria guardar por ejemplo del usuario su id, su nombre, su correo y su dirección, en el caso de los productos guardaría su id su nombre, su precio e imagen si tiene, esto lo haría así pare reducir el costo de las consultas por que estaría quedando todo en un mismo json por lo que tanto del lado del BE como de la BD no tendría que hacer operaciones complejas a la hora de mostrarme un listado o un detalle de una orden.

IV. Shopify (Liquid + Integraciones JS)

• ¿Qué es Liquid y cómo funciona en Shopify?

Liquid es el lenguaje de las plantillas que usa Shopify pare renderizar contenido dinámico, el servidor interpreta las plantillas y sustituye las variables dinámicas por datos reales

• ¿Cómo modificarías una sección reutilizable en un tema de Shopify?

Buscaría la forma de tener en cuenta los casos de uso actuales, y cuales son los que se quieren actualizar o crear, esto con el objetivo de no romper el como se ve actualmente y esto nos ocasione un problema a la hora de visualizar en los lugares donde este implementada la sección, una vez hecha la validación comenzaría con la edición de la plantilla

• ¿Cómo inyectarías funcionalidades externas con JavaScript en una tienda Shopify?

No he trabajado con Shopify, pero entiendo que es parecido a Woocommerce o Magento, en este caso se haría por medio de un script de javascript donde se inyecta html base de la plantilla para que pueda ser usado.

• ¿Qué experiencia tienes con personalización de checkouts o carritos?

No he trabajado directamente con Shopify ni he personalizado carritos o checkouts en esa plataforma, pero entiendo que el checkout en Shopify tiene restricciones de personalización salvo en planes como Shopify Plus. He trabajado con conceptos similares en plataformas como WooCommerce y Magento, donde personalicé la experiencia de carrito y flujo de compra, así que puedo adaptarme rápidamente a Shopify.

V. Backend con Node.js

• ¿Cómo estructuras un proyecto Node.js que escale bien?

Para estructurar un proyecto Node.js escalable separo las responsabilidades en carpetas específicas:

- **controllers**: manejan las peticiones HTTP y responden al cliente.
- **services**: contienen la lógica de negocio para mantener separados los controladores de la lógica central.
- **models**: definen los esquemas de la base de datos.
- **repositories**: actúan como capa intermedia para las operaciones con la base de datos, facilitando cambios en la implementación del almacenamiento.
- **common**: agrupa elementos reutilizables como interfaces, enums y constantes, organizados también por tipo.
- **utils**: contiene funciones auxiliares que pueden ser usadas en cualquier parte del proyecto.

Esta estructura modular permite que el proyecto sea más mantenible, facilita las pruebas unitarias y la integración de nuevas funcionalidades a medida que la app crece

• ¿Has creado middleware personalizado? ¿Puedes dar un ejemplo?

Si, lo que hacia este middleware es revisar si el usuario que estaba haciendo la petición al endpoint tenia acceso al mismo haciendo uso de una estructura de roles donde tenían un campo que definia por rol a que rutas dentro del backend podía hacer peticiones y también a que verbo http de esa ruta tenia acceso, entonces el middleware lo que hacia era destructurar la petición en url, entidad, verbo http para revisar si el usuario tenia acceso o no, entonces una vez hechas las validaciones si tenia acceso simplemente se dejaba pasar la petición, pero en caso de que no tuviera acceso se devolvía un error http 403 Forbiden.

• ¿Qué ORM has usado en tus proyectos? ¿Qué ventajas encuentras en uno como Prisma?

Los ORM's que utilizado son:

1. TypeORM
2. Mongoose
3. Sequelize

Las ventajas que tiene prisma es que te permite tener un ORM fuertemente tipado y moderno donde se prioriza la estructura a la flexibilidad, ideal para evitar errores de capa 8 tanto en tipado como a la hora de hacer las consultas, además de que obliga al desarrollador a cumplir con el contrato de que es lo que espera y también tiene la facilidad de que la ser un ORM fuertemente tipado también el dev sabe que debe esperar a la hora de hacer una operación a base de datos.

• ¿Cómo implementarías autenticación y autorización en una API REST?

- Autenticacion:
  - JWT: Usaría jwt para identificar que es un usuario al inicar sesión.
  - Contraseña: Esta seria almacenada por medio de un hash para que solo el servidor backend pueda comprobar si es la misma por medio de un secret que solo tiene acceso el servidor backend.
- Autorizacion:
  - En cada endpoint validaría el token con un middleware. Donde el middleware se encargaría de revisar si el token es valido y si representa un usuario dentro de nuestra aplicación

• ¿Cómo manejarías errores globales en una aplicación Express o Nest.js?

Haria un interceptor global que reciba cada una de las respuesta a enviar en la cual tiene que detectar si es un error o no, en caso de ser un error debe revisar si es una instancia de error de nest.js o no, en caso de no ser una instancia de error de Nest.js entonces debe mappear todo el contexto necesario y enviarlo de formar clara y concisa retornando el código de error 500, en caso de ser una instancia de error de Nest.js debería retornar el error normal y de no ser un error solo retornar la respuesta.

VI. Buenas Prácticas y Testing

• ¿Cómo aseguras la calidad del código que entregas?

Siempre he usado el concepto de probar las cosas como queriendo que se arruinen eso me asegura lograr encontrar escenario no cubiertos, además de siempre seguir los patrones de diseño y las mejores practicas en cuanto a codificación se refiere.

• ¿Has implementado pruebas unitarias o de integración? ¿Qué librerías usas?

Si, las librerías que uso regularmente son:

1. Jest
2. Mocha
3. Nyc
4. Sinon
5. Chai
6. Supertest

• ¿Cómo configuras un pipeline de CI/CD básico para despliegue continuo?

1. Generaría ssh key para poder conectarme desde el repositorio hacia el servidor en cuestión
2. Crearía secretos para la llave ssh y la ip, para que no se puedan ver desde la ejecución del pipeline y así mantenerlo seguro
3. Configuraría el pipeline siguiendo los siguientes pasos:
    a. Fase de instalación en esta fase se instalan todas las dependencias necesarias para que se puedan compilar el proyecto.
    b. Fase de construcción en esta fase se hace la compilación del proyecto independientemente que sea para backend o frontend procurando que la compilación sucede en el pipeline y no en el servidor de destino, además aquí se podría incluir una fase de pruebas, donde se corran las UT'S o E2E para asegurarse que el codigo esta bien
    c. Fase de despliegue en esta fase copiaria la compilacion del proyecto a unas carpetas dentro del servidor donde deberian de ejecutarse
    d. Fase de ejecucion aqui ira la parte donde en caso de ser web se haria la configuracion correspondiente a nginx o apache o en caso de ser backend se usaria algunas de las librerias para ejecucion continua con generacion de logs como lo son forever o PM2

• ¿Qué estrategias sigues para mantener seguridad en aplicaciones web?

Validación y sanitización de datos: siempre valido la entrada del usuario en el backend para evitar inyecciones SQL/NoSQL y sanitizo la salida para prevenir XSS.

Autenticación y autorización robustas: uso tokens JWT con expiración corta y encripto las contraseñas con bcrypt. También implemento control de acceso basado en roles (RBAC) para restringir funcionalidades.

Protección contra ataques comunes: aplico medidas contra CSRF (tokens anti-CSRF), limitación de peticiones para prevenir ataques de fuerza bruta (rate limiting) y uso HTTPS para cifrar las comunicaciones.

Manejo seguro de errores: no expongo detalles internos en las respuestas para evitar dar pistas a posibles atacantes.

Actualización de dependencias: reviso y actualizo periódicamente las librerías para evitar vulnerabilidades conocidas."