Backend · Docker · CI/CD

1️⃣ ¿Cómo se relacionan frontend, backend y base de datos dentro de una arquitectura web moderna?

Explica el rol de cada uno y cómo se comunican entre sí.

Pues el front es como un escaparate que hace peticiones http/https y sirve los datos traídos mediante apirest del backend y este sirviéndose de base de datos el back es como un puente que recoge los datos de base de datos los ordena filtra o maqueta para servírselos al front y este poder recibir lo que necesita para hacer una app funcional y que carge dinamicamente todo el contenido.

en resumen el front es lo que interactua con el usuario el back es la capa logica y l base de datos es el almacenamiento y todo mediante solicitudes HTTP al backend El backend procesa esas solicitudes, aplica la lógica de negocio y consulta la base de datos. La base de datos almacena los datos de forma persistente. El backend actúa como intermediario controlado entre el frontend y la base de datos, normalmente mediante APIs REST y formato JSON.

2️⃣ ¿Qué ventajas ofrece una API REST frente a otros modelos de comunicación entre sistemas?

Menciona al menos dos ventajas desde el punto de vista de escalabilidad o mantenimiento.

Una API REST ofrece varias ventajas frente a otros modelos de comunicación entre sistemas.
Una ventaja importante es que usa HTTP estándar, lo que permite que el frontend y el backend se comuniquen de forma simple mediante solicitudes y respuestas, normalmente usando JSON, que es un formato ligero y fácil de manejar. Esto hace que el sistema sea más fácil de mantener y modificar, ya que cada parte puede cambiar sin romper la otra mientras se mantenga la misma API.

Otra ventaja es que REST es stateless, es decir, cada solicitud es independiente y contiene toda la información necesaria. Esto permite escalar mejor el sistema, porque el backend puede manejar muchas solicitudes de diferentes clientes sin tener que guardar el estado de cada uno, facilitando el uso de múltiples servidores y mejorando el rendimiento.
En resumen, REST facilita la comunicación entre sistemas de forma estándar, escalable y fácil de mantener, separando el frontend del backend y permitiendo que ambos evolucionen de forma independiente.

3️⃣ ¿Qué implica diseñar un backend desacoplado del frontend?

Explica por qué es importante en proyectos grandes o a largo plazo.

Diseñar un backend desacoplado del frontend implica que el backend funciona de forma independiente y no depende directamente de cómo esté hecho el frontend. La comunicación entre ambos se hace mediante una API REST usando solicitudes HTTP y normalmente datos en formato JSON.
Esto significa que el backend se encarga solo de la lógica de negocio, procesar datos y comunicarse con la base de datos, mientras que el frontend se encarga de la interfaz y la interacción con el usuario. Ninguno necesita conocer la implementación interna del otro, solo cómo usar la API.

4️⃣ ¿Qué diferencias conceptuales existen entre contenedores Docker y máquinas virtuales?

Indica al menos dos diferencias claras a nivel de recursos o arquitectura.

Los contenedores Docker y las máquinas virtuales se diferencian principalmente en cómo usan los recursos y cómo están construidos a nivel de arquitectura.
Una diferencia clara es que las máquinas virtuales incluyen un sistema operativo completo con su propio kernel, por lo que consumen más recursos como memoria y CPU. En cambio, los contenedores Docker comparten el kernel del sistema operativo anfitrión, lo que los hace mucho más ligeros y eficientes.

5️⃣ ¿Qué significa que una aplicación sea *portable* y cómo ayuda Docker a conseguirlo?

Explica por qué esto es clave en entornos profesionales.

Que una aplicación sea portable significa que puede ejecutarse en diferentes entornos sin tener que modificar su configuración ni su código, independientemente del sistema operativo, servidor o máquina donde se ejecute.
Docker ayuda a conseguir esto porque permite empaquetar la aplicación junto con todas sus dependencias, configuraciones y entorno dentro de un contenedor. De esta forma, el contenedor incluye todo lo necesario para que la aplicación funcione igual en cualquier sitio, ya sea en el ordenador del desarrollador, en un servidor o en la nube.

6️⃣ ¿Qué es la integración continua (CI) y qué problemas evita en un equipo de desarrollo?

Relaciona la respuesta con errores de código o coordinación.

La integración continua (CI) es una práctica en la que los desarrolladores integran sus cambios de código en un repositorio compartido de forma frecuente, y cada vez que se hace un cambio se ejecutan automáticamente pruebas y verificaciones para comprobar que todo funciona correctamente.
Esto ayuda a detectar errores rápidamente, ya que si un desarrollador introduce un fallo, el sistema lo detecta de inmediato en lugar de descubrirlo mucho más tarde cuando el proyecto está más avanzado y es más difícil de arreglar.

7️⃣ ¿Qué es el despliegue continuo (CD) y qué requisitos debería cumplir un proyecto para poder usarlo con seguridad?

Habla de estabilidad, tests o control de versiones.

El despliegue continuo (CD) es una práctica en la que los cambios en el código se despliegan automáticamente a producción después de pasar una serie de pruebas y verificaciones, sin necesidad de hacerlo manualmente. Esto permite que las nuevas funcionalidades, mejoras o correcciones lleguen más rápido y de forma constante al entorno real.

Para poder usarlo con seguridad, el proyecto debe tener una buena estabilidad y una base de código bien organizada, ya que cualquier error podría desplegarse automáticamente y afectar a los usuarios.

8️⃣ ¿Por qué la automatización es un elemento clave en DevOps?

Explica cómo afecta al tiempo, a la calidad del software y al equipo.

La automatización es un elemento clave en DevOps porque permite ejecutar tareas repetitivas de forma automática, reduciendo el tiempo necesario para desarrollar, probar y desplegar una aplicación.
A nivel de tiempo, la automatización acelera procesos como la integración, los tests y el despliegue, evitando tener que hacerlo manualmente cada vez. Esto permite entregar cambios más rápido y de forma continua, mejorando la eficiencia del desarrollo.
A nivel de calidad del software, la automatización permite ejecutar tests y verificaciones de forma constante, lo que ayuda a detectar errores antes y evita que código con fallos llegue a producción. Esto hace que el sistema sea más estable y fiable.

9️⃣ ¿Qué diferencias existen entre un error detectado en fase de desarrollo y uno detectado en producción?

Explica las consecuencias técnicas y organizativas.

Un error detectado en fase de desarrollo es mucho más fácil y rápido de solucionar porque el código aún está en un entorno controlado y no afecta a los usuarios reales. El desarrollador puede corregirlo directamente, probar la solución y asegurarse de que todo funciona correctamente antes de desplegarlo. Además, el impacto es bajo porque no afecta al funcionamiento del sistema en producción.
En cambio, un error detectado en producción tiene consecuencias más graves, ya que afecta directamente a los usuarios y al funcionamiento real de la aplicación. Puede provocar fallos, pérdida de datos, mal funcionamiento o una mala experiencia de usuario. A nivel técnico, solucionarlo es más complejo porque hay que analizar el problema en un entorno real, corregirlo sin romper otras partes del sistema y desplegar una nueva versión.

🔟 ¿Por qué es importante separar la configuración del código en un backend?

Da ejemplos conceptuales (entornos, seguridad, mantenimiento).

Separar la configuración del código en un backend es importante porque permite cambiar valores y comportamientos del sistema sin tener que modificar el código fuente. Esto hace que la aplicación sea más flexible, segura y fácil de mantener.
Por ejemplo, en diferentes entornos como desarrollo, testing y producción, la aplicación puede usar distintas configuraciones, como diferentes bases de datos, URLs o credenciales. Si la configuración está separada, solo hay que cambiar las variables de entorno sin tocar el código.