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.

El frontend solicita datos vía HTTP, el backend procesa la lógica y seguridad, y la base de datos garantiza la persistencia. Se comunican mediante APIs (JSON), permitiendo que cada capa escale y se mantenga de forma independiente.

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.

Ofrece alta escalabilidad al ser stateless (sin estado en el servidor) y gran facilidad de mantenimiento por su interfaz estándar. Permite desacoplar el cliente del servidor, facilitando actualizaciones sin romper la compatibilidad entre sistemas.

3️⃣ ¿Qué implica diseñar un backend desacoplado del frontend?

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

Consiste en separar totalmente la lógica de datos de la interfaz, permitiendo que una misma API sirva a múltiples clientes (web, móvil, IoT). Es vital para la escalabilidad y para que equipos especializados trabajen en paralelo sin bloquearse.

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

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

Las VM virtualizan hardware y cargan un SO completo, siendo pesadas y lentas. Docker virtualiza a nivel de SO compartiendo el kernel del host, lo que lo hace mucho más ligero, rápido y eficiente en el uso de recursos

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

Explica por qué esto es clave en entornos profesionales.

La portabilidad asegura que la app funcione igual en cualquier entorno. Docker lo logra empaquetando el código y sus dependencias en imágenes inmutables, eliminando errores de configuración y el clásico "en mi máquina sí funciona".

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.

Automatiza la fusión de código y la ejecución de tests tras cada cambio en el repositorio. Evita conflictos de integración masivos y detecta errores de forma temprana, garantizando que el código base siempre sea estable y funcional.

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.

Automatiza el lanzamiento a producción tras superar todas las pruebas de calidad. Requiere una cobertura de tests muy alta y sistemas de rollback automáticos para asegurar la estabilidad del servicio ante cualquier fallo inesperado.

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

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

Es clave para eliminar errores humanos, reducir tiempos de entrega y garantizar procesos repetibles. Permite que el equipo se enfoque en desarrollar nuevas funcionalidades en lugar de perder tiempo en tareas operativas o manuales.

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

Explica las consecuencias técnicas y organizativas.

En desarrollo el error es barato y fácil de corregir sin impacto. En producción genera costes críticos, posible pérdida de datos y daño reputacional, exigiendo una intervención de emergencia que interrumpe el flujo de trabajo del equipo.

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

Da ejemplos conceptuales (entornos, seguridad, mantenimiento).

Aumenta la seguridad al no exponer credenciales en el repositorio y permite usar la misma imagen de software en distintos entornos. Se gestiona mediante variables de entorno, facilitando cambios de configuración sin necesidad de recompilar.