2) Preguntas técnicas
Responder brevemente las siguientes preguntas
A. ¿Qué diferencia hay entre una clase abstracta y una interfaz?
Las diferencias son:

1. La clase abstracta no soporta herencia múltiple y la interfaz si.
   por ejemplo:
   class Perro extends AMamifero { …} //Solo permite heredar de la clase abstracta AMamifero
   class Perro implements IVertebrado,ICarnivoro {..} //La clase perro puede implementar el comportamiento de las interfaces IVertebrado e ICarnivoro
2. La clase abstracta puede definir metodos abstractos o tener metodos implementados y que sean sobreescritos por las subclases que heredan de ella. Mientras que en una interfaz implícitamente todos los métodos son abstractos y estoy obligado a implementarlos en las clases que implementen la interfaz
3. La clase abstracta tiene constructor y la interfaz no.
4. La clase abstracta puede definir variables y constantes, la interfaz no, solo define la interfaz (mensajes que entiende)
5. La clase abstracta puede tener modificadores de acceso (private, protected,public). La interfaz tiene todo sus mensajes públicos.
6. La clase abstracta puede tener miembros estáticos (variables o métodos de clase), la interfaz no

Creo que a grandes rasgos estas son las principales diferencias

B. ¿Para qué sirve el web.xml de un servlet container como Tomcat?
Desconocía para qué se usa, pero investigando veo que es para asignar las URL en una aplicación web con backend en JAVA. Abajo un fragmento de lo que leí:
Las aplicaciones web de Java usan un archivo descriptor de implementación para determinar cómo se asignan las URL a los servlets, qué URL requieren autenticación y más información. Este archivo se llama web.xml y se encuentra en el WAR de la aplicación dentro del directorio WEB-INF/. web.xml es parte del estándar del servlet para aplicaciones web.

C. ¿Cuáles son las diferencias entre servicios REST y SOAP?
Ambos son protocolos de comunicación entre dos componentes a través de un servicio web dentro de la arquitectura SOA.
SOAP se comunica utilizando un archivo XML con los atributos que desea comunicar, es más robusto y menos performante que REST. REST se apoya en el protocolo HTTP para el envío de mensajes (GET,POST,PUT, DELETE, etc) y utiliza sus códigos de respuesta (200, 404, 500, etc) y puede enviar practicamente cualquier dato (imagenes, mult

D. ¿Qué partes de una aplicación Web se ejecutan client-side y qué partes server side?
Entiendo que la pregunta va enfocada a que la vista (código HTML, CSS y Javascript) se ejecutan del lado del cliente y que del lado del servidor se ejecutan las validaciones contra la base de datos o del dominio de la aplicación, por ejemplo si el login es correcto se levanta

una vista con la pantalla principal de la aplicacion (lado del cliente), pero previo a esto del lado del servidor se tuvo que cargar en un repositorio de la base de datos los datos los usuarios (acceder a la base de datos)

E. ¿Qué significa IoC (Inversion of Control) y para qué sirve?
Es un patrón de diseño de sofrware o una buena práctica. En vez de tener un GOD object que tiene toda la responsabilidad de mi dominio, éste objeto delega la construccion y manejo de objetos, como por ejemplo inyectando dependencias (una interfaz) a través de setters. Ejemplos de esto se pueden ver con patrones como abstract Factory, factory method

```
class Persona {
	TipoContribuyente tipo;
	String nombre;
	String apellido
	void pagarImpuesto(Impuesto impuesto) {
		tipoContribuyente.pagarImpuesto(impuesto);
	}
```

En este ejemplo la persona no tiene que implementar cómo pagar el impuesto segun su tipo (monotributista, responsable inscripto, consumidor final) sino que la lógica de cómo se paga el impuesto lo sabe el tipo (que es una interfaz TipoContribuyente) de la que implementan Monotributista, ResponsableInscripto,ConsumidorFinal y cada uno implemente el método pagarImpuesto(Impuesto impuesto) según la lógica que tenga en el dominio.

G. ¿Qué diferencia hay entre herencia y composición?

La herencia marca una relación entre clases (es estática), la superclase tiene características más generales mientras que la subclase toma comportamiento específico y cuando es necesario lo redefine. En la composición no hay una jerarquía de clases, sino que intervienen dos instancias: una conoce a la otra y le envía mensajes. La herencia es un mecanismo de diseño de un solo tiro, es decir, si heredo una clase de una superclase, por ejemplo Perro hereda de Animal, la clase perro no va a poder cambiar su contrato. En cambio con la composición, puedo redefinirle comportamiento como en el ejemplo que di mas arriba de Persona y tipo de contribuyente. La composición es un mecanismo más flexible a la hora de diseñar objetos.

H. ¿Qué diferencia hay entre una librería y un framework?

Una biblioteca es código que puedo reutilizar que generalmente puedo importar a mi proyecto o ya viene disponible en en el lenguaje de programación, por ejemplo manejar fechas o arrays o strings. Una biblioteca me expone un contrato y yo sé que puedo utilizar un método X sabiendo qué esperar de él pero no cómo está implementado, por ejemplo strLen(cadena). Un framework es algo más abstracto, me provee de bibliotecas y además de un marco de trabajo que alguien más ya pensó y yo puedo utilizarlo en mi proyecto, por ejemplo JUnit para hacer el testeo unitario de mi proyecto.