Juan David Rodriguez Cabrera
Ingeniero de sistemas
Celular: 301 200 1162

1. **¿Qué es la reflexión en Java y cómo la utilizas en el desarrollo de aplicaciones?**

La Reflexión en Java te permite explorar y manipular la estructura interna de clases y objetosa lo largo del tiempo. Esta característica es muy útil en bastantes áreas del desarrollo de aplicaciones.

Marco de inyección de dependencias: herramientas como Spring utilizan espejos para automatizar la creación y gestión de objetos. Esto permite a los desarrolladores concentrarse en la lógica de negocio sin tener que preocuparse por cómo están conectados los componentes de la aplicación.

Formato y actualización: cuando se desea convertir un objeto a un formato que se pueda almacenar o exportar (como JSON o XML), se facilita el acceso a datos contenidos en un objeto. De esta forma se podrá realizar un cambio sin conocer todos los detalles de implementación.

Automatización de pruebas: marcos como JUnit aprovechan la reflexión para definir y administrar automáticamente métodos de prueba. Esto libera a los desarrolladores de la carga de ejecutar manualmente cada prueba, lo que aumenta la eficiencia de las pruebas.

API dinámicas: Reflexión le permite crear APIs que pueden funcionar con diferentes tipos de objetos sin conocer las propiedades de los objetos en el momento de la compilación. Esto es bueno para situaciones que requieren flexibilidad y adaptabilidad.

Notas y patrones de diseño: el espejo se utiliza para administrar notas personalizadas e implementar patrones de diseño como Factory. Esto aumenta la flexibilidad de su código al permitirle crear objetos de forma dinámica y estructural.

2. **¿Cuál es la diferencia entre @Component, @Service y @Repository en Spring Boot?**

@Component: Es una anotación de carácter general destinada a identificar una clase como componente Spring y se emplea para determinar cualquier clase que necesite ser administrada por el contenedor de Spring, por lo que no posee un objetivo concreto.

@Service: Se refiere a una especialización de @Component, lo que señala que la categoría que lo comprende incluye lógica empresarial. Además, puede ser descrito como un servicio en la esfera empresarial y puede acoger transacciones.

@Recipiente: También se encuentra en la capa de acceso a datos, siendo una especialización de @Component. Se emplea para ofrecer esa encapsulación en la gestión de datos y

Juan David Rodriguez Cabrera
Ingeniero de sistemas
Celular: 301 200 1162
frecuentemente convierte las excepciones enviadas desde la base de datos en datos.

3. **¿Cómo se puede configurar un contenedor Docker para exponer un puerto específico?**

Para mostrar un puerto concreto en un contenedor Docker, tienes la opción de ejecutar el contenedor con la opción -p. Asi:

docker run -p 8080:80 nombre_imagen

4. **¿Cómo implementarías la autenticación basada en token JWT en una aplicación SpringBoot?**

Implementación de autenticación basada en JWT:

Integración: agregue la integración necesaria a pom.xml para Spring Security yJWT.
Configuración de seguridad
: corrija Spring Security para interceptar solicitudes y validar JWT en filtros de autenticación.
Generador de tokens: cree un servicio que genere tokens JWT en la entrada.
Verificación de firma: aplique un filtro de verificación de firma a cadasolicitud
al área protegida.
Almacenamiento: puede almacenar el token en
el lado del usuario (almacenamiento local o cookie) y pasarlo al encabezado
de Autorización en la solicitud.

5. **¿Cómo implementarías la tolerancia a fallos en una aplicación Spring Boot?**

Puedes implementar la tolerancia a fallos usando:

Circuit Breaker: use bibliotecas como Resilience4j o Spring Cloud Circuit Breaker
para evitar que el sistema falle cuando fallan los servicios externos.
Retries: establezca una política de reintento para reintentar operaciones fallidas.

Fallback: define un método de respaldo que se ejecutará cuando falla la función principal.

6. **¿Cuál es la diferencia entre una conexión JDBC y una conexión JNDI?**

Conectividad de bases de datos Java (JDBC): una API que permite a Java interactuar con bases de datos. Se utiliza para crear relaciones, ejecutar consultas y obtener resultados precisos de bases de datos configuradas en el proyecto.
Interfaz de directorio y nombres de Java (JNDI): proporciona una forma de acceder a recursos, como conexiones de bases de datos. JNDI puede recuperar y acceder a recursos

Juan David Rodriguez Cabrera
Ingeniero de sistemas
Celular: 301 200 1162

configurados en un servidor de aplicaciones, como recursos de conexión, sin gestionar los datos de conexión.

7. **¿Qué es el patrón DAO (Data Access Object) y cuál es su propósito en el desarrollo deaplicaciones Java?**

El patrón DAO es un patrón de diseño que proporciona una forma abstracta de acceder a datos (como bases de datos). Su propósito es separar la lógica empresarial de la lógica de entrada de datos. Esto le permite cambiar la implementación de la persistencia (por ejemplo, de JPA a JDBC) sin afectar la lógica de negocio.

8. **¿Qué es la anotación @Transactional en Spring y cómo la utilizarías en un método deservicio?**

La anotación @Transaction se utiliza para indicar que un método debe ejecutarse dentro de una transacción. Si ocurre un error, se revertirán todos los cambios realizados en la base de datos durante la transacción. Generalmente se utiliza en métodos de servicio
que implican operar múltiples bases de datos. Por ejemplo:

```
@Service
public class MiServicio {
  @Transactional
  public void miMetodo() {
    // Operaciones de base de datos
  }
}
```

9. **¿Qué es HATEOAS (Hypermedia as the Engine of Application State) y cómo se aplica en eldesarrollo de APIs RESTful?**

HATEOAS es un principio de diseño que permite a los clientes interactuar con una aplicación a través de Enlaces, en vez de tener las URL de antemano, el cliente que consume puede descubrir las diferentes acciones que hay disponibles mediante la navegación de los enlaces proporcionados en las respuestas de las API

Esto se puede implementar en Spring Boot agregando un enlace al DTO que representanalgún recurso

Juan David Rodriguez Cabrera
Ingeniero de sistemas
Celular: 301 200 1162

10. ¿Cuándo elegirías una solución NoSQL sobre una base de datos relacional en unaaplicación Java Backend?

Eliges NoSQL cuando:

Necesitas escalar horizontalmente.
Los datos están semiformateados o no formateados.
por Alta disponibilidad sin mucha latencia.
No necesitas operaciones ACID estricta y se puede tolerar consistency.La flexibilidad del esquema es importante y cambia frecuentemente.

11. ¿Qué son los métodos de referencia (method references) en Java 8 y cómo se utilizan enla programación funcional?

Los métodos directos son atajos para expresar expresiones lambda que llaman a métodos existentes. Se utilizan para mejorar la legibilidad del código. Hay cuatro tipos de métodos para referencia:

Referencia a un método estático.
Referencia a un método de instancia de un objeto particular. Referencia a un método de instancia de un objeto de un tipo específico.Referencia a un constructor.

Un ejemplo de un puntero a un método estático:

```
List<String> nombres = Arrays.asList("Juan", "Ana", "Pedro");
nombres.sort(String::compareToIgnoreCase);
```

12. ¿Qué es la anotación @FunctionalInterface en Java y cuál es su propósito?

La anotación @FunctionalInterface se utiliza para indicar que la interfaz es una interfaz funcional, es decir, una única interfaz completamente definida. Esto permite que esta función se utilice como tipo de referencia para expresiones lambda y métodos de referencia. Si el formulario tiene más de un significado, el constructor arrojará un error.

```
@FunctionalInterface
public interface MiInterfaz {
  void metodoUnico();
}
```