Parcial 2

¿Cuál es una diferencia clave entre una aplicación monolítica y una basada en microservicios?

Los microservicios se pueden desplegar y escalar de manera independiente

Las monolíticas no pueden usar Spring Boot

Los microservicios requieren hardware dedicado

Las monolíticas solo funcionan en la nube

¿Qué es el ApplicationContext en Spring?

El contenedor central que gestiona los beans y sus dependencias

Un archivo que contiene las rutas de los controladores

El entorno de ejecución de la base de datos

El punto de entrada de una aplicación Spring Boot

¿Qué ocurre cuando una clase es anotada con `@Component` y está dentro del paquete escaneado?

Spring la registra como un bean en el contexto

Se ejecuta automáticamente en un hilo nuevo

Spring la ignora si no tiene un `@Bean`

Es tratada como una entidad persistente

¿Cuál es el ciclo de vida típico de un bean singleton en Spring?

Se crea una vez al iniciar la aplicación y se destruye al apagar el contexto

Se crea cada vez que se inyecta

Se crea al final de la ejecución y se destruye al comienzo

No se destruye nunca

¿Qué ocurre si dos beans del mismo tipo están disponibles y se intenta hacer una inyección sin calificador?

Se lanza una excepción por ambigüedad.

Se elige el primero definido.

No se inyecta ninguno.

Spring crea uno nuevo.

¿Qué anotación se utiliza para mapear un parámetro de ruta a un argumento del método?

`@PathVariable`

`@RequestParam`

`@RequestBody`

`@Value`

¿Qué tipo de inyección permite usar todos los beans de un mismo tipo como colección?

Inyección de una colección con `@Autowired List<Tipo>`

Inyección por constructor con `@Qualifier("list")`

Inyección por atributo usando `@Autowired private Set<Tipo>`

No es posible inyectar múltiples beans

¿Qué ventaja tiene usar interfaces en lugar de clases concretas al inyectar dependencias?

Facilita el cambio de implementaciones y la escritura de tests

Mejora la compatibilidad con Lombok

Permite usar anotaciones como `@PostConstruct`

Acelera la ejecución de la aplicación

¿Qué ocurre si un atributo anotado con `@Autowired` no puede ser satisfecho por ningún bean del contexto?

Se lanza una excepción en tiempo de ejecución

Se ignora el atributo sin errores

Spring crea una instancia vacía del objeto

El contexto se vuelve a escanear automáticamente

¿Qué sucede si una clase anotada con `@Entity` no tiene un constructor sin argumentos?

Hibernate no puede instanciarla al leerla desde la base de datos

Spring ignora la entidad

La aplicación compila pero no la detecta como bean

No afecta, ya que Spring usa reflexión

Preguntas parcial 2

1. ¿Cuál es la función del comando ng serve?
   Iniciar un servidor de desarrollo local para ejecutar y probar la aplicación
   Compilar el código TypeScript de la aplicación a JavaScript
   Monitorear los cambios en los archivos y recargar automáticamente la página
   Construir y optimizar la aplicación para producción
   Generar nuevos componentes
   Ninguna es correcta
2. ¿Cuál es la principal diferencia en la creación de un componente tradicional frente a un
   Standalone Component (Angular 17+)?
   Los Standalone Components no requieren declararse en un módulo
   Se utiliza el mismo comando ng generate component, pero añadiendo la opción --
   standalone para los autónomos
   Los componentes tradicionales se crean manualmente, mientras que los autónomos usan
   CLI
   Los Standalone Components no tienen archivo HTML
   Los componentes tradicionales no tienen archivo TypeScript
   Ninguna es correcta
3. ¿Cuáles son algunas de las ventajas de los Standalone Components (Angular 17+)
   mencionadas en el apunte?
   Elimina la necesidad de declarar componentes en módulos
   Reduce archivos de configuración y boilerplate
   Simplifica el Lazy Loading en rutas

Mejora la reusabilidad al ser autocontenidos
Son incompatibles con la Inyección de Dependencias
Todas son correctas
4. ¿Cuáles son algunos mecanismos mencionados en el apunte para implementar la
asincronía?
Callbacks

Promesas (Promises)
Async/Await
Ejecución secuencial de tareas
Sincronización automática de datos
Todas son correctas
5. En el contexto de Angular y RxJS, ¿qué es un Observable?
Un objeto que representa una secuencia de datos asíncronos
Una forma de observar (escuchar) cambios y reaccionar a ellos asincrónicamente
Puede emitir varios valores en una secuencia
Una función que se ejecuta cuando se completa una operación asíncrona
Un método para actualizar el DOM directamente
Todas son correctas
6.¿Por qué es importante desuscribirse de un Observable cuando ya no se necesita
observar sus valores?
Para evitar pérdidas de memoria
Para prevenir problemas de rendimiento
Para asegurar que el observable se complete
Solo es necesario para Observables creados manualmente
HttpClient lo hace automáticamente en todas las situaciones
Ninguna es correcta
7. ¿Cómo se configura HttpClient en una aplicación Angular, según el apunte?
Mediante inyección de dependencia
Utilizando la función auxiliar provideHttpClient() en la configuración de proveedores
Se configura automáticamente al instalar Angular CLI
Requiere modificar archivos de configuración del servidor web
Se configura en el archivo package.json
Ninguna es correcta
8.¿Cuáles son algunas de las etapas o eventos del ciclo de vida de un componente Angular
mencionadas en el apunte?
ngOnChanges
ngOnInit
ngOnDestroy
ngAfterCreation
ngBeforeUpdate

Todas son correctas
9. Según el apunte, ¿cuáles son los tres tipos principales de Directivas en Angular?
Directivas estructurales
Directivas de atributos
Directivas personalizadas
Directivas de componentes
Directivas de servicio
Ninguna es correcta
10.¿Qué nuevas directivas estructurales se introducen en Angular 18 para reemplazar y
modernizar *ngIf, *ngFor y *ngSwitch?
@if
@for
@switch
ngIf
ngFor
Todas son correctas