# Prueba FWK Spring Boot

## Preguntas teóricas

### **Pregunta 1** (0,1 puntos)

¿Cuál es el nombre del *bean* definido en la siguiente clase de configuración? Selecciona una única respuesta.

```
@Configuration
public class ApplicationConfig {

	@Autowired
	private DataSource dataSource;

	@Bean
	ClientRepository clientRepository() {
		ClientRepository accountRepository = new JpaClientRepository();
		accountRepository.setDataSource(dataSource);
		return accountRepository;
	}

}
```

1. *JpaClientRepository*
2. *jpaClientRepository*
3. *clientRepository*
4. Se definen dos *beans*: *datasource* y el repositorio

### **Pregunta 2** (0,1 puntos)

¿Cómo se auto-inyecta en un campo los *beans* de Spring por su nombre? Selecciona una o más respuestas.

1. Con el atributo *name* de la anotación *@Autowired.*
2. Usando independiente de la anotación *@Qualifier*.
3. Usar las anotaciones *@Autowired* y *@Qualifier*.
4. Usar la anotación *@Autowired* y nombrar el campo con el nombre del *bean*.

## **Pregunta 3** (0,1 puntos)

¿Cuáles son las principales ventajas de usar interfaces cuando diseñamos servicios? Selecciona una o más respuestas.

1. Facilidad de *mocking/stubbing* del servicio.
2. Ser capad de utilizar la auto-inyección de Spring.
3. Poder hacer comprobación de dependencias.
4. Bajo nivel de acoplamiento del código.

## **Pregunta 4** (0,1 puntos)

Selecciona una o más respuestas correctas sobre el ciclo de vida de Spring.

1. El método anotado con *@PostConstruct* se llama después de la instanciación del *bean* y antes de la fijación de las propiedades de este.
2. El método *@PreDestroy* de un *bean* prototipo es llamado cuando el *bean* es recolectado por la basura.
3. El método *init()* declarado en el atributo *init-method* de un *bean* es llamado antes del *callback afterPropertiesSet* de la interfaz *InitializingBean*.
4. El método anotado con *@PostConstruct* se llama antes del método de devolución de llamada *afterPropertiesSet* de la interfaz *InitializingBean*.

## **Pregunta 5** (0,1 puntos)

Dada la siguiente clase de configuración, ¿cuáles son las afirmaciones correctas? Seleccione una o varias respuestas.

```
public class ApplicationConfig {

	private DataSource dataSource;

	@Autowired
	public ApplicationConfig(DataSource dataSource) {
		this.dataSource = dataSource;
	}

	@Bean(name="clientRepository")
	ClientRepository jpaClientRepository() {
		return new JpaClientRepository();
	}

}
```

1. Falta la anotación *@Configuration*.
2. Falta el constructor por defecto o sin carga.
3. El nombre de *@Bean* es ambiguo.
4. El ámbito de *@Bean* es *prototype*.

## **Pregunta 6** (0,1 puntos)

¿Cuál es la política de *rollback* por defecto en la gestión de transacciones?

1. Genera *rollback* para cualquier excepción.
2. Genera *rollback* en *RuntimeException*.
3. Genera *rollback* en *checked* capturadas.
4. Siempre termina en *commit*.

## **Pregunta 7** (0,1 puntos)

En comparación con las aplicaciones monolíticas, ¿cuáles son las ventajas de los microservicios?

1. El código base es fácil de entender.
2. Implican un sistema distribuido sencillo.
3. Despliegue más fácil.
4. Escalamiento más personalizado.

## **Pregunta 8** (0,1 puntos)

¿Cuál es el nombre del archivo de configuración del entorno por defecto de Spring Boot?

1. configuration.spring
2. application.yml
3. configuration.xml
4. application.properties
5. application.json

## **Pregunta 9** (0,1 puntos)

Seleccione las firmas de los métodos que coincidan con el siguiente *pointcut*:

```
execution(* com.test.service..*.*(*))
```

1. `void com.test.service.MyServiceImpl#transfert(Money amount)`
2. `void com.test.service.MyServiceImpl#transfert(Account account, Money amount)`
3. `void com.test.service.account.MyServiceImpl#transfert(Money amount)`
4. `void com.test.service.account.MyServiceImpl#transfert(Account account, Money amount)`
5. Ninguno de los anteriores.

## **Pregunta 10** (0,1 puntos)

¿Qué es verdadero sobre el módulo de pruebas de Spring?

1. Proporciona una capa de abstracción para los principales *frameworks mock* de código abierto
2. Proporciona la anotación *@Mock*.
3. Genera dinámicamente los objetos *mock*.
4. Todo lo anterior.
5. Ninguna de las anteriores.

## **Pregunta 11** (0,2 puntos)

¿Cómo se configura el Envoy encargado de la autenticación y autorización en Mercadona?

1. Mediante argumentos en la ejecución del contenedor.
2. Usando un fichero Open API 3 o Swagger versión 2.
3. Mediante variables de entorno.
4. Todo lo anterior.
5. Ninguna de las anteriores.

## **Pregunta 12** (0,2 puntos)

Selecciona las afirmaciones ciertas sobre la gestión de errores en el *starter* del consumidor de Kafka del FWK Spring Boot.

1. Por defecto no se realiza ninguna gestión del error, reintentando el procesamiento del mensaje hasta un máximo de 10 veces antes de pasar al siguiente.
2. No se puede definir e implementar una gestión de errores propia.
3. El modo DLQ *(dead letter queue),* en caso de producirse un error, se enviará el mensaje al tópico correspondiente.
4. Se puede configurar el modo ERROR, para que, en caso de error en el *listener*, se lance un estado de microservicio caído.
5. No existe gestión de errores en el starter del consumidor, solo en el *starter* del productor.

## **Pregunta 13** (0,2 puntos)

Selecciona las afirmaciones ciertas sobre la seguridad del microservicio.

1. Usando el *starter* de seguridad, por defecto se autentican todas las peticiones del adaptador API REST. En caso de necesitar autorización, por ejemplo, podemos anotar con *@PreAuthorize* el servicio correspondiente.
2. Usando el *starter* de seguridad, no se garantiza autenticación ni autorización.
3. La autenticación y autorización se deja en mano de una pieza externa al microservicio.
4. Usando el starter de seguridad, por defecto se autentican todas las peticiones del adaptador API REST. En caso de necesitar autorización se anota con *@RolesFwk* el servicio correspondiente.
5. Usando el *starter* de seguridad, se ofrece un servicio que accediendo al *SecurityContext* de Spring se proporcionan los *claims* del *token*.

## **Pregunta 14** (0,2 puntos)

Selecciona las afirmaciones ciertas sobre el versionado de la base de datos.

1. Desde el FWK Spring Boot, actualmente no se ofrece ninguna herramienta para la gestión de migraciones en base de datos relacionales.
2. Actualmente se ofrece la funcionalidad basándose en la librería de Flyway.
3. Actualmente se ofrece la funcionalidad basándose en la librería de Liquidbase.
4. Los scripts SQL de migración se clasifican en dos carpetas según su funcionalidad *rollback* y *migrations*.
5. Todos los ficheros de migración necesitan tener un nombre con un formato concreto para su correcto funcionamiento.
6. No hay posibilidad de ejecutar un script de migración en todos los despliegues.
7. Para la ejecución de las migraciones hay que usar el usuario ADM para su ejecución.

## **Pregunta 15** (0,2 puntos)

Selecciona las afirmaciones ciertas sobre cuestiones generales de FWK Spring Boot.

1. El arquetipo web te ofrece un *scaffolding* de un adaptador API REST y un adaptador repositorio SQL.
2. No existe un *scaffolding* con personalizaciones para adaptadores de mensajería.
3. Existe un arquetipo pensado para publicar librerías en Artifactory y poder usarse desde distintos microservicios.

4. El versionado de las librerías gestionadas por *framework* se realiza a través del uso del *parent-seed.*
5. El uso de *contract-first* nos permite generar tanto la parte cliente como la parte de los POJOS e interfaz del controlador REST usando un *plugin* de Maven. En el caso de querer subir el artefacto a Artifactory, debemos usar la *pipeline*.
6. Los archivos `values-*.yaml` nos permiten configurar los perfiles locales de Spring Boot y autogenerar los adaptadores.
7. Los archivos `values-*.yaml` se usan en las *pipelines* de despliegue para generar los descriptores necesarios para Kubernetes.

## **Ejercicio práctico** (hasta 8 puntos)

En Mercadona todos los productos tienen un EAN asociado *(código de barras)*. Estos EAN tienen el formato:

- PPPPPPP+NNNNN+D

Ejemplos: 8437008459059, 8480000160072.

Los dígitos indicados con P hacen referencia al proveedor que fabrica el producto. Los dígitos indicados con N hacen referencia al código del producto dentro de Mercadona y el dígito indicado con D hace referencia a un digito de destino.

Surge la necesidad de consultar los datos de un producto a partir del código EAN.

### **Reglas de negocio**

- Todos los EAN son solo numéricos. No pueden venir letras en el código de barras.
- El último número se utiliza para el destino que lleva el producto en cuestión.
    - Si el número esta entre el 1 y el 5, ese producto va destinado a tiendas Mercadona España.
    - Si el número es 6, ese producto va destinado a tiendas Mercadona Portugal.
    - Si el número es 8, ese producto va destinado a Almacenes.
    - Si el número es 9, ese producto va destinado a Oficinas Mercadona.
    - Si el número es 0, ese producto va destinado a Colmenas.
- Todos los EAN están formados por 13 números.
- El proveedor puede venir el número de Hacendado *(8437008)* o el numero asignado a otro proveedor diferente a marcas de Mercadona.

*Nota: aquellas personas de fuera de Mercadona que opten a este puesto deben hacer igualmente todos los ejercicios, pero sin hacer uso del Framework de desarrollo de Mercadona. En este caso no es necesario usar la guía de estilos de Mercadona, deberá desarrollar algo parecido.*

**Criterios de aceptación**

Hay que montar un microservicio que expone una API REST y desplegarlo en la nube de Google de Mercadona.

Servicio *(obligatorio)*: A partir de un código EAN con el formato explicado anteriormente, debe responder con el detalle del producto: datos básicos del producto, proveedor y del destino *(en formato JSON)*. En caso de que no exista alguno de los datos devolver una excepción.

Servicios *(opcionales)*: Diseña e implementa también los servicios necesarios CRUD para las entidades.

**Valoración**

Se puntuará positivamente:

- Utilización y buenas prácticas de git.
- Utilización de la herramienta Spinnaker.
- Utilización de la herramienta Cloudbees.
- Securización del servicio *(JWT con audiencia de ADFS)*.
- Persistir la información en una BBDD. Por ejemplo, base de datos H2.
- Usar gestor de migraciones.
- Usar caché en el servicio. Por ejemplo, caché in-memory *(Caffeine)*.
- Validaciones de campos.
- Buenas prácticas en el diseño de la API y los formatos de respuesta.
- Modelado de datos.
- Testing y TDD.
- Colección Postman.
- Manejo de excepciones.
- Buenas prácticas de desarrollo *(clean code).*
- Utilización de arquetipo de Fwk Spring Boot *backend*.