

Spring Boot: trucos que te ahorran dolores de cabeza

De principiante a experto, una alpaca a la vez. 🦙



Desliza para subir de nivel 🖱️

 Nivel: Principiante

No empieces desde cero.

Deja que Spring prepare tu estructura base. Selecciona dependencias y genera tu proyecto al instante.



¿Para qué voy a picar 40 archivos si puedo empezar con lo importante?

🌱 Nivel: Principiante

Piensa en Starters, no en librerías sueltas.

Un starter agrupa decenas de dependencias compatibles para evitar conflictos de versiones.

❌ Evita esto

spring-web

tomcat

jackson

tomcat

tomcat

✅ Haz esto

spring-boot-starter-web

No colecciones dependencias. Colecciona soluciones.



Inyección por constructor (DI).

Evita `@Autowired` en propiedades. Usar constructores hace tus dependencias explícitas y facilita los tests.

❌ Evita:

```
@Autowired
private UserService service;
```

✅ Usa (con Lombok):

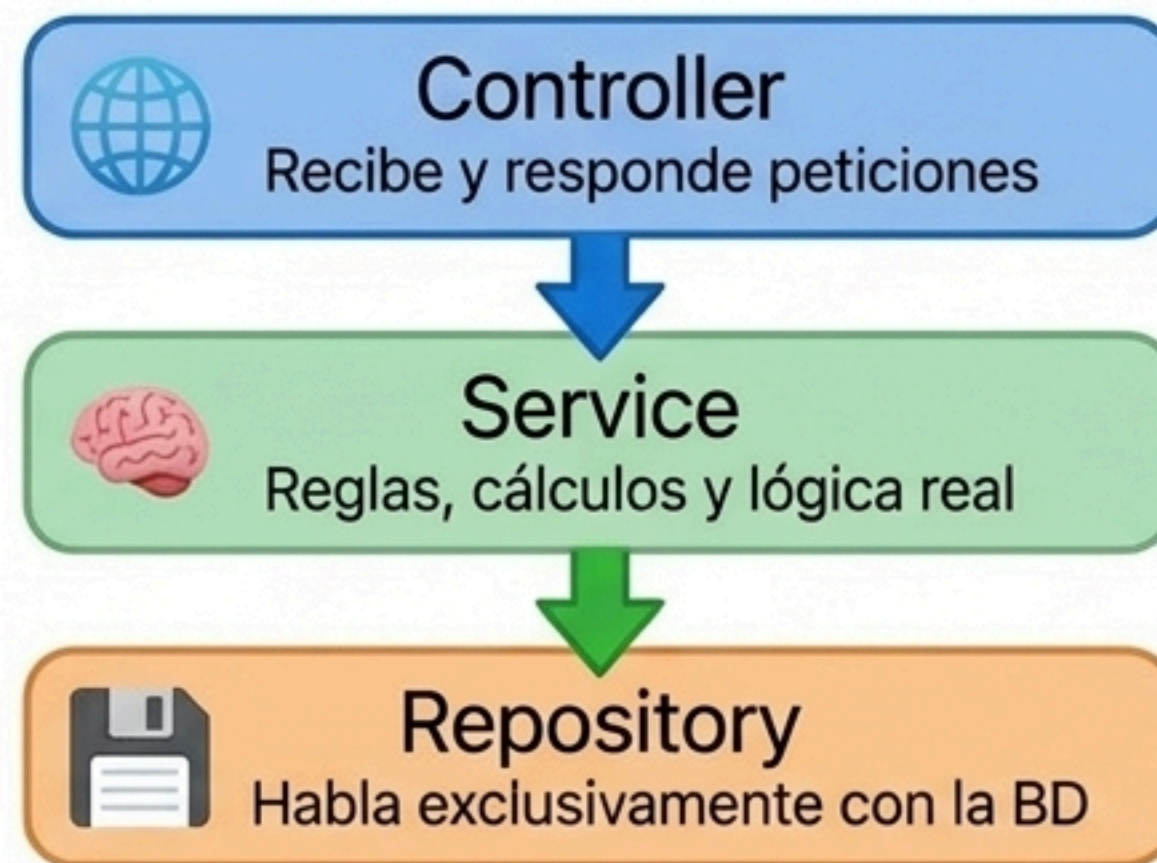
```
@RequiredArgsConstructor
@RestController
public class UserController {
    private final UserService service;
}
```



Si necesito algo, lo declaro. No voy a perseguirlo por toda la aplicación.

Controller $\neq$ Lógica de negocio.

El Controlador solo recibe peticiones HTTP. La inteligencia va en el Service, y la persistencia en el Repository.



Cada alpaca en su corral. ¡No mezcles responsabilidades!



Tu código no es una caja fuerte.

Extrae configuraciones sensibles a variables de entorno y usa perfiles separados (dev, prod).

application.yml

```
spring:
  datasource:
    url: ${DB_URL}
    username: ${DB_USER}
    password: ${DB_PASSWORD}
```

Tu contraseña no es código fuente, criatura.



Seguridad en la entrada y fallos elegantes.

En lugar de hacer múltiples `if (dato == null)`, delega la validación a Spring y centraliza los errores.

```
public ResponseEntity<?>  
    create(@Valid  
          @RequestBody User  
          user)
```



```
@RestControllerAdvice
```

Que falle bien también es parte de un diseño impecable.



@Transactional no es decoración.

Define la unidad real de negocio. Si una de múltiples operaciones de BD falla, todo se revierte (Rollback).

Atómico: Todo o Nada

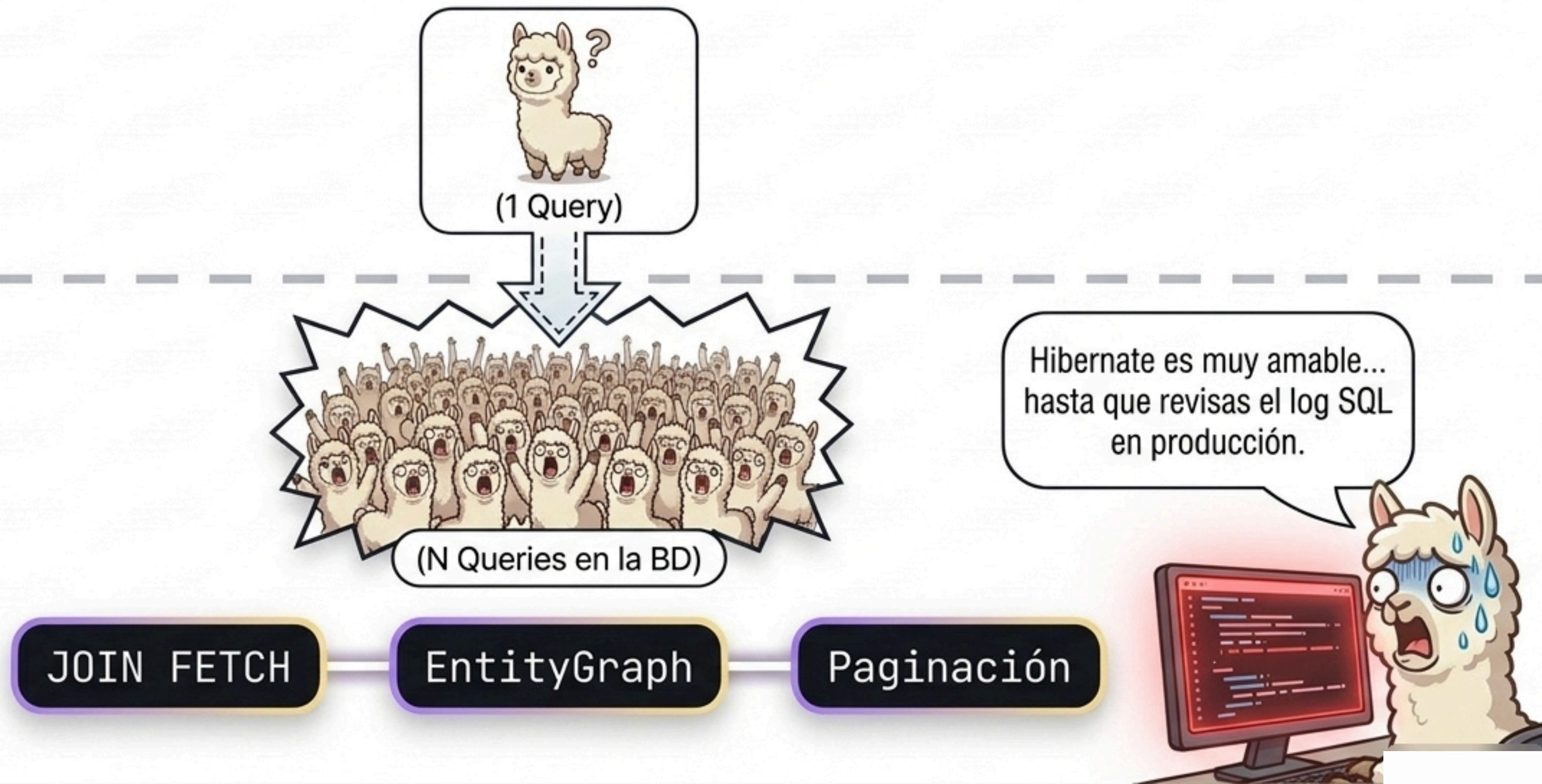
```
@Transactional  
public void transferirDinero() {  
    debitar();  
    acreditar();  
}
```

Varias operaciones. Una sola historia. Todo o nada.



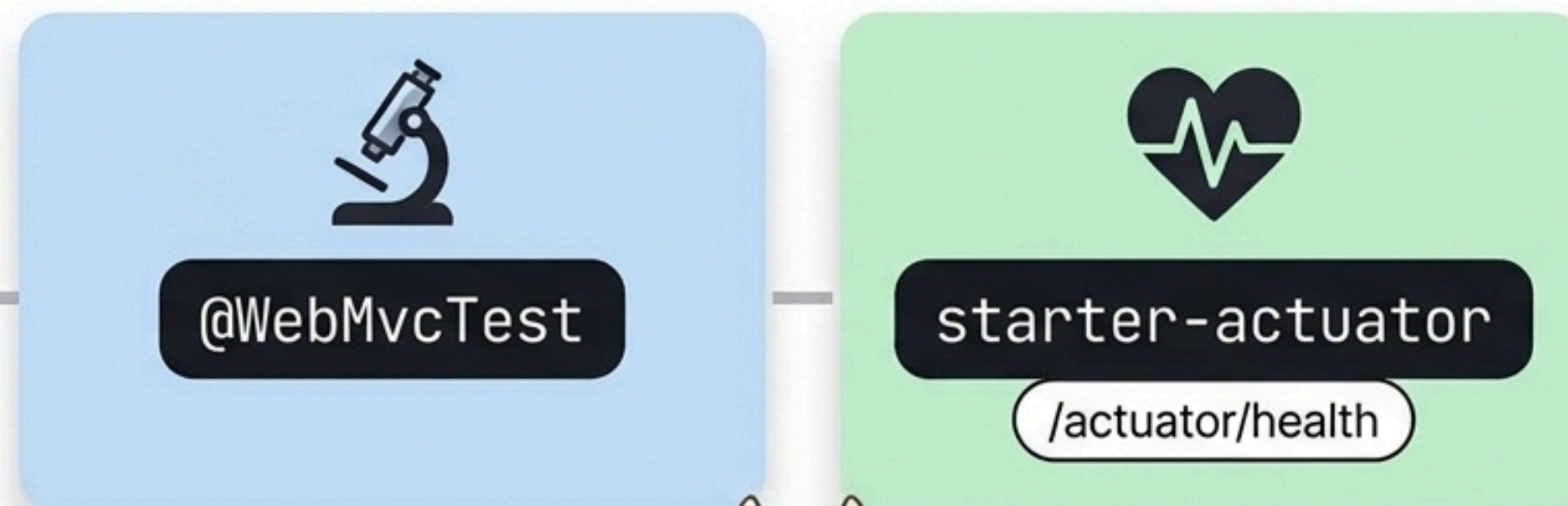
El enemigo silencioso: JPA y el problema N+1.

Recorrer listas asociadas puede disparar cientos de consultas ocultas. Optimiza tus relaciones.



Visibilidad y Testing quirúrgico.

No levantes toda la app para probar un endpoint. En producción, vigila la salud de tu app con Actuator.



Si solo quieres probar una pata,
no despiertes a toda la alpaca.
Con Spring, tú tienes el control.