🎓GUION DIDÁCTICO – ERRORES REALES EN SPRING BOOT + DOCKER + JPA

Este guion está construido **a partir de errores reales**, no de ejemplos ideales. El objetivo es que el alumnado entienda **por qué Spring Boot falla**, cómo diagnosticarlo y cómo razonar soluciones.

✂️Contexto del proyecto

Arquitectura: - Frontend: Vue (MVC en esta fase) - Backend: Spring Boot (Java 17) - Persistencia: MySQL 8 (Docker) - ORM: Spring Data JPA (Hibernate 6)

Objetivo funcional: - Exponer `/api/productos` - Leer datos desde una base de datos inicializada con `init.sql` - Devolver JSON al frontend

---

✒️Error 1 – "Arranca pero no funciona"

Síntoma

- Spring Boot arranca sin errores
- `/` responde correctamente
- `/api/productos` devuelve **500 Internal Server Error**

Diagnóstico didáctico

> En Java, **que la aplicación arranque NO significa que funcione**.

Spring puede: - Crear beans - Inicializar repositorios - Arrancar Tomcat

... y aun así **fallar en el primer acceso real a la base de datos**.

👉Lección clave:

> El error importante ocurre **en tiempo de request**, no en el arranque.

🛌Error 2 – Hibernate y la falsa sensación de seguridad

Mensaje real observado

```
HHH000342: Could not obtain connection to query metadata
```

Qué significa realmente

- Hibernate **no pudo leer la metadata** de la base de datos
- Continuó arrancando igualmente
- El fallo se manifiesta al ejecutar la consulta

👉Lección clave:

> Hibernate **NO siempre falla cuando debería**.

🪂 Error 3 – `ddl-auto=none` es peligroso

Configuración inicial:

```
spring.jpa.hibernate.ddl-auto=none
```

Problema

- Hibernate no valida entidades
- No comprueba si la tabla existe
- No detecta discrepancias de columnas

Solución didáctica

```
spring.jpa.hibernate.ddl-auto=validate
```

👉Lección clave:

> `none` oculta errores `validate` enseña

🌡️ Error 4 – Dialect obsoleto (Hibernate 6)

Mensaje real:

```
MySQL8Dialect has been deprecated
```

Explicación

- Hibernate 6 detecta automáticamente el dialect
- Forzarlo provoca warnings y comportamientos inconsistentes

Solución

Eliminar completamente:

```
spring.jpa.properties.hibernate.dialect=...
```

👉Lección clave:

> En frameworks modernos, **menos configuración suele ser más correcta**.

🎒Error 5 – El schema importa

Problema típico

- Tabla creada correctamente en MySQL
- Hibernate no la encuentra

Causa

- Hibernate puede no usar el schema esperado

Solución explícita

```
@Table(name = "productos", schema = "tienda_forestal")
```

👉Lección clave:

> En producción, la ambigüedad es enemiga de la estabilidad.

🎓Error 6 – Constructor, getters y serialización

Falsas creencias comunes

- "El repository funciona aunque no tenga getters"

Realidad

- JPA puede leer
- Jackson **NO puede serializar** sin getters

👉Lección clave:

> Persistencia ≠ Serialización

🎩Error 7 – Docker no espera a MySQL

Problema

- Spring arranca antes de que MySQL esté listo

Síntoma

- Errores intermitentes
- Conexiones fallidas
- Metadata inconsistente

👉Lección clave:

> Docker Compose **no garantiza orden lógico**, solo orden de arranque.

🎽Aprendizaje transversal

El alumnado aprende que: - Java no perdona suposiciones - Spring no oculta errores, pero hay que saber leerlos - JPA es potente pero estricta - Docker amplifica errores de timing

---

🕐Preparación para el BLOQUE B y C

Este bloque prepara mentalmente para:

💎Java en Docker

- Maven dentro de contenedores
- Multi-stage builds
- Errores reales de dependencias
- Timing de servicios

MVC en Spring

- Controller ≠ lógica
- Service ≠ persistencia
- Repository ≠ base de datos
- Entity ≠ JSON

🔗Conclusión didáctica

> "Cuando todo funciona a la primera, no se aprende nada."

Este proyecto enseña **cómo pensar como desarrollador backend real**, no como ejecutor de tutoriales.