ARQUITECTURA DE SOFTWARE · CLASE 7 · TALLER

Rómpelo a propósito

Nombre: _____ · Dominio elegido: _____

De qué se trata. Una IA les va a construir un sistema en cinco minutos. Eso ya no es mérito de nadie. **Lo que sí es mérito, y es lo que se califica hoy, es decidir qué fronteras tiene ese sistema, escribirlas, y demostrar que se hacen cumplir solas.**

Los cinco pasos

1 10 min

La IA construye

Elijan un dominio **que no sea canchas**: biblioteca, parqueadero, cafetería, gimnasio, veterinaria, lavandería. Péguenle esto a la IA que usen, cambiando solo la primera línea:

```
Actúa como desarrollador. Construye un sistema web mínimo de [MI DOMINIO].

Requisitos estrictos:
1. Exactamente 4 archivos en una carpeta src/, uno por responsabilidad.
2. Módulos ES6 con import/export. SIN frameworks, SIN npm, SIN build.
3. Un index.html en la raíz que importe src/app.js, con HTML y CSS simples.
4. Cada módulo hace UNA cosa y expone funciones claras.
5. IMPORTANTE: por ahora los módulos se llaman entre sí DIRECTAMENTE
   con import. No uses eventos todavía.
6. Datos en memoria, sin base de datos.
7. Que funcione abriendo index.html en el navegador.

Entrégame los archivos completos, uno por uno, listos para copiar.
```

Creen la carpeta `src/` y peguen ahí los cuatro archivos. Abran `index.html` y confirmen que corre.

2 10 min

Ustedes deciden los drivers

Miren el sistema que salió. Escriban **dos drivers**: cosas que el negocio necesita y que la arquitectura tiene que proteger. No son funcionalidades — son **propiedades**.

Ejemplos de driver bien escrito: «si el correo se cae, el préstamo no se puede caer» · «el catálogo solo consulta, nunca modifica» · «un error de búsqueda no puede cobrarle a un socio».
Mal escrito: «que sea rápido» · «que use buenas prácticas» · «que sea escalable». Eso no protege nada.

Driver 1:

Driver 2:

---

3 12 min

Escriban los dos ADR

Uno por driver. Cada uno con estas cinco secciones. **La quinta es la que más pesa.**

```
# ADR-001 · [título corto de la decisión]

## Contexto
[qué situación obliga a decidir esto]

## Driver que manda
[el del paso 2, tal cual]

## Decisión
[qué módulo NO puede importar a cuál, y por qué]

## Alternativa descartada
[qué era lo obvio, y por qué no]

## Qué pagamos
[toda decisión cuesta algo. Si esto está vacío, el ADR no sirve]
```

4 10 min

Conviertan UN ADR en una regla

Descarguen el **kit del taller** y peguen su `src/` adentro. Abran `arquitectura/reglas.json` y llénenlo:

```
[
  {
    "id": "R1",
    "adr": "ADR-001",
    "modulo": "prestamos",
    "no_puede_importar": ["avisos"],
    "porque": "Si el correo se cae, el prestamo NO se puede caer."
  }
]
```

El campo `porque` va en **lenguaje de negocio**, no técnico. Es lo que va a leer quien rompa la regla dentro de seis meses.

5 13 min

Rómpanlo, y demuéstrenlo

Aquí está el punto del taller. Tres commits, en este orden:

#	QUÉ HACEN	QUÉ DEBE PASAR
1	Suben el sistema sano	Pipeline **verde**
2	Agregan a mano el import que su regla prohíbe	Pipeline **ROJO**, con el mensaje de su `porque`
3	Quitan esa línea	Pipeline **verde** otra vez

El commit rojo es la entrega. Sin él no hay prueba de que su regla funciona. Una regla que nunca se violó a propósito no se sabe si sirve.

La conexión con lo que vimos hoy

El prompt les pidió a la IA que los módulos **se llamen directamente**. Eso es lo que vimos en el simulador como **llamada directa**: si una pieza espera a otra y la otra falla, **se cae toda la cadena**.

Su **ADR-002** tiene que ser exactamente esa decisión: **¿qué módulo de su sistema no debería depender de que otro responda?** Casi siempre es el que avisa, el que notifica, el que registra. Ese es el que se separa.

Si les alcanza el tiempo · el punto extra. Cambien esa llamada directa por un anuncio: en vez de `import { avisar }` y llamarlo, que el módulo **publique un evento** y el otro lo escuche. Pídanselo a la misma IA — sabe hacerlo. Y después corran el verificador: **debe seguir en verde**, porque ya no hay import prohibido.

Qué se entrega

QUÉ	DÓNDE
Esta hoja, con los dos drivers escritos	En papel, al final de la clase
El repositorio público con `src/`, los dos ADR y `reglas.json`	GitHub · enlace por el canal
Los tres commits: verde, rojo, verde	Visible en el historial

Antes de entregar · verifiquen

☐ El sistema corre al abrir `index.html`
☐ Los dos drivers están escritos como **propiedades**, no como funcionalidades
☐ Los dos ADR tienen las cinco secciones, incluida **«qué pagamos»**
☐ `reglas.json` tiene el campo `porque` en lenguaje de negocio
☐ En Actions se ven **tres corridas: verde, roja, verde**
☐ El mensaje de cada commit dice qué hicieron, no «cambios»