ARQUITECTURA DE SOFTWARE · ING. DE SISTEMAS · UTADEO · 2026-2

Proyecto Integrador · Canchas Bogotá

Parcial del segundo corte · **40% de la nota** · Trabajo en grupo

Lea esto primero. Este proyecto **no es construir una aplicación grande**. El sistema base ya está hecho y funciona. Lo que usted entrega es **el rastro de sus decisiones de arquitectura**: qué decidió, por qué, qué descartó, qué cuesta — y una regla automática que hace cumplir cada decisión.

1. Qué van a entregar

Un repositorio en GitHub, público, desplegado y con el pipeline en verde. Adentro:

QUÉ	DÓNDE	CUÁNTOS
La aplicación funcionando	URL pública de GitHub Pages	1
ADR — decisiones documentadas	`arquitectura/ADR-00X-*.md`	Los 3 base + **1 por clase**
Reglas verificables	`arquitectura/reglas.json`	Una por cada ADR propio
Historial de commits	El repositorio	Mensajes que citen el ADR

2. La regla de oro

Si cambia la arquitectura, cambia el ADR y cambia la regla.
Un ADR que nadie hace cumplir es un papel. Una regla sin ADR es una imposición sin razón. **Las tres cosas o ninguna:** decisión escrita, regla que la verifica, pipeline que la aplica.

3. Cómo crece, clase por clase

CLASE	TEMA	LO QUE LE AGREGAN AL REPOSITORIO
6	Git, CI/CD y funciones de aptitud	Fork desplegado + ADR-004 con su regla
7	Microservicios	Separan un módulo con contrato explícito. ADR + regla que lo protege.
8	Arquitecturas basadas en eventos	Comunican dos piezas por eventos. ADR + regla.
9	CQRS	Separan el modelo de lectura del de escritura. ADR + regla.
10+	Refinamiento	Ajustes y preparación de la sustentación.

No es trabajo extra. Es el mismo contenido de esas clases, aplicado a su repositorio en vez de quedarse en la diapositiva.

4. Cómo se califica

CRITERIO	PESO	QUÉ SE OBSERVA
El pipeline en verde	**20%**	La URL responde y la verificación de arquitectura pasa. Es objetivo: está o no está.
Los ADR	**30%**	Uno por clase. Cada uno con: el driver que manda, la decisión, **la alternativa descartada** y **qué se paga**.
Las reglas	**25%**	Cada ADR propio tiene su regla en `reglas.json`, con el campo `porque` escrito con una razón real. Y la regla **de verdad detecta** la violación.
Sustentación final	**25%**	El profesor elige **una** de sus decisiones y ustedes la defienden **sin notas**: por qué esa y no la alternativa, y qué están pagando.

Lo que baja la nota de inmediato: un ADR sin el campo «qué pagamos» · una regla sin razón escrita · borrar una regla en vez de resolver el problema · un commit que diga solo «cambios».

5. El sistema base

Canchas Bogotá — reserva de canchas sintéticas. Cuatro canchas, cuatro horarios, reserva y pago con confirmación.

MÓDULO	QUÉ HACE	CARACTERÍSTICA ARQUITECTÓNICA
`nucleo`	Datos compartidos y bus de eventos	Base. No depende de nadie.
`catalogo`	Muestra canchas y horarios	Solo lectura. Tráfico alto.
`reservas`	Bloquea un horario	Recurso único: consistencia fuerte.
`pagos`	Cobra la reserva	No puede caerse por culpa del correo.
`notificaciones`	Avisa al cliente	Escucha eventos. Nadie la llama.

Las tres reglas que ya vienen

REGLA	ADR	QUÉ DICE
R1	ADR-001	`catalogo` no puede importar `reservas`, `pagos` ni `notificaciones`
R2	ADR-002	`pagos` no puede importar `notificaciones` ni `catalogo`
R3	ADR-003	`nucleo` no puede importar a ningún módulo

6. Dónde consultar cuando algo falle

Guía de campo · Git y GitHub — está en los recursos del curso. Es una sola página, con todos los comandos listos para copiar:

Módulo 1 — el ciclo completo en su máquina, con VS Code y Git Bash.
Módulo 2 — comandos de Git: ramas, conflictos y las cinco formas de volver atrás.
Módulo 3 — cómo usar **su** cuenta en un computador compartido de la universidad.
Módulo 4 — el paso a paso desde GitHub, sin instalar nada.

Ábrala antes de preguntar: tiene una sección de **qué hacer cuando el pipeline falla** con los dos errores más comunes.

7. Montaje · paso a paso

Todo es **gratis**, sin tarjeta y **sin instalar nada**. Solo hace falta una cuenta de GitHub y un navegador.

1. **Fork.** Entren al repositorio del curso y presionen **Fork**. Queda una copia en su cuenta.
2. **Activen Pages.** `Settings → Pages → Source` y elijan **GitHub Actions**.
3. **Activen los flujos.** Pestaña `Actions` → botón para habilitarlos.
4. **Esperen el verde.** Cuando el pipeline termine, su URL aparece en `Settings → Pages`.
5. **Anoten la URL.** Es parte de la entrega.

Verificar la arquitectura desde su máquina (opcional)

```
node tools/verificar-arquitectura.js
```

Sale **0** si todo está bien y **1** si hay una violación. Es exactamente lo que corre el pipeline.

8. Plantilla de ADR

```
# ADR-00X · [titulo corto de la decision]
Fecha: [fecha] · Estado: aceptada · Grupo: [numero y nombres]

## Contexto
[que esta pasando que obliga a decidir]

## Drivers que mandan
- [driver 1, como escenario medible]
- [driver 2]

## Decision
[que se decidio, en una o dos frases]

## Alternativas consideradas
[que mas se pudo hacer y por que se descarto]

## Consecuencias
Ganamos: [...]
Pagamos: [...]        <-- si esto esta vacio, el ADR esta incompleto

## Como se verifica
Regla [RX] en arquitectura/reglas.json
```

9. Plantilla de regla

```
{
  "id": "R4",
  "adr": "ADR-004",
  "modulo": "reservas",
  "no_puede_importar": ["notificaciones"],
  "porque": "[la razon real, no 'porque si'. Se muestra cuando falla.]"
}
```

Prueben que su regla sirve: viólenla a propósito, confirmen que el pipeline se pone rojo, y déjenla en verde. Una regla que nunca se probó no se sabe si funciona.