

Arquitectura de Software

Repaso

De los requisitos a una arquitectura decidida, documentada y evaluada

Gerson Garrido

Semana 8

UNIVERSIDAD NACIONAL DE INGENIERÍA

Agenda

Sesión A: repaso

1. El hilo de las siete semanas
2. Lo que hay que tener preciso, semana por semana
3. Un caso nuevo: Apachi
4. Diez ejercicios resueltos
5. Su P2, con estos criterios

Sesión B: examen parcial

- Partes 1 y 2 del curso, semanas 1 a 7.

Laboratorio

- No hay: el tiempo es para cerrar P2.

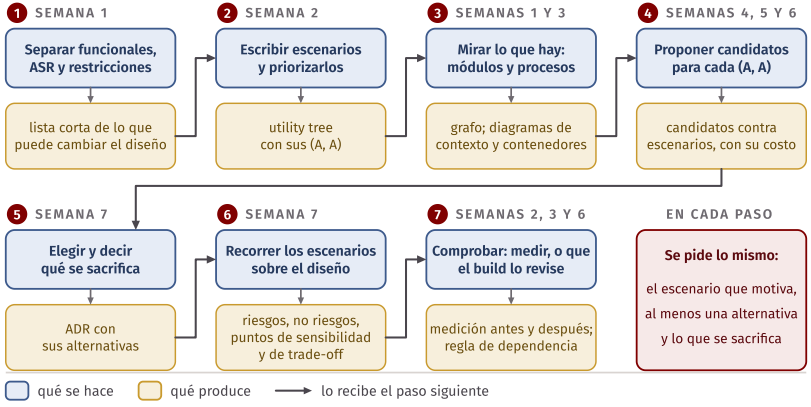
> **Lectura**
>
> Las *Preguntas de repaso* de las guías de las semanas 1 a 7 y `exams/README.md`.
> Pesan más: *Software Architecture in Practice*, cap. 1, 3, 8, 9 y 4; el artículo de Parnas (1972); Nygard, *Documenting Architecture Decisions* (2011).

SECCIÓN 01

El hilo de las siete semanas

01

Siete semanas, un procedimiento



Cada paso recibe lo que produjo el anterior. Una pregunta toma **un** paso y da lo necesario para hacerlo: conviene saber en cuál están.

Lo que lleva toda respuesta

1. El **escenario** que motiva la decisión, con su medida.
2. Al menos una **alternativa**. Casi siempre, una es no cambiar nada.
3. Lo que se **sacrifica**: qué atributo paga.

Es la regla de trabajo de la semana 1. Sirve para revisar cualquier respuesta antes de entregarla.

Error común

Estudiar memorizando listas de tácticas o de estilos. El caso es nuevo y pide decidir y justificar. Una respuesta que no dice qué se sacrifica está incompleta.

SECCIÓN 02

Semana por semana

02

Semana 1: qué es la arquitectura

Dos definiciones

Las **estructuras** necesarias para **razonar** sobre el sistema.

Las decisiones que **cuesta cambiar**: ¿cuántos archivos, equipos y despliegues toca deshacerla?

Estructura	Responde
Módulos	¿Qué toca un cambio?
Componentes y conectores	¿Qué pasa con la latencia y con las fallas?
Asignación	¿Quién lo construye y dónde corre?

Semana 1: qué la determina

- **ASR**: un requisito que, si cambiara, cambiaría la arquitectura. Dice cuánto, qué tan rápido, con qué frecuencia cambia algo o qué pasa si falla.
- **Restricción**: una decisión que ya viene tomada. Un requisito se negocia; una restricción, no.
- **C4**: contexto (el sistema como una caja) y contenedores (lo que corre por separado). Toda caja dice qué es y con qué tecnología; toda flecha, qué viaja y por qué protocolo.

Error común

Clasificar por el tema y no por la prueba: cámbienlo y miren si el diseño tendría que ser otro.

Semana 2: el escenario y sus medidas

Seis partes

1. Fuente del estímulo
2. Estímulo
3. Artefacto
4. Entorno
5. Respuesta
6. Medida de la respuesta

Sin medida no es un escenario.

Percentil por rango más cercano

Con n mediciones ordenadas, el percentil p es la de la posición $\lceil p/100 \cdot n \rceil$, contando desde 1.

Recordar

El promedio mezcla poblaciones. Los percentiles no se promedian.

Semana 2: disponibilidad y prioridad

- **En serie** (la operación necesita a todas):

$$A = A_1 \cdot A_2 \cdots A_k$$

- **En paralelo** (basta una copia):

$$1 - (1 - A_1)(1 - A_2)$$

- Las dos suponen fallas independientes.

Utility tree

Atributos, refinamientos y escenarios en las hojas, con dos notas: valor para el negocio y dificultad técnica. Los **(A, A)** definen la arquitectura.

Trade-off

Una decisión que mejora un atributo y cuesta otro.

Semana 3: límites y acoplamiento

- **Parnas**: dividir por lo que puede cambiar, no por el orden en que se ejecuta.
- **Ocultamiento de información**: la interfaz es todo lo que los demás pueden suponer del módulo.
- **Connascence**: de nombre, de tipo, de significado, de posición, de algoritmo; y las dinámicas. Cuanto más lejos están dos elementos, más débil debe ser lo que los ata.
- La medida que sirve para diseñar: **cuántos módulos toca un cambio probable**.

Semana 3: el grafo de dependencias

- Una flecha de A a B si A importa algo de B.
- Un **ciclo** hace de sus módulos uno solo con varios nombres.
- Lo que cambia mucho depende de lo que cambia poco: $I = C_e/(C_a + C_e)$.
- Romper un ciclo: fusionar, subir la función, **invertir** una dependencia. Un módulo `shared` casi siempre lo esconde.
- Las reglas se comprueban con `depcruise`, y se las ve fallar.

Inversión

Quien necesita algo declara la interfaz; el otro la implementa; un tercero los conecta. El `import` cambia de dirección; la llamada, no.

Semana 4: un estilo es lo que prohíbe

Estilo de arquitectura

Un vocabulario de elementos y conectores más un conjunto de **restricciones**. Sus propiedades salen de las restricciones.

Monolítico o distribuido es una pregunta sobre el **despliegue**. Monolito no es big ball of mud.

Estilo	Abarata
Capas	Cambiar una tecnología
Modular monolith	Un cambio de negocio
Pipeline	Reemplazar un paso
Microkernel	Agregar una variante

Semana 4: modular monolith

Un solo despliegue, con módulos por dominio que cumplen tres condiciones **que se comprueban**:

1. Cada módulo tiene una interfaz pública.
2. Los datos de cada módulo son suyos.
3. Las dependencias están declaradas y no tienen ciclos.

Lo que da

Buena parte de la modificabilidad de los microservicios, sin pagar la red.

Lo que no da

Lo que depende del despliegue: desplegar y escalar una parte sola.

Semana 5: cruzar la red

- Se distribuye para cumplir un **escenario** que el monolito no cumple.
- La llamada tarda más, su tiempo es una distribución, y falla de maneras que quien llama no distingue. Un timeout no cancela nada.
- **Service-based**: pocos servicios grandes, base de datos compartida. **Microservicios**: cada servicio es owner de sus datos.
- Una cadena síncrona es una serie: cuatro componentes de 99,9 % dan 99,6 %.

Semana 5: eventos, y cómo se elige

Comando: pide que algo pase, a un destinatario.

Evento: dice que algo ya pasó, a nadie en particular.

Event-driven saca trabajo del camino de la respuesta. Cobra consistencia eventual y tres preguntas: falla a la mitad, duplicados, orden.

Elegir un estilo

1. Tomar los escenarios (A, A).
2. Cómo cumple cada candidato cada uno, con una frase concreta.
3. Sumar el costo de operarlo.
4. Elegir y escribir qué se sacrifica.

Semana 6: tácticas

Táctica

Una decisión de diseño que influye sobre la respuesta a un estímulo de **un** atributo. Lo que les hace a los demás es el trade-off.

Atributo	Grupos de tácticas
Modificabilidad	Aumentar la cohesión, reducir el acoplamiento, diferir el enlace
Rendimiento	Controlar la demanda, administrar los recursos
Disponibilidad	Detectar, recuperarse, prevenir
Seguridad	Detectar, resistir, reaccionar, recuperarse

Semana 6: primero se mide

- El tiempo de una petición es tiempo de **procesamiento** más tiempo **bloqueado**.
- **N+1**: reducir el costo de cada petición.
- **Pool de conexiones**: un tope a la concurrencia; más conexiones no es más rápido.
- **Cache**: se gana latencia; se abren los datos viejos y la invalidación.
- **Retry**: recupera de una falta transitoria, y puede duplicar una operación.

Error común

Optimizar sin medir: la táctica correcta para un cuello de botella que no es el que limita no cambia nada.

Semana 7: el ADR

Secciones

1. Contexto, con el escenario
2. Alternativas consideradas
3. Decisión
4. Estado
5. Consecuencias, también las malas

No se reescribe

Si la decisión cambia, se escribe otro que lo reemplaza. En el anterior solo cambia el estado.

Error común

Un ADR sin alternativas: nadie puede saber si sigue vigente cuando cambie el contexto.

Semana 7: evaluar con escenarios

ATAM recorre los escenarios del utility tree sobre los enfoques de arquitectura.

Hallazgo	Qué es
Riesgo	Una decisión, o una que falta, que puede impedir un escenario
No riesgo	Una decisión que se analizó y está bien, bajo un supuesto
Punto de sensibilidad	Una propiedad de la que depende mucho un atributo
Punto de trade-off	Una propiedad que mueve dos atributos en sentidos opuestos

Lo que no hace

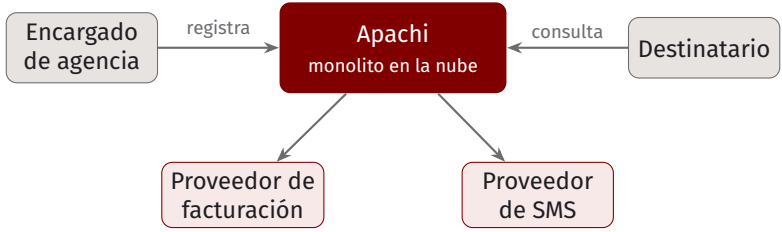
No dice si la arquitectura es buena, no da números y no reemplaza a medir. Dice dónde mirar.

SECCIÓN 03

Un caso nuevo: Apachi

03

Apachi: encomiendas entre 38 agencias



- El encargado registra una encomienda y entrega una guía con un código de seguimiento.
- El destinatario consulta con el código, desde su celular.
- Cada encomienda lleva un comprobante electrónico y un SMS.

Lo que dijo la empresa

1. Registrar una encomienda y entregar la guía.
2. Consultar el estado con el código.
3. Un SMS al registrar y otro al llegar.
4. Para recoger, el DNI; se guarda quién entregó.
5. «Robusto y fácil de usar.»
6. Campaña: 9 000 encomiendas al día y 40 consultas por segundo.
7. Doce agencias pierden el enlace por horas; el mostrador no puede parar.
8. Las tarifas cambian seis veces al año; cada cambio demora dos semanas.
9. Comprobante electrónico, con un proveedor ya contratado.
10. Nube ya contratada; cinco personas; TypeScript y PostgreSQL.
11. Una encomienda no se pierde ni se entrega a otro.

SECCIÓN 04

Ejercicios resueltos

04

1. ¿Requisito funcional, ASR o restricción?

#	Clasificación	Por qué
1, 2, 4	Funcional	Un formulario, una consulta, un dato más
3	Funcional, y esconde un ASR	Si el SMS no responde, ¿se deja de registrar?
5	Todavía no es un requisito	No se puede decir cuándo se incumple
6, 7, 8	ASR	Cuánto; qué pasa si falla; con qué frecuencia cambia
9	Restricción, y esconde un ASR	El proveedor ya está contratado
10	Restricciones	La nube, el equipo y lo que conoce
11	Dos ASR sin medida	Falta el número

La pregunta que destapa

«¿Y si falla?» La empresa respondió: el mostrador sigue; el SMS y el comprobante pueden llegar después.

2. El requisito 7, como escenario

- **Fuente:** el enlace a internet de una agencia de la sierra.
- **Estímulo:** deja de funcionar durante 2 horas.
- **Artefacto:** el registro de encomiendas en el mostrador.
- **Entorno:** campaña de diciembre, 40 encomiendas por hora.
- **Respuesta:** el mostrador sigue registrando y entrega la guía con su código.
- **Medida:** 0 encomiendas perdidas, y el 100 % se puede consultar en 10 minutos o menos desde que el enlace vuelve.

E1

Disponibilidad.
Es el único **(A, A)**: vale mucho y, con un mostrador servido desde la nube, es difícil.

2. Corregir un escenario

«Fuente: los usuarios. Estímulo: consultan sus encomiendas. Artefacto: el sistema. Respuesta: el sistema responde. Medida: siempre en un tiempo razonable.»

```
make errors

errors/vague-scenarios.md
  ✗ El seguimiento debe ser rápido (línea 7)
      Entorno: falta
      Medida de la respuesta: no tiene ningún número: ¿cómo se comprueba?
      Medida de la respuesta: "siempre" no se puede comprobar: escriban el número
```

- El revisor ve la forma. Una persona ve el resto: ¿cuántos usuarios?, ¿con qué frecuencia?, ¿qué parte del sistema?
- Corregido (E2): 40 consultas por segundo durante 2 horas, en campaña; **p95 menor que 500 ms** y menos de 0,5 % de errores.

2. El utility tree

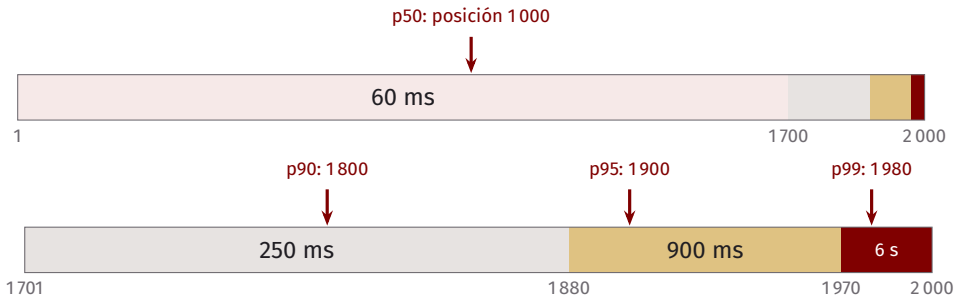
Atributo	Escenario	Negocio	Técnica
Disponibilidad	E1. Sin enlace 2 h: 0 perdidas, consultable en 10 min	A	A
Rendimiento	E2. 40 consultas por segundo: p95 $<$ 500 ms	A	M
Modificabilidad	E3. Tarifa nueva: 1 día-persona, 1 módulo	M	B
Seguridad	E4. 10 000 códigos probados: 0 datos personales devueltos	A	M

Cómo se lee

El (A, A) define la arquitectura. La dificultad depende del diseño que se tiene.

3. Una tabla de latencias

2 000 consultas: 1 700 de 60 ms, 180 de 250 ms, 90 de 900 ms y 30 de 6 s.



Con una tabla de frecuencias basta saber qué posiciones ocupa cada fila.

3. Lo que calcula el revisor

```
make percentiles

n = 2000
posiciones      valor
1 a 1700        60 ms
1701 a 1880     250 ms
1881 a 1970     900 ms
1971 a 2000     6000 ms
promedio = 204.0 ms
p50  posición 1000  60 ms
p90  posición 1800  250 ms
p95  posición 1900  900 ms
p99  posición 1980  6000 ms
p100 posición 2000  6000 ms
```

$$\frac{1700 \cdot 60 + 180 \cdot 250 + 90 \cdot 900 + 30 \cdot 6000}{2000} = 204$$

$$\lceil 0{,}95 \cdot 2000 \rceil = 1900$$

El promedio es 204 ms; el p95 está en la posición 1 900.

E2 no se cumple

Pide p95 menor que 500 ms, y el p95 es 900 ms.

3. Lo que esconden los números

- El promedio (204 ms) queda por debajo de 500 y **no describe a ningún pedido**: el 85 % tardó 60 ms.
- El p90 (250 ms) también queda por debajo. Quien mire solo eso dirá que está bien.
- Cuatro valores tan separados son **poblaciones** distintas: pedidos que hacen trabajos distintos.
- A quien decide se le muestra la medida del escenario: «p95 de 900 ms contra un objetivo de 500; el 1,5 % tarda 6 s».

Una variante

`api-1`: 1 900 pedidos de 60 ms. `api-2`: 100 pedidos de 900 ms. El promedio de sus p95 es 480 ms; el p95 de los 2 000 juntos es **60 ms** (posición 1 900). Y el p96 ya es 900 ms.

4. ¿Qué necesita registrar una encomienda?



$$A = 0{,}992 \cdot 0{,}9995 \cdot 0{,}9995 \cdot 0{,}995 \cdot 0{,}999 \approx 0{,}98507$$

$$(1 - 0{,}98507) \cdot 8760\ \text{h} \approx 130{,}8\ \text{h al año}$$

En serie

El conjunto siempre es peor que su peor parte, y cada dependencia en el camino resta.

4. Dos alternativas, con su cuenta

Diseño	Registrar	Al año
Actual: todo dentro de la petición	98,5067 %	130,8 h
(a) El comprobante y el SMS salen después	99,1008 %	78,8 h
(b) Lo mismo, más un segundo enlace 4G de 97 %	99,8760 %	10,9 h

Los dos enlaces, en paralelo: $1 - (1 - 0{,}992)(1 - 0{,}97) = 1 - 0{,}008 \cdot 0{,}03 = 0{,}99976$

Independencia

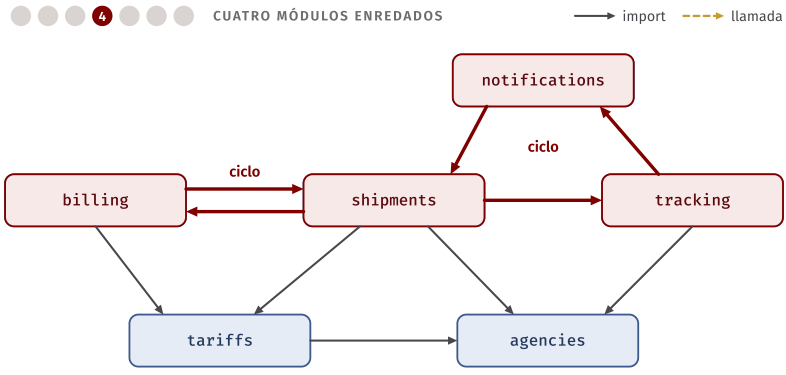
Si los dos enlaces dependen de la misma energía o de la misma torre, se caen juntos.

El número no responde E1

E1 describe 2 horas sin enlace: un 99,88 % anual puede no cumplirlo nunca.

5. Un grafo de dependencias

4 CUATRO MÓDULOS ENREDADOS



Los dos ciclos pasan por shipments: cuatro de los seis módulos son, en la práctica, uno solo.

No comparten ninguna flecha: hay que quitar al menos 2, una de cada ciclo.

5. Lo que dice el revisor, y lo que no

```
make cycles

6 módulos, 10 dependencias
Ciclos: 2
  billing -> shipments -> billing
  notifications -> shipments -> tracking -> notifications
Módulos enredados (de cada uno se llega a todos los demás del grupo):
  billing, notifications, shipments, tracking
Hay que quitar al menos 2 flechas; formas de hacerlo: 6
```

- El programa dice **cuántas** flechas sobran. **Cuáles**, lo decide la regla de estabilidad: lo que cambia mucho depende de lo que cambia poco.
- La facturación cambia con cada norma y cada proveedor; el registro es lo más básico. Sobra `shipments -> billing`.

5. Invertir la dependencia

shipments/ports.ts

```ts
// La interfaz es de shipments, que es quien la usa.
export interface ReceiptIssuer {
  issue(shipment: Shipment): Promise<string>;
}
```

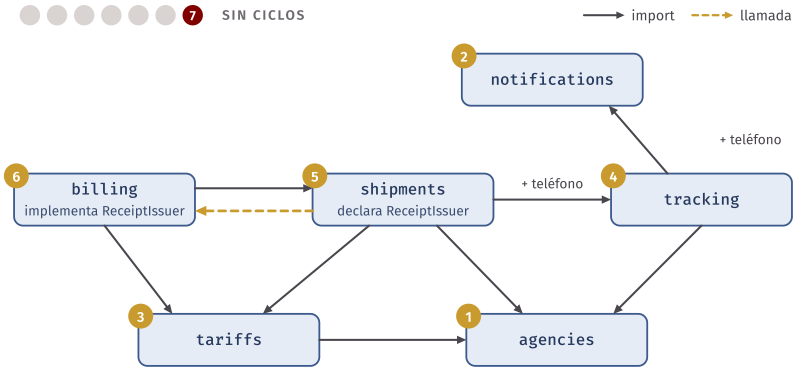
billing/receipts.ts

```ts
import type { ReceiptIssuer, Shipment } from '../shipments/index.ts';

export type Billing = ReceiptIssuer & { issued(): readonly Receipt[] };
```

`main.ts` los conecta. La llamada no cambió de dirección; el `import`, sí.

5. Una flecha menos en cada ciclo



Seis módulos, ocho dependencias, sin ciclos: ya hay un orden, primero lo que no depende de nadie.

billing quedó al final: depende de otros y nadie depende de él.

5. La decisión, como regla que se ve fallar

.dependency-cruiser.cjs

```
        name: 'shipments-does-not-import-billing',
        from: { path: '^src/shipments/' },
        to: { path: '^src/billing/' },
```

make errors

```
  error shipments-does-not-import-billing: src/shipments/counter.ts → src/billing/index.ts
  error no-circular: src/billing/index.ts →
      src/billing/receipts.ts →
      src/shipments/index.ts →
      src/shipments/counter.ts →
      src/billing/index.ts

x 2 dependency violations (2 errors, 0 warnings). 7 modules, 12 dependencies cruised.
```

6. Un fragmento con acoplamiento

coupling/quote.ts

```ts
export function quoteByPosition(weightKg: number, declaredValue: number,
  service: number): number
```

De posición: el orden de dos números; al revés, compila igual. **De significado**: ¿soles o céntimos?, ¿qué es `1`?

coupling/quote.ts

```ts
export type Service = 'standard' | 'express';
export type Parcel = {
  weightKg: number; declaredValueCents: number; service: Service;
};
export function quoteCents(parcel: Parcel): number
```

7. Tres candidatos contra tres escenarios

	A. Modular monolith	**B. Microservicios**	**C. Monolith y nodos de agencia**
E1	No: sin enlace no hay mostrador	No: los servicios están igual de lejos	Sí: registrar necesita solo la computadora de la agencia
E2	Depende de la consulta, no del estilo	Podría escalar solo el seguimiento	Igual que A
E3	Sí: un módulo, un despliegue	Sí: un servicio	Sí; la tarifa tiene que llegar a 38 nodos
Costo	El más bajo	Seis servicios para cinco personas	38 instalaciones; consistencia eventual; envíos repetidos

7. Elegir, y decir qué se sacrifica

- B se descarta primero, y no por caro: **no responde el escenario**. El problema no es cuántas piezas hay en la nube, sino que el mostrador depende del enlace.
- Entre A y C decide E1, el (A, A). **Elegimos C**.
- Se sacrifica: el seguimiento deja de ser inmediato; hay 38 instalaciones que operar; aparecen las tres preguntas de los mensajes (falla a la mitad, duplicados, orden).
- El nodo le cuenta a la central un hecho que ya ocurrió: es un **evento**.

No cambiar nada también es un candidato

Seguir con A y llenar la guía en papel cuesta cero, y es lo correcto si E1 deja de ser (A, A).

8. Un escenario y tres tácticas

Medición por partes de E2: una consulta por cada punto del recorrido; una conexión nueva por pedido; los pedidos de 6 s esperaron una conexión.

Táctica	Grupo	Qué cuesta
Una sola consulta con el recorrido	Controlar la demanda	Casi nada: por eso va primero
Un pool de conexiones	Administrar los recursos	Los pedidos esperan en una cola: hay que elegir el tamaño
Cache con TTL de 30 s	Administrar los recursos	Actualidad (30 s de dato viejo) y una pieza más que puede caerse

El orden importa

Se aplica una, se mide con la misma carga y solo entonces se decide la siguiente. Empezar por la cache esconde la consulta mal escrita.

9. La decisión, como ADR

ADR 0001: Las agencias registran encomiendas en un nodo local

- **Contexto:** doce agencias pierden el enlace; registrar tiene 98,5 %. **Escenario:** E1.
- **Alternativas:** A, no cambiar el diseño; B, efectos diferidos y un segundo enlace (99,88 % si son independientes); C, un nodo local.
- **Decisión:** C. **Estado:** Aceptado.
- **Consecuencias:** una buena (E1 se puede cumplir) y varias malas.

Las malas

Seguimiento eventualmente consistente.
Tarifa vieja en un nodo sin enlace.
El mismo envío, dos veces.
Una encomienda en un solo disco.
Datos personales en 38 computadoras.

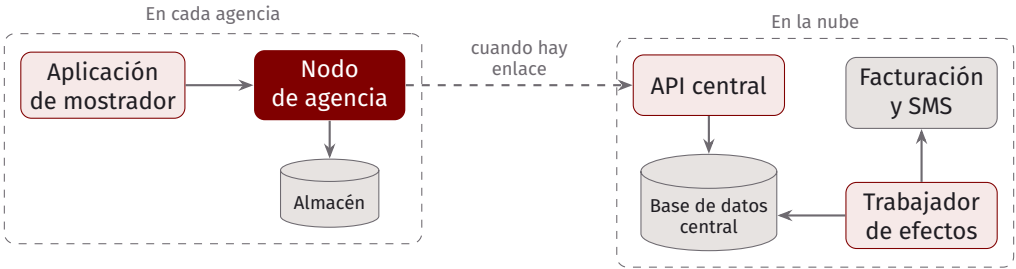
9. Lo que hace que sirva dentro de un año

- El contexto trae el **escenario** y las **cuentas**, no opiniones.
- Hay tres alternativas, y una es no hacer nada.
- B no se descarta por mala: su número depende de un supuesto que aquí no se cumple. Si un día se cumple, quien lea sabrá que la decisión merece revisarse.
- Las consecuencias malas son el trabajo que viene: el envío repetido es la idempotencia *(semana 12)*.

Error común

«Contexto: las agencias pierden el enlace. Decisión: usamos un nodo local. Consecuencias: funciona sin enlace.»

10. El diseño elegido



- Registrar no cruza el enlace: el nodo guarda en su almacén y responde.
- Al volver el enlace, el nodo envía lo pendiente: unas 80 encomiendas después de 2 horas.

10. Recorrer E1 y clasificar

Tipo	Decisión	Razonamiento
No riesgo	El nodo guarda antes de responder	Registrar no usa el enlace ni a los proveedores
Riesgo	Hasta que se envía, la encomienda está en un solo disco	Si se daña, se pierden unas 80; la medida pide 0
Sensibilidad	Cada cuánto envía el nodo lo pendiente	Hay 10 min: con un reintento cada 15, no se cumple
Riesgo	Nadie decidió cómo reconoce la central un envío repetido	La misma encomienda, dos veces, con dos comprobantes
Trade-off	Qué datos se guardan en la agencia	Más disponibilidad sin enlace; más datos personales en 38 computadoras

Tema de riesgo

El contrato entre el nodo y la central está sin decidir. Se resuelve antes de instalar el primer nodo.

SECCIÓN 05

Su P2 y lo que sigue

05

Revisar su P2 con estos criterios

1. Cada decisión nombra su **escenario**, con medida.
2. Cada decisión tiene al menos una **alternativa**, y una es no cambiar nada.
3. Cada decisión dice qué **cuesta**: si todas las consecuencias son buenas, falta analizar.
4. Cada número dice cómo se obtuvo: carga, corridas, máquina y método del percentil.
5. Cada táctica fue a la causa que midieron.
6. Los diagramas coinciden con lo que corre.
7. Pueden defender cada decisión en tres frases: escenario, alternativa y costo.

El código de la semana

```bash
$ node check.ts percentiles data/tracking-latencies.txt
$ node check.ts availability data/register-central.txt
$ node check.ts cycles data/modules.txt --without "shipments -> billing"
$ make test
```

- `code/week-08/`: los revisores de las cuentas y los documentos de Apachi (*`case/`*).
- Para practicar: cambien una tabla de `data/`, hagan la cuenta en papel y comparen.
- Las pruebas fijan cada número de hoy, y los comparan con las funciones de la semana 2.

Para la próxima clase

- Esta semana se entrega **P2 · La apertura de ventas** y se publica **P3 · Lecturas que escalan**.
- Los ejercicios del capítulo, en papel.
- Semana 9, los datos como decisión de arquitectura. Leer de Kleppmann y Riccomini, *Designing Data-Intensive Applications* (2.ª ed.), cap. 1 *Trade-Offs in Data Systems Architecture*.

Pregunta

En Apachi, la encomienda vive primero en el disco de la agencia y después en la base de datos central. ¿Cuál de las dos copias manda cuando no coinciden?

Lecturas de hoy: las *Preguntas de repaso* de las guías 1 a 7; `exams/README.md`; *Software Architecture in Practice*, cap. 1, 3, 4, 8 y 9; Parnas (1972); Nygard (2011).