Arquitectura de Software

Definiciones, Modelos/Vistas y Arquitecturas comunes

AYD - 1

October 23, 2025

Índice

¿Qué es la arquitectura de software?

- Decisiones estructurales de alto nivel.
- Estructura estática (Componentes) vs. Comportamiento dinámico (Interacción).
- Impacto directo en Atributos de Calidad (ADCs).

Atributos de Calidad y Objetivos de Negocio

- Organización modular $\rightarrow$ Desarrollo paralelo.
- Impacto:
  - Desempeño (Tiempo de respuesta)
  - Usabilidad (Facilidad de interacción)
  - Modificabilidad (Facilidad de cambio)
- Alineación con Objetivos de Negocio.

Beneficios: Reducir costos de desarrollo

- Reduce costos a largo plazo.
- Clave: **Reutilización de componentes**.
- Menos "re-trabajo" (rework).
- Facilita mantenimiento e implementación.

Reutilización: Un factor clave

Reutilización = Código + Componentes + Patrones de diseño.

Responsables y decisiones clave

- **Arquitecto de software:** Patrones, límites, no-funcionales.
- **Equipo de desarrollo:** Implementación, feedback de viabilidad.
- **Operaciones/DevOps:** Despliegue, CI/CD, infraestructura.
- **Stakeholders del negocio:** Requisitos, restricciones, prioridades.

Vistas en la Documentación de Arquitectura

- Documentación = ¿Cómo está estructurado el sistema?
- Sistemas complejos $\rightarrow$ Múltiples vistas.
- Vista = "Fotografía" de una parte (ejecución, física, etc.).
- Compuesta por: Diagrama + Texto explicativo.
- Ayuda a diferentes interesados.

Tipos de vistas más comunes

Recomendación

Documentar al menos tres clases de vistas:

- **Vista Lógica**
- **Vista de Comportamiento**
- **Vista Física**

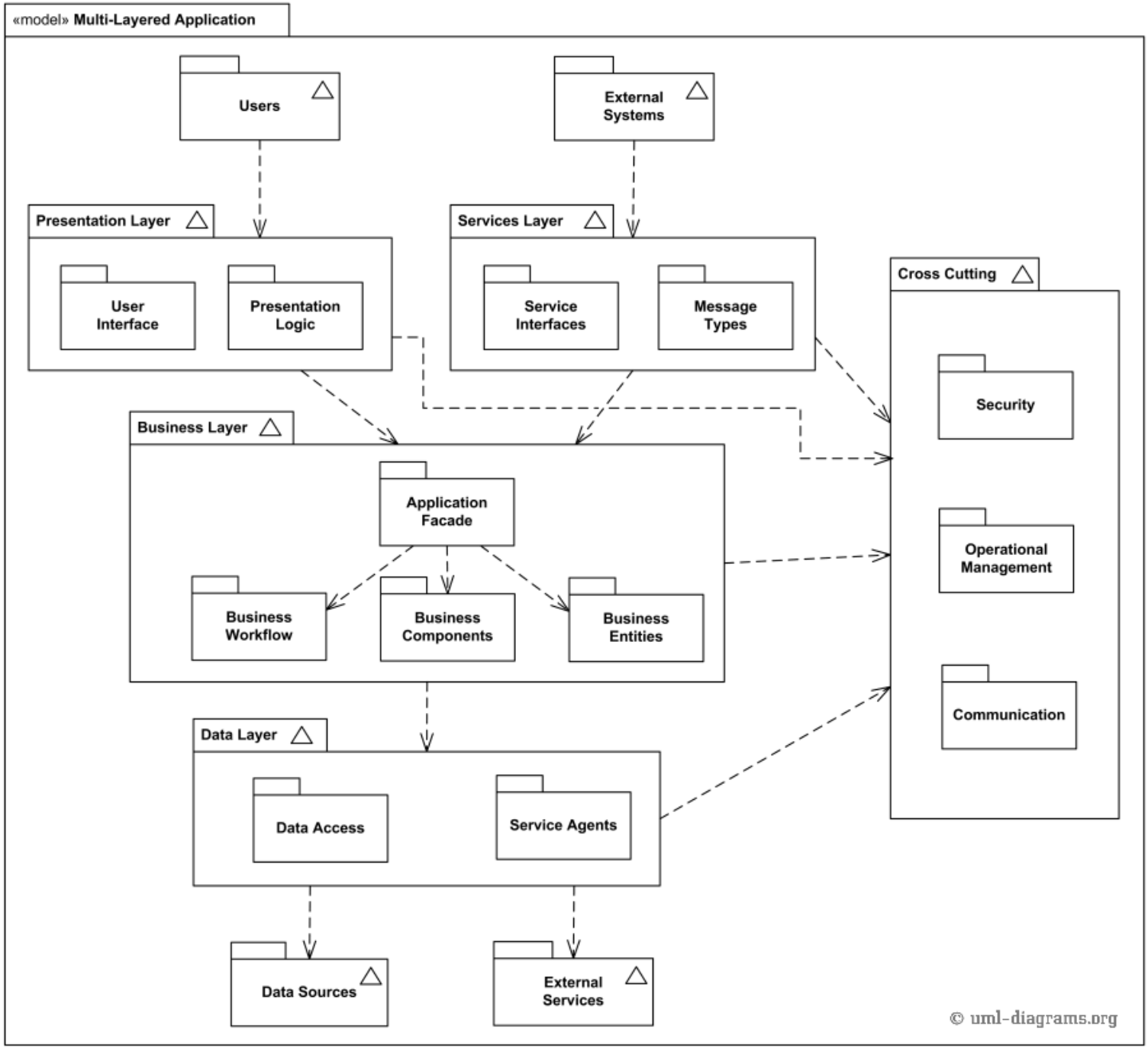
Por qué usar múltiples vistas (Refuerzo)

- Un sistema complejo no se entiende con un solo diagrama.
- Diferentes interesados (Desarrolladores, Operaciones, Negocio).
- Estándar: Modelo **4+1** de Philippe Kruchten.

Vista Lógica

- **Pregunta:** ¿Qué hace el sistema? (Funcionalidad).
- Unidades de implementación (Clases, Paquetes, Módulos).
- Propiedades: Responsabilidades, Interfaces.
- Relaciones: Dependencias, Jerarquías (*depende-de*).

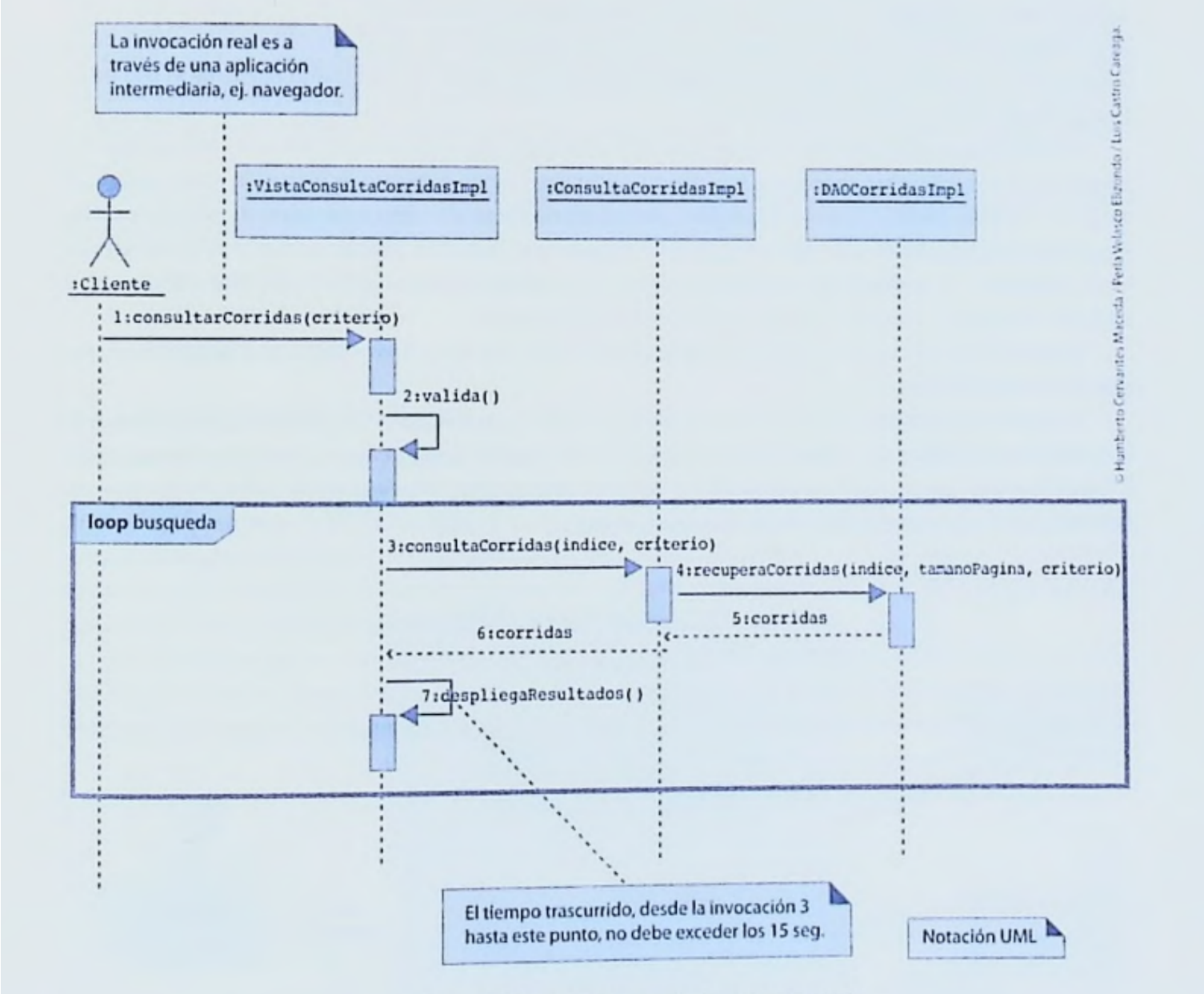
Organización interna, funcional y estructural.


«model» Multi-Layered Application
Users
External Systems
Presentation Layer
User Interface
Presentation Logic
Services Layer
Service Interfaces
Message Types
Cross Cutting
Security
Operational Management
Communication
Business Layer
Application Facade
Business Workflow
Business Components
Business Entities
Data Layer
Data Access
Service Agents
Data Sources
External Services
© uml-diagrams.org

Vista de Comportamiento

- **Pregunta:** ¿Cómo se comporta en ejecución?
- Entidades en tiempo de ejecución (Instancias, Procesos, Clientes).
- Propiedades: Atributos observables (Confiabilidad, Desempeño).
- Relaciones: Protocolos de comunicación (Conectores).

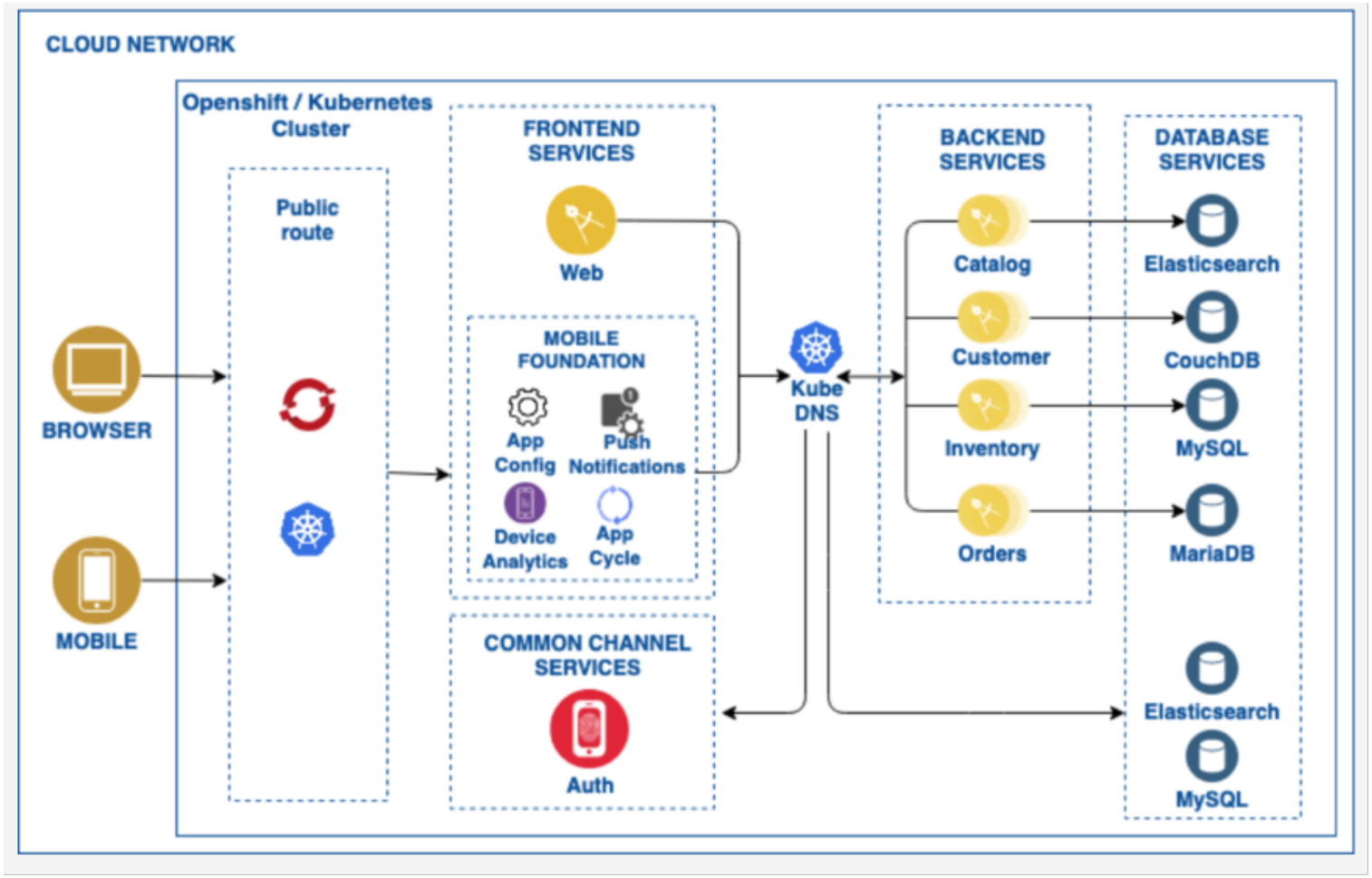
Interacción, flujo de comunicación y coordinación.


La invocación real es a través de una aplicación intermediaria, ej. navegador.
:Cliente
:VistaConsultaCorridasImpl
:ConsultaCorridasImpl
:DAOCorridasImpl
1:consultarCorridas(criterio)
2:valida()
loop busqueda
3:consultaCorridas(indice, criterio)
4:recuperaCorridas(indice, tamanoPagina, criterio)
5:corridas
6:corridas
7:despliegaResultados()
El tiempo trascurrido, desde la invocación 3 hasta este punto, no debe exceder los 15 seg.
Notación UML

Vista de Despliegue (Física)

- **Pregunta:** ¿Dónde se ejecuta? (Infraestructura).
- Elementos Físicos (Hardware: Servidores, Sensores).
- Software de soporte (SO, Contenedores).
- Asignación: ¿Qué software corre en qué hardware?
- Relaciones: *reside en*, *se ejecuta en*.

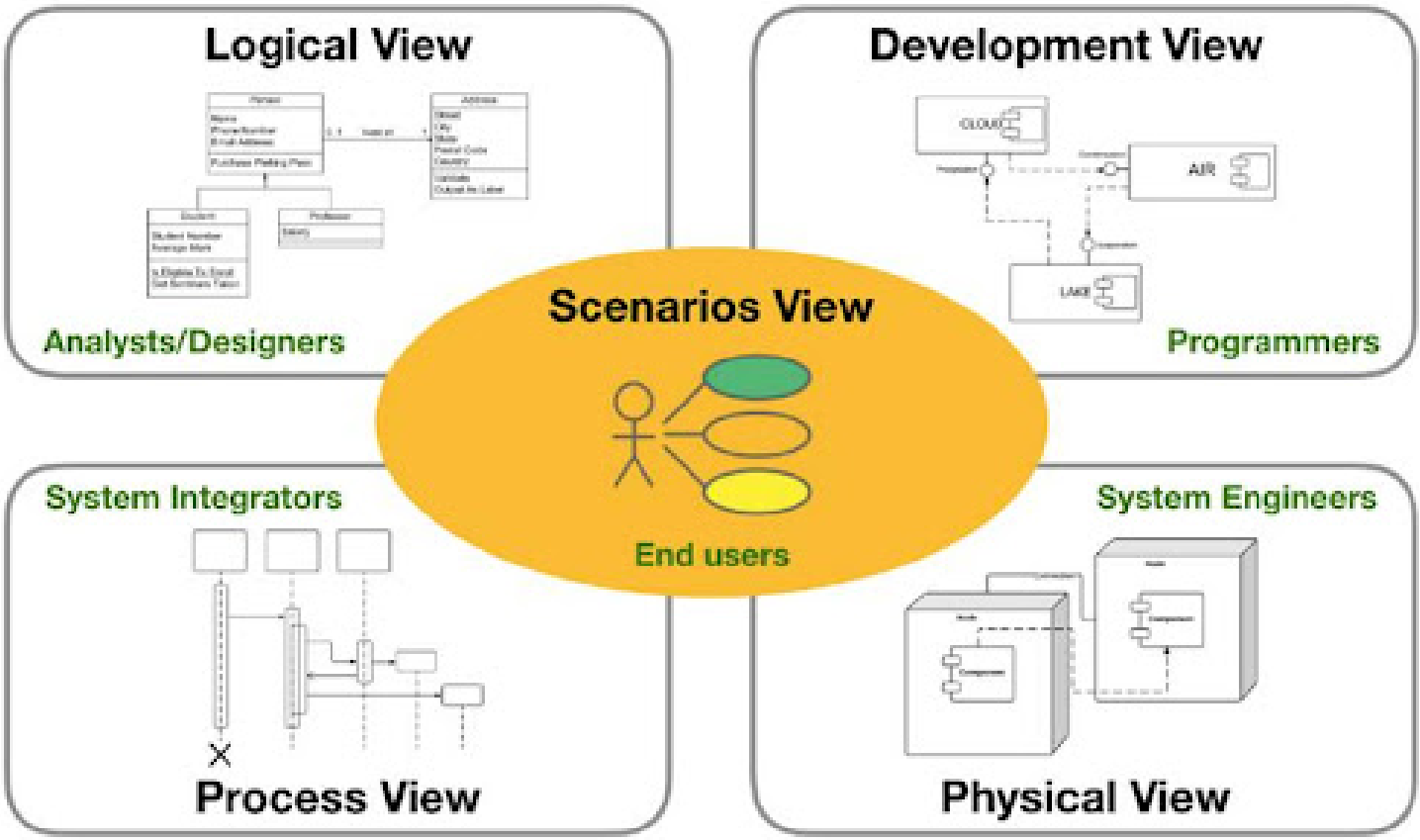
Asignación a la infraestructura. Planificación de despliegue.


CLOUD NETWORK
Openshift / Kubernetes Cluster
Public route
FRONTEND SERVICES
Web
MOBILE FOUNDATION
App Config
Push Notifications
Device Analytics
App Cycle
COMMON CHANNEL SERVICES
Auth
BROWSER
MOBILE
Kube DNS
BACKEND SERVICES
Catalog
Customer
Inventory
Orders
DATABASE SERVICES
Elasticsearch
CouchDB
MySQL
MariaDB
Elasticsearch
MySQL

Vista de Escenarios / 4+1

- Modelo 4+1 (Philippe Kruchten, 1995).
- **Lógica** (Usuarios).
- **Procesos** (Concurrencia, Rendimiento).
- **Desarrollo** (Desarrolladores).
- **Física** (Infraestructura).
- **"+1" (Escenarios):** Casos de uso que validan las otras 4.

Los escenarios integran y validan las demás vistas.


Logical View
Analysts/Designers
Development View
Programmers
Scenarios View
End users
System Integrators
Process View
System Engineers
Physical View

Panorama general

- Estilos arquitectónicos = Patrones probados.
- La elección depende de los requisitos (funcionales y no-funcionales).
- No existe la "mejor" arquitectura, solo la "adecuada".

Microservicios

- **Definición:** Servicios pequeños, independientes, APIs ligeras.
- **Ventajas:** Escalabilidad independiente, despliegues frecuentes, resiliencia.
- **Desventajas:** Alta complejidad operativa, monitoreo, transacciones distribuidas.
- **Uso:** Sistemas grandes, cloud-native.

Arquitectura en Capas

- **Definición:** Capas jerárquicas (Presentación, Negocio, Datos).
- **Ventajas:** Separación de responsabilidades, mantenibilidad.
- **Desventajas:** Rigidez, posible sobrecarga de comunicación.
- **Uso:** Aplicaciones empresariales tradicionales.

Arquitectura basada en Eventos (EDA)

- **Definición:** Comunicación asíncrona mediante eventos.
- **Ventajas:** Alta escalabilidad, desacoplamiento, reactividad.
- **Desventajas:** Complejidad en depuración y monitoreo, consistencia eventual.
- **Uso:** Sistemas en tiempo real, IoT, streaming.

Conclusión

- La arquitectura es una disciplina de decisión y **trade-offs** (compensaciones).
- No existe la "arquitectura correcta", solo la "adecuada" al contexto.
- Documentar (vistas) es clave para comunicar y evolucionar.