

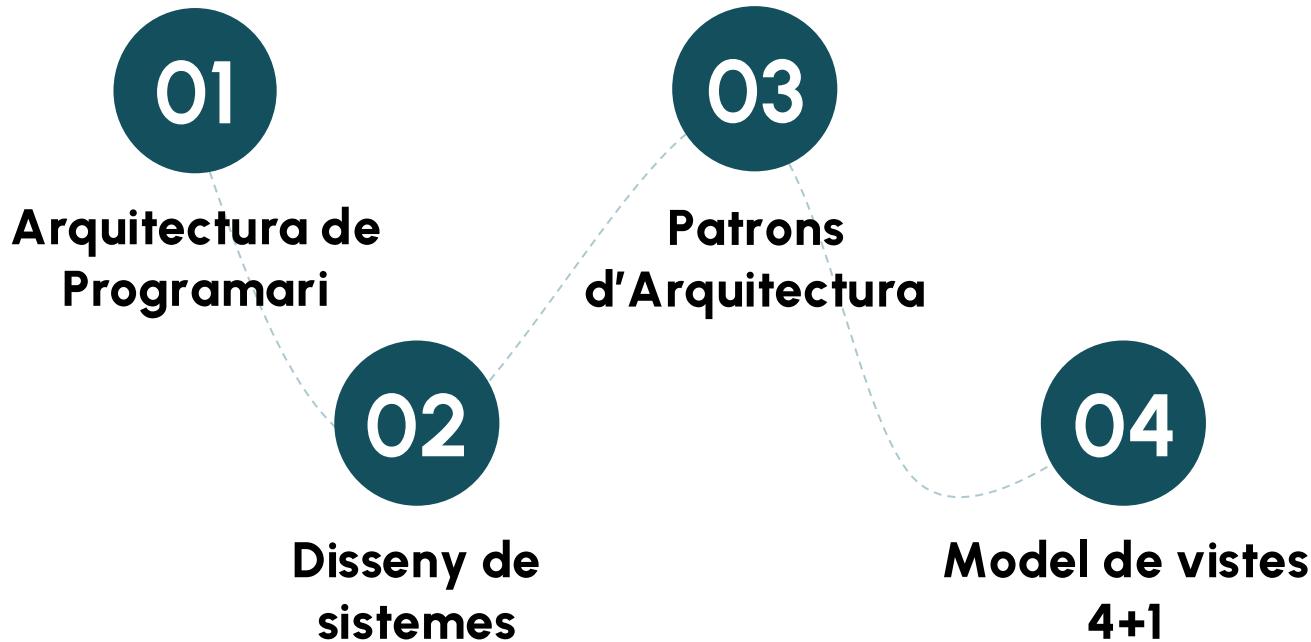
2

Tipologies de Disseny en l'Arquitectura de Programari

Oriol Alàs
Dídac Colominas
Èric Monné



ÍNDEX





01

Arquitectura de Programari vs Disseny de Sistemes

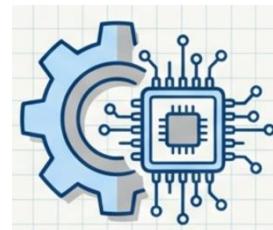
Arquitectura de Programari vs Disseny de Sistemes



El què

Defineix l'esquelet i els principis.

Microserveis és una decisió d'arquitectura

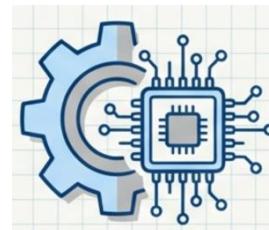


El com

Defineix els òrgans i com funcionen.

Quins endpoints, quin esquema de dB, protocols...

Arquitectura de Programari vs Disseny de Sistemes



El què

El com

Enfocament

Patrons de disseny,
contractes entre
components i dependències

Fluxos de dades, detalls
d'implementació i interfícies
de components

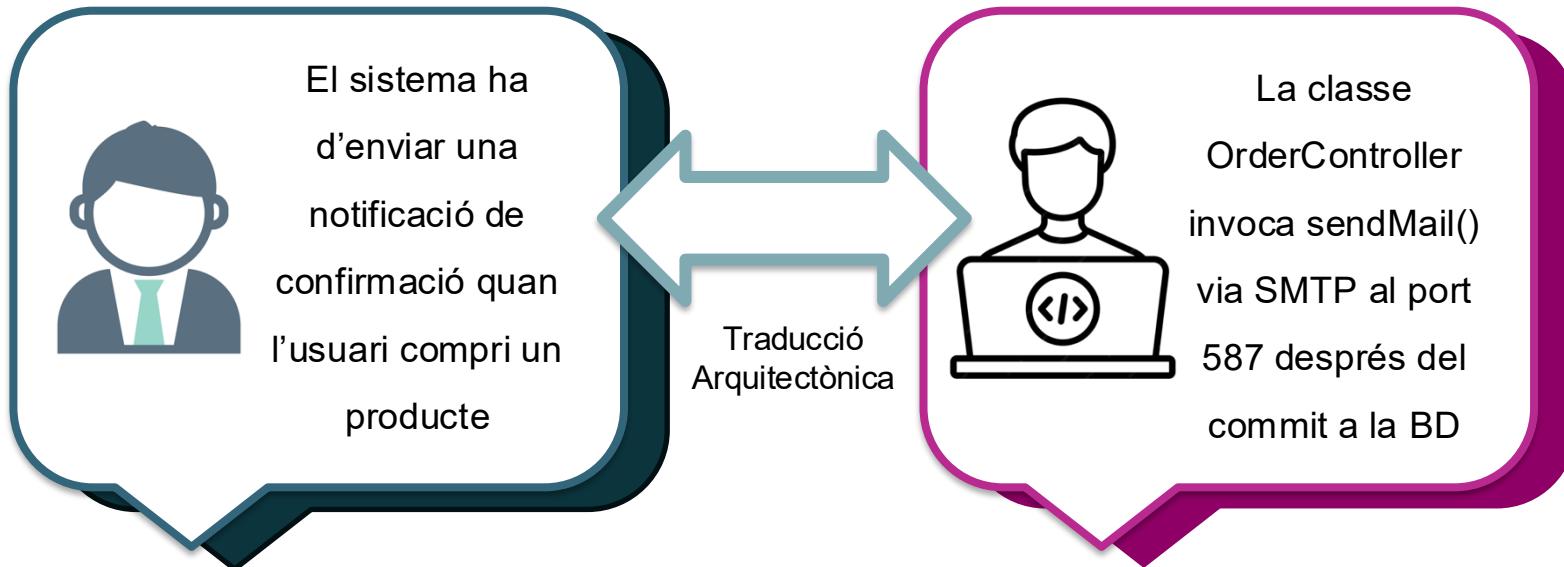
Preocupacions

Seguretat, mantenibilitat

Infraestructura, balanceig de
càrrega, latència i execució



Llenguatge Conceptual vs Tècnic



La taxonomia del disseny: Tres Nivells d'Abstracció

1. Disseny Conceptual

Perspectiva del propietari

Defineix el "Què" i el "Per què"
Llenguatge de negoci

*Exemple: Magatzem
de Dades de clients*

2. Disseny Lògic

Perspectiva del dissenyador

Estructura funcional
Agnòstic de la plataforma

*Exemple: Diagrames
UML, classes...*

3. Disseny Físic

Perspectiva del constructor

Desplegament
tecnològic real

*Exemple: Servidor
MySQL v8.0 en AWS*



La taxonomia del disseny: Tres Nivells d'Abstracció

1. Disseny Conceptual

Perspectiva del propietari

Defineix el "Què" i el "Per què"
Llenguatge de negoci

2. Disseny Lògic

Perspectiva del dissenyador

Estructura funcional
Agnòstic de la plataforma

3. Disseny Físic

Perspectiva del constructor

Desplegament
tecnològic
real

A mesura que baixem,
augmenta el
compromís tecnològic
i disminueix la
flexibilitat per canviar
decisions sense cost





02

Patrons d'Arquitectura

Tipus de patrons d'arquitectura

Patrons estructurals

Aquests defineixen la forma "macro" del programari (Capa, Monòlit modular, microserveis...)

Patrons orientats a domini

Aquests es fan servir per evitar la lògica de negoci quedí contaminada (Clean Architecture, Onion Architecture, Hexagonal)

Patrons d'integració

Aquests s'utilitzen per definir com parlen els components (Event-driven, SAGA, Message Queue, API Gateway)

Patrons de presentació

Com presentar el nostre projecte (MVC, MVT, MVVM)

Patrons de dades

Com gestione la persistència (ORMs, Repository, Database per service)



Tipus de patrons d'arquitectura

Patrons estructurals

Aquests defineixen la forma "macro" del programari (Monòlit modular, microserveis...)

Monòlit

Un sol deploy

Monòlit modular

Un deploy, però amb mòduls ben separat

Microserveis

Serveis petits, desplegables independentment

SOA

Similar a microserveis, però més legacy enterprise



Tipus de patrons d'arquitectura

Patrons de domini

Aquests es fan servir per evitar la lògica de negoci quedí contaminada (Clean Architecture, Onion Architecture, Hexagonal)

Onion
Architecture

p.e, Transport → Domini / Lògica → Repositoris

Clean
Architecture

Capes concèntriques com una ceba

Hexagonal
Architecture

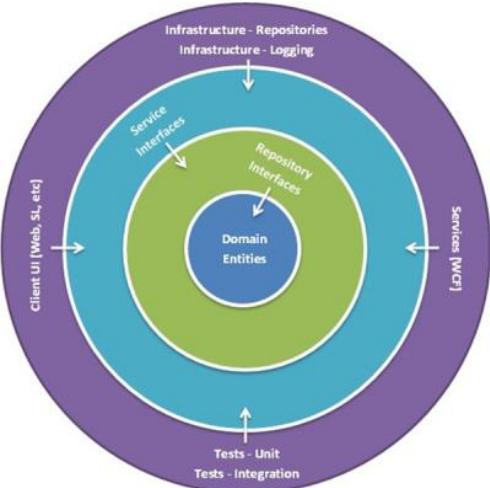
Un “core” (domini) al centre i diversos costats per connectar-hi adaptadors. Necessites que totes les característiques de negoci estiguin definides per adelantat.

Domain-Driven
Design

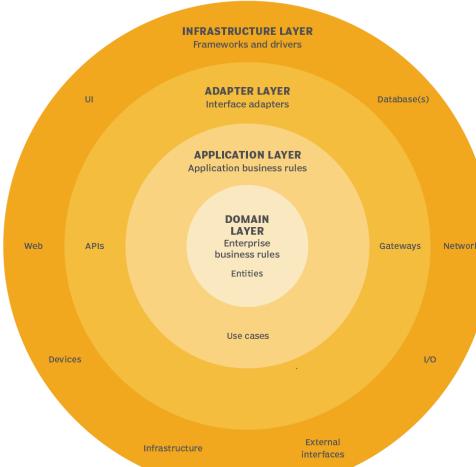
Idea comuna: El domini ha d'estar independent de framework, BD, UI...



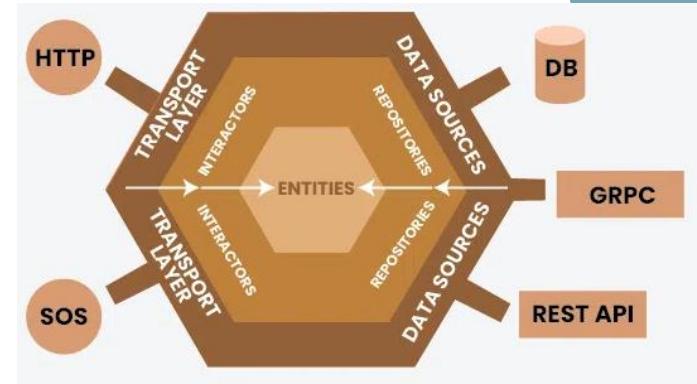
Tipus de patrons d'arquitectura



Les capes externes
depenen de les
internes



Afegeix rols més
explícits (interface
adapters) per posar
èmfasi en com entra i
surt informació



Posa tots els detalls externs
(web, DB, cues) com a
adaptadors intercanviables.
Canviar de ORM sense tocar
el use-case



Tipus de patrons d'arquitectura

Patrons d'integració

Aquests s'utilitzen per definir com parlen els components (Event-driven, SAGA, Message Queue, API Gateway)

Event-driven

Un estil d'arquitectura on els components publiquen esdeveniments (fets que han passat) i altres components reaccionen

Message Queue

Una infraestructura (component de middleware) que guarda missatges fins que un worker/consumidor els processa.

Api Gateway

Un **punt d'entrada únic** per a clients (web/mòbil/partners) cap als serveis interns



Tipus de patrons d'arquitectura

Patrons de presentació

Aquests s'utilitzen per definir com parlen els components (Event-driven, SAGA, Message Queue, API Gateway)

MVC

Separà dades (Model), interfície (View) i coordinació/entrada d'usuari (Controller).

MTV

Variant de MVC pròpia de Django: Model – Template (HTML) – View (Lògica).

MVVM

Model – View – ViewModel : El VM exposa estat i accions perquè la View s'hi enllaci (binding).





03

Model de vistes 4+1

Model de vistes 4+1

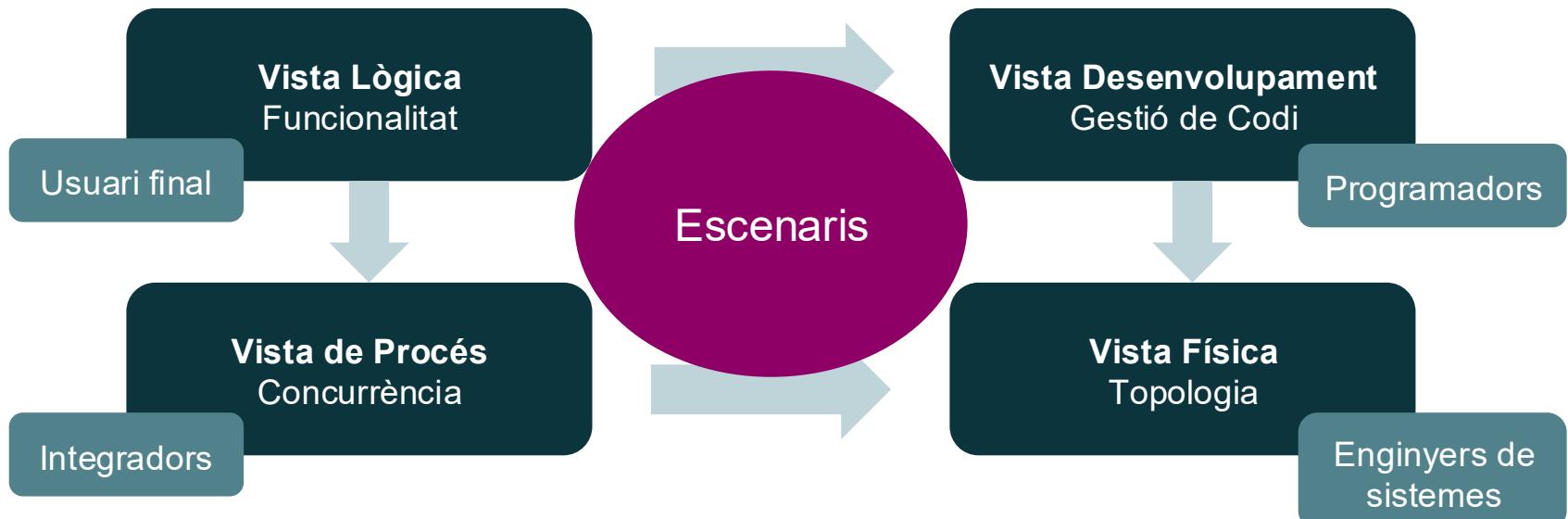
Anomenat Model Kruchten, ideat el 1995.

- Atendre les preocupacions de diversos grups d'interessats (stakeholders) de manera separada.



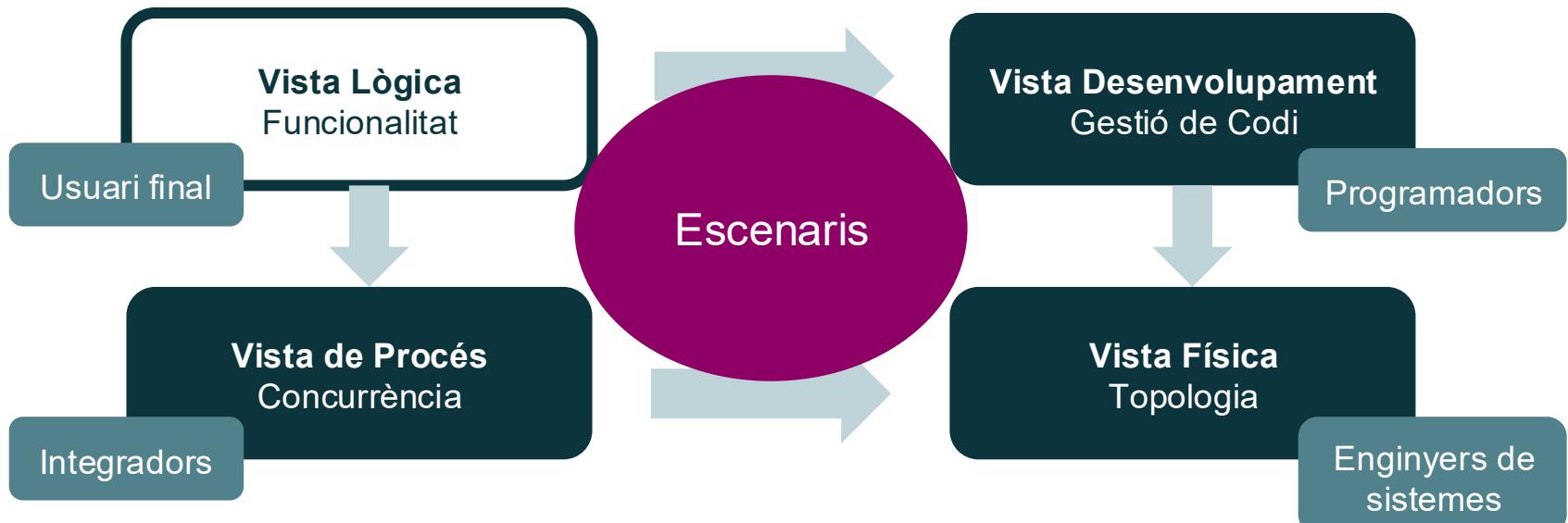
Model de vistes 4+1

L'arquitectura d'un sistema complex no es pot representar en un únic plàtol. Kruchten proposa 5 vistes concurrents per satisfer els stakeholders.



Model de vistes 4+1

L'arquitectura d'un sistema complex no es pot representar en un únic plàtol. Kruchten proposa 5 vistes concurrents per satisfer els stakeholders.



Vista Lògica

Objectiu

Es centra en els requisits funcionals. Identificar mecanismes i elements de disseny comuns.

Components

Classes, objectes i les seves relacions

Notació

Diagrames de classe, d'objectes i d'estats en UML

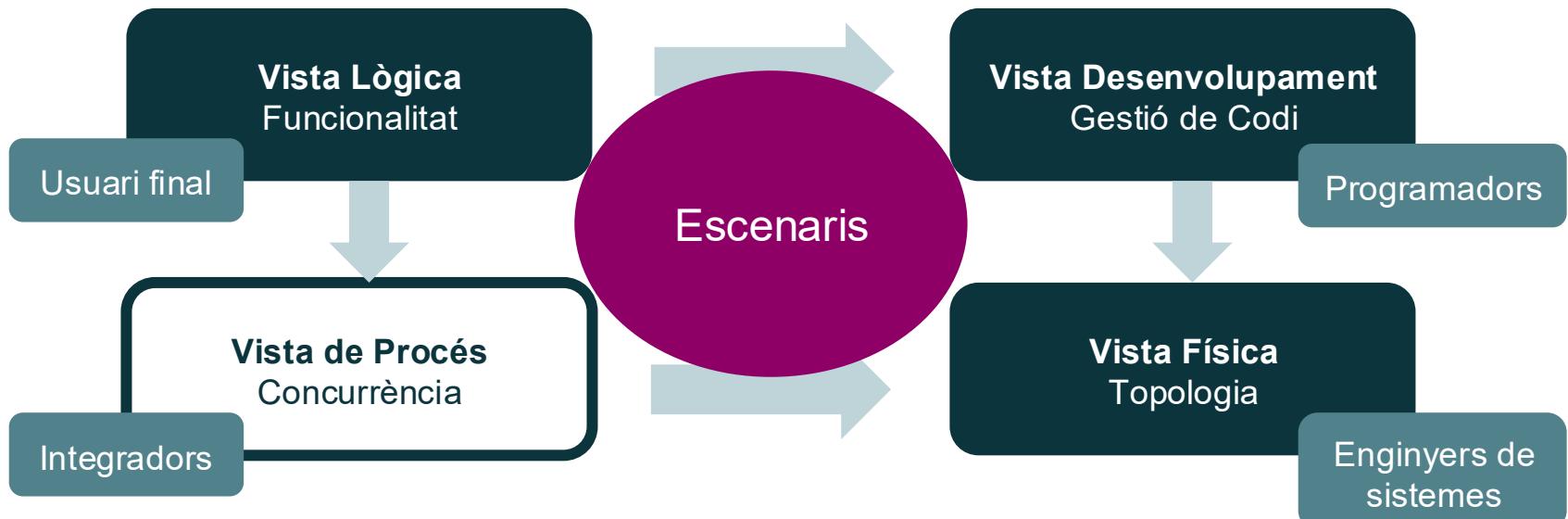
A qui s'adreça?

Usuaris finals



Model de vistes 4+1

L'arquitectura d'un sistema complex no es pot representar en un únic plàtol. Kruchten proposa 5 vistes concurrents per satisfer els stakeholders.



Vista Procés

Objectiu

Aborda els requisits no funcionals com el **rendiment**, la disponibilitat, la concorrència i la tolerància a fallades

Components

Processos i tasques (fils de control o *threads*)

Notació

Diagrames UML dinàmics com els d'**activitat**, **seqüència** i **comunicació**.

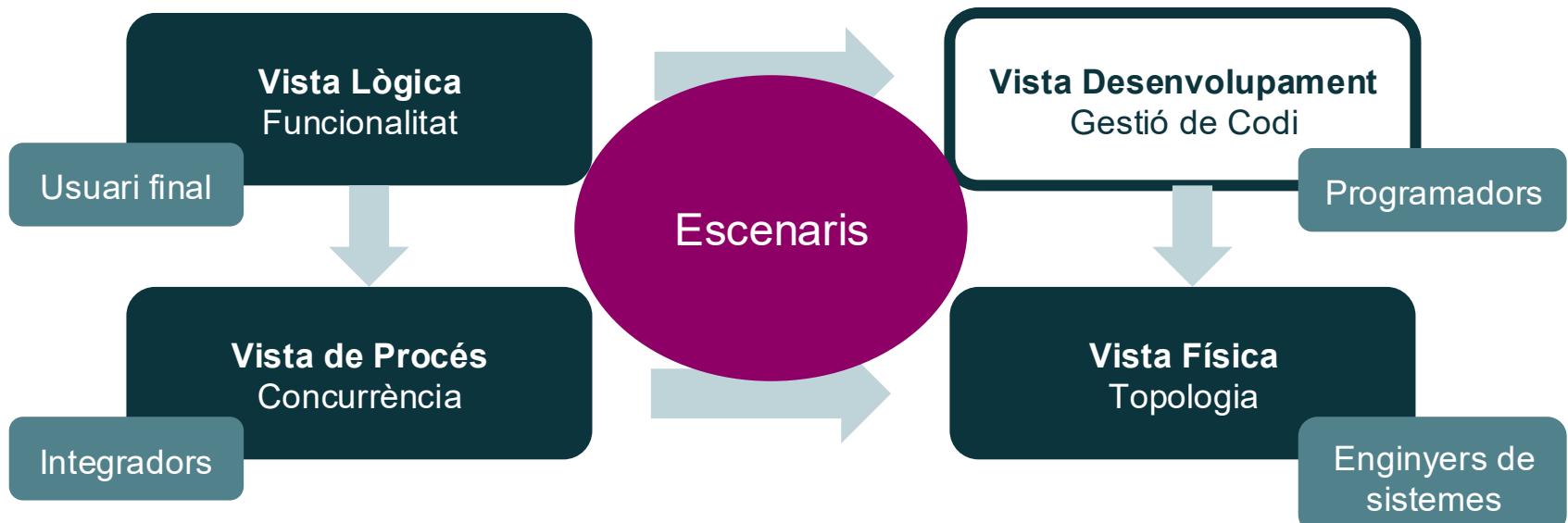
A qui s'adreça?

Integradors de sistemes



Model de vistes 4+1

L'arquitectura d'un sistema complex no es pot representar en un únic plàtol. Kruchten proposa 5 vistes concurrents per satisfer els stakeholders.



Vista Desenvolupament

Objectiu

Descriu l'organització estàtica del programari en el seu entorn de construcció

Components

Mòduls, subsistemes, llibreries i paquets de programari

Notació

Mitjançant diagrames de **paquets** i de **components** de l'UML.

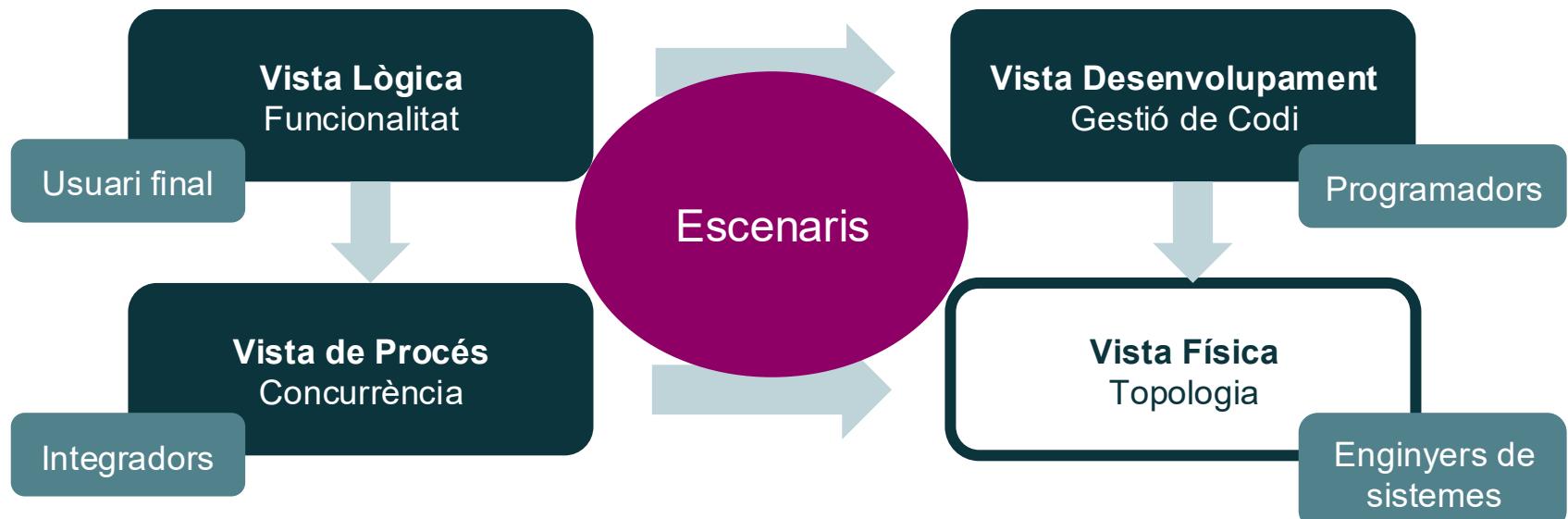
A qui s'adreça?

Als **programadors** i gestors de programari



Model de vistes 4+1

L'arquitectura d'un sistema complex no es pot representar en un únic plàtol. Kruchten proposa 5 vistes concurrents per satisfer els stakeholders.



Vista Física

Objectiu

Descriu el mapeig del programari sobre el maquinari

Components

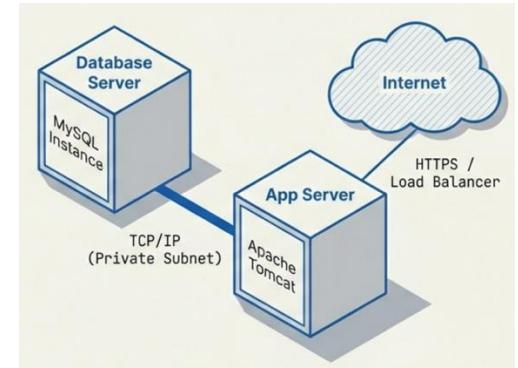
Nodes (nodes de processament), processadors, dispositius i línies de comunicació

Notació

Principalment amb el diagrama de **desplegament** de l'UML

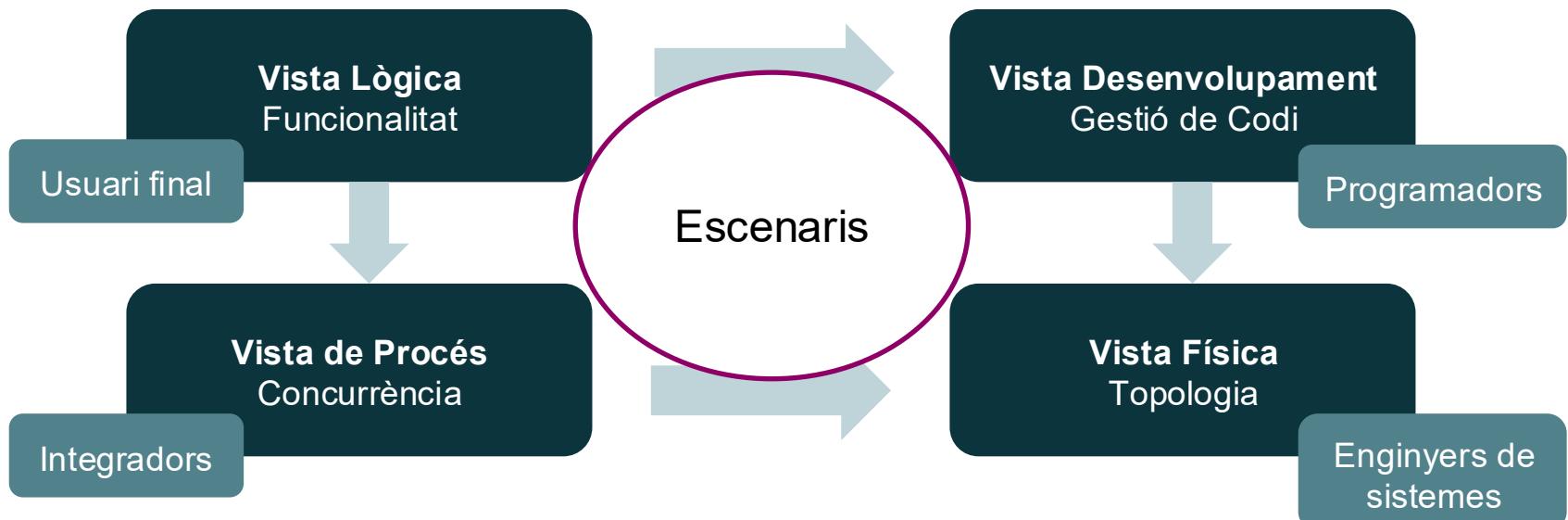
A qui s'adreça?

Als **enginyers de sistemes** i administradors



Model de vistes 4+1

L'arquitectura d'un sistema complex no es pot representar en un únic plàtol. Kruchten proposa 5 vistes concurrents per satisfer els stakeholders.



Escenaris

Objectiu

Actua com la vista que **uneix les altres quatre**

Components

Seqüències d'interaccions entre objectes i entre processos i instàncies

Notació

Diagrames de casos d'ús i diagrames d'interacció

A qui s'adreça?

A tots els stakeholders



Beneficis

Claredat

El model "4 + 1" ofereix una manera clara i estructurada de visualitzar i comunicar diferents aspectes de l'arquitectura del sistema.

Alineació

El model ajuda a alinear l'arquitectura tècnica amb els requisits de l'usuari, garantint que el sistema compleixi el seu propòsit previst.

Facilitat de Comunicació

Cada vista s'adapta a un públic específic, cosa que facilita la comunicació de detalls tècnics a les parts interessades amb diferents nivells d'experiència.

Exhaustivitat

En dividir l'arquitectura en cinc vistes diferents, garanteix que es tinguen en compte tots els aspectes essencials del sistema.

Eficiència

El model ajuda a un desenvolupament més eficient, ja que proporciona una guia per al disseny i la implementació tant d'alt com de baix nivell.



Diagrams

Logical View

Class Diagram, Object Diagram,
Component Diagram, Package Diagram,
Composite Structure Diagram

Process View

Activity Diagram, State Machine
Diagram, Sequence Diagram, Timing
Diagram, Interaction Overview Diagram

Physical View

Deployment Diagram

Development View

Component Diagram, Package Diagram



Exemple

L'usuari prem Play i veu el capítol

L'usuari vol iniciar la reproducció d'un capítol (o reprendre'l) amb el mínim temps d'espera i amb qualitat adaptada a la seva connexió.

Precondicions

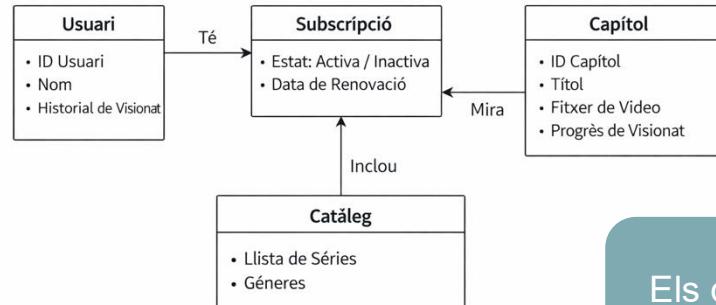
- L'usuari està autenticat (o pot autenticar-se).
- El capítol existeix al catàleg i és disponible a la seva regió.
- L'usuari té una subscripció activa (o un dret de visualització vàlid).
- El dispositiu té connectivitat a Internet.



Lògic

Exemple

Classes com **Usuari** (dades del perfil), **Subscripció** (si ha pagat o no), **Capítol** (el fitxer i les seves metadades) i **Catàleg** (la llista de sèries)



Els diagrames UML no
han de ser tècnics

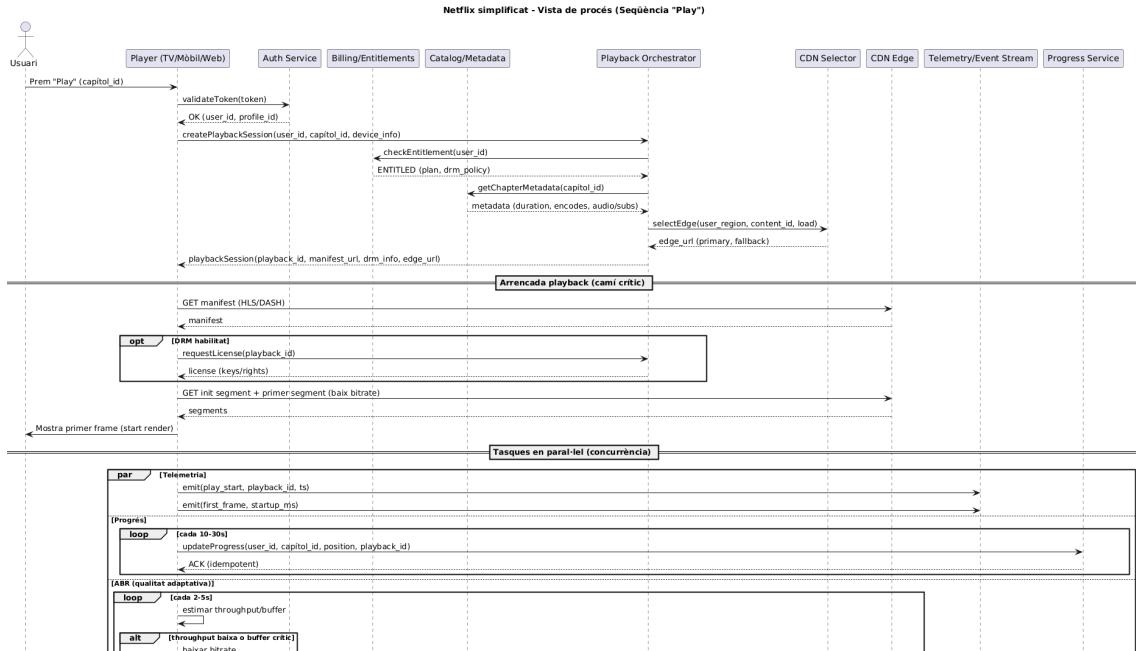


Lògic

Procés

Exemple

Quan l'usuari mira el vídeo, diversos processos treballen alhora: un gestiona el **flux de dades (streaming)**, un altre ajusta la qualitat segons la velocitat d'internet i un altre registra el progrés de visionat en segon pla...



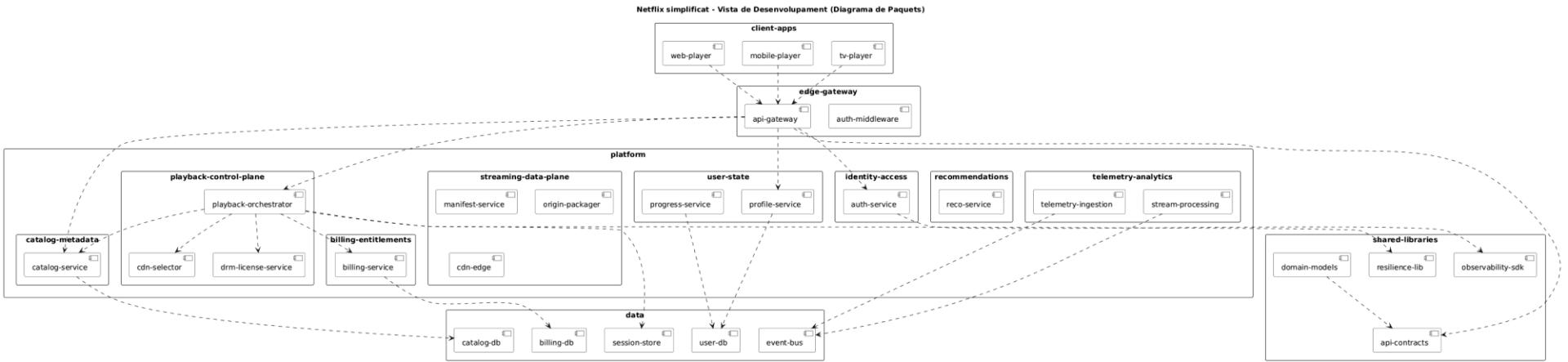
Lògic

Procés

Desenvolupament

Exemple

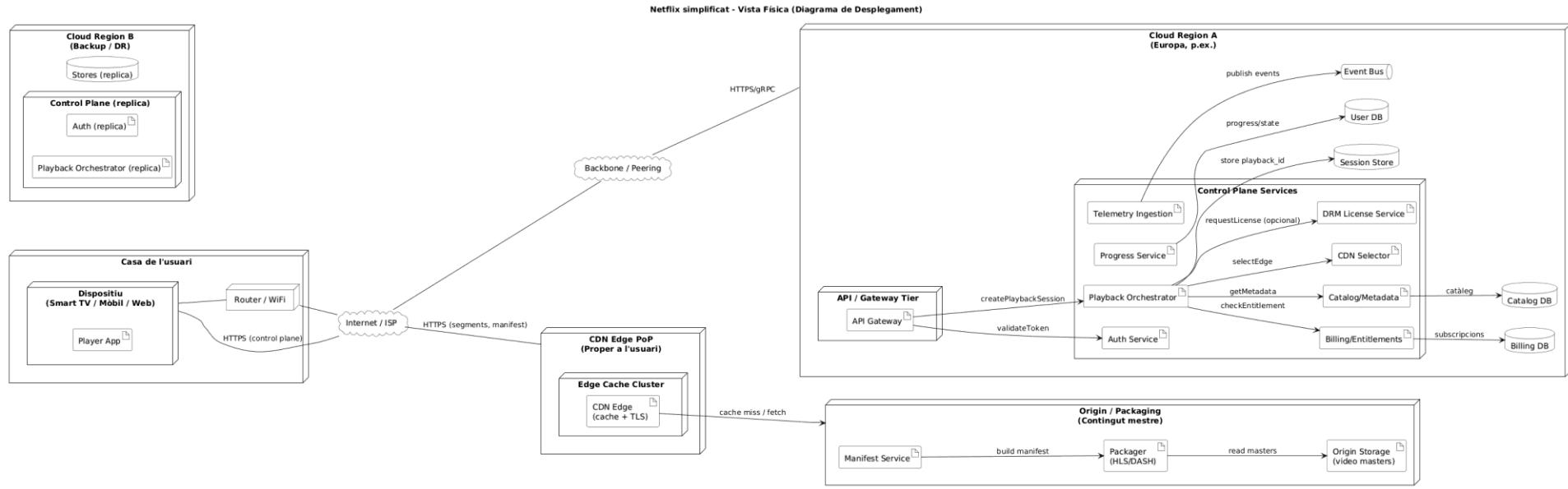
El codi s'organitza en **microserveis** independents: el servei de "Reproducció de Vídeo", el servei de "Facturació" i el servei de "Motor de Recomanacions"



Exemple

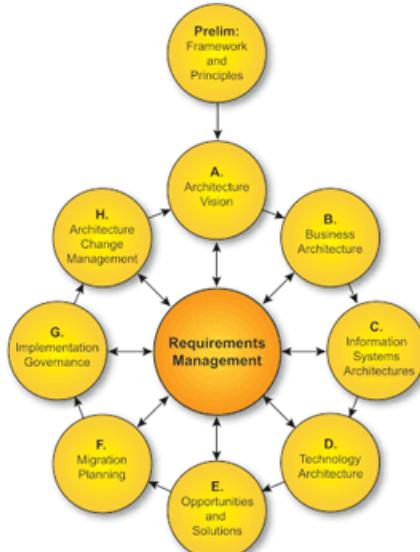
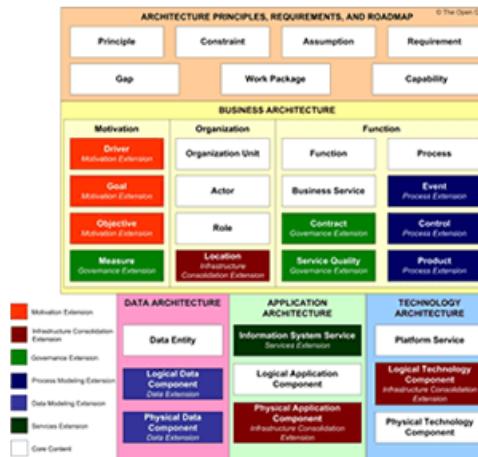
Físic

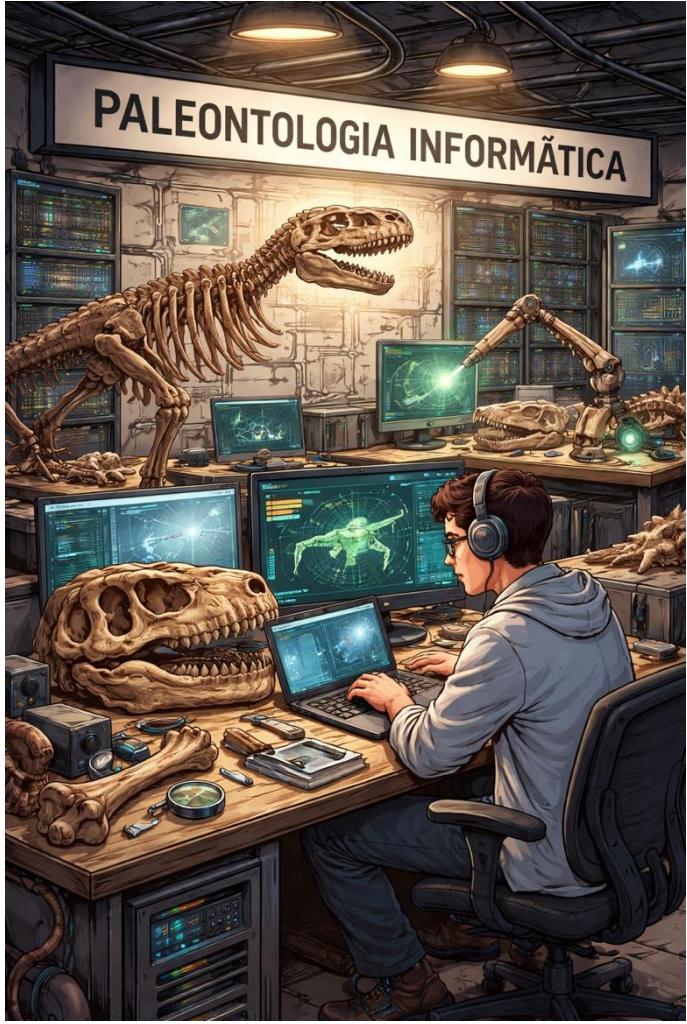
Netflix no envia el vídeo des d'un sol lloc centralitzat, sinó que utilitza una xarxa de servidors anomenada **CDN** repartida per tot el món per estar físicament prop de casa de l'usuari



Altres dissenys d'arquitectures

TOGAF®





Disseny en Entorns Àgils

L'arquitectura ha de ser contínua i evolutiva

Emergent vs. Intencional

- Emergent: Decisions de l'equip en sprints
- Intencional: Guies estratègiques i sincronització



Eines pel disseny d'arquitectures

