40431: Modelação e Análise de Sistemas

Arquitetura Evolutiva - complemento

Ilídio Oliveira

v2020-01-08 | TP10c (formato videoaula disponível)

universidade
departamento de e
telecomunicações e in

Estilos de arquitetura

Apesar da enorme diversidade dos produtos de software, é possível falar-se em certos "padrões de arquitetura" nas soluções empresariais

Um "padrão de arquitetura" reflete uma abordagem-tipo, i.e., uma maneira de organizar a solução que se provou adequada para certos tipos de projetos.

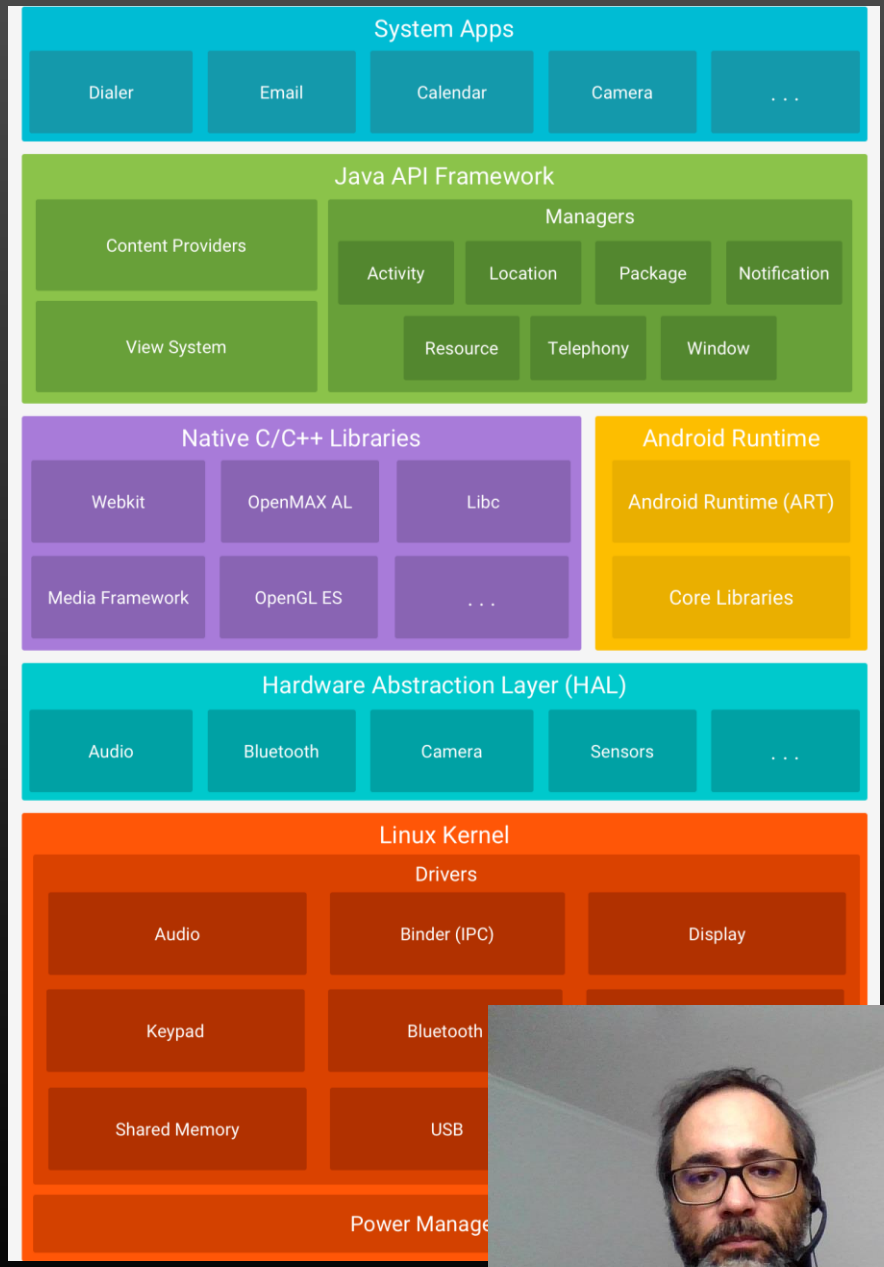
Exemplos:

- Microkernel Architecture
- Layered Architecture
- Event-Driven Architecture
- Service-oriented Architecture
- Microservices Architecture
- ...

Qual é o “padrão”?

Uma arquitetura por camadas

- Camadas “debaixo” fornecem serviços às camadas “de cima”
- Cada camada comunica apenas com as camadas adjacentes (consome serviços da de baixo, oferece serviços à de cima)
- Há geralmente um crescendo no nível de abstração: camadas abaixo mais próximas do hardware/armazenamento; camada superior voltada para o utilizador.
- Cada camada pode ser organizada em módulos, mostrando componentes no mesmo nível de abstração



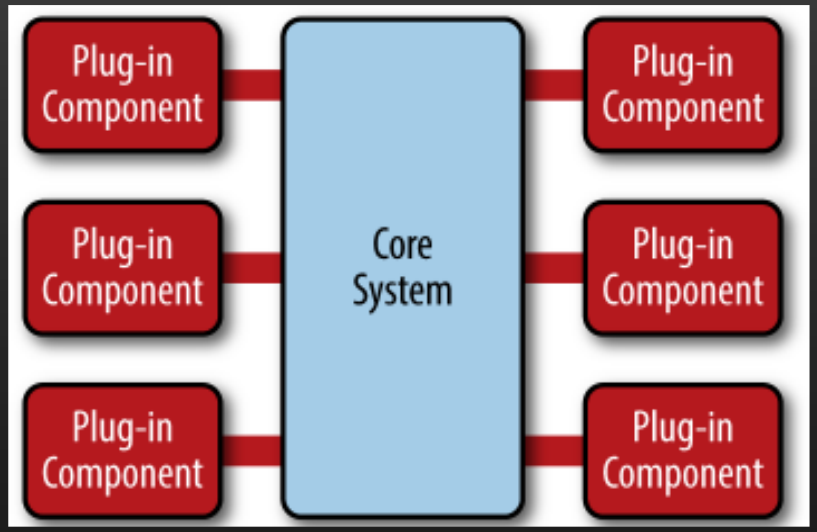
https://developer.android.com

Arquitetura *Microkernel*

Núcleo da aplicação (estável) **+** ***plugins*** (dinâmico)

Novas funcionalidades da aplicação são adicionadas como plug-ins

Vantagens: extensibilidade, separação de assuntos e isolamento.



Processamento de eventos (*event-driven*)

O padrão de “arquitetura por eventos” é um estilo de arquitetura assíncrono e distribuído, usado em aplicações que precisam de ser muito escaláveis (e.g.: processar milhares de leituras de sensores por segundo)

A arquitetura por eventos é composta por componentes de processamento de eventos altamente dissociados (*decoupled*) e especializados (dedicados a um tipo de processamento) que recebem e processam os eventos.

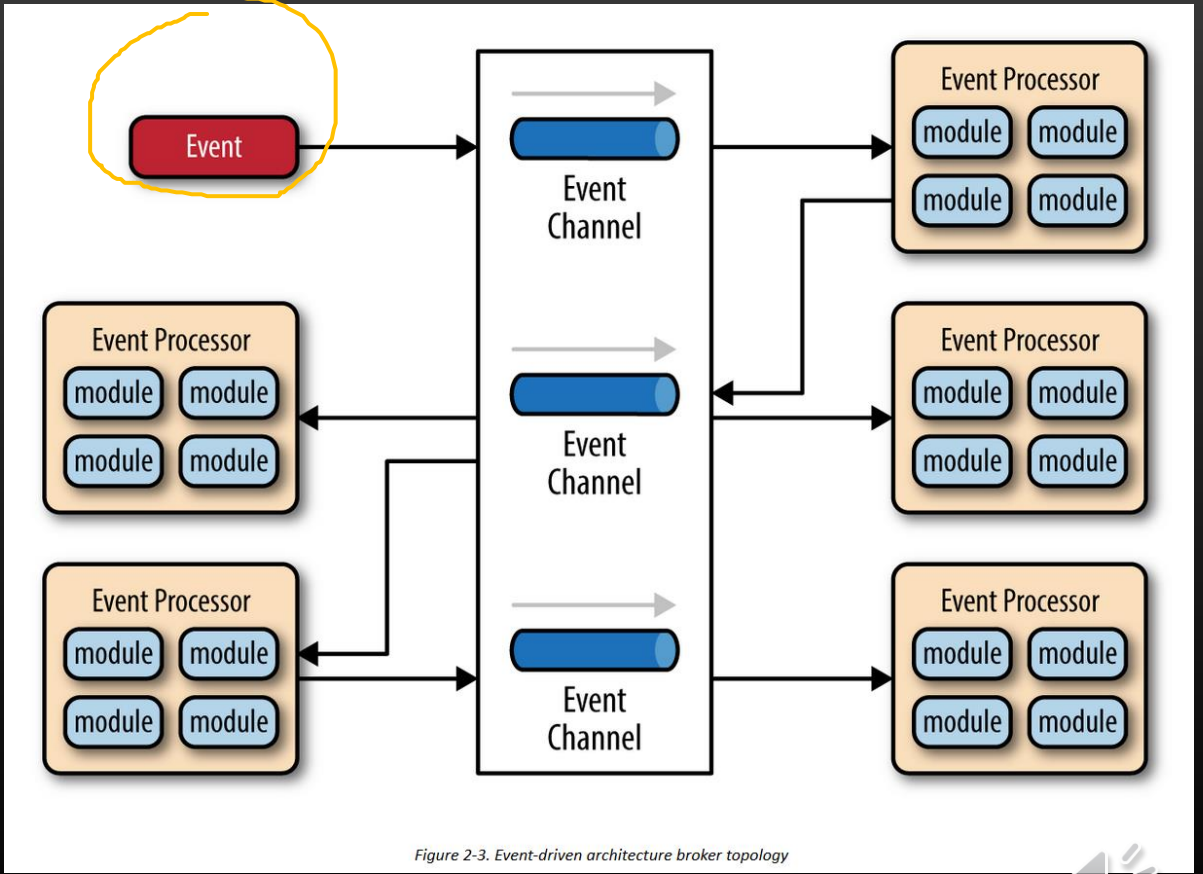


Figure 2-3. Event-driven architecture broker topology

https://www.oreilly.com/library/view/software-architecture-patterns/9781491971437/ch02.html

Um exemplo da indústria: uma solução de M2M (*machine-to-machine*)

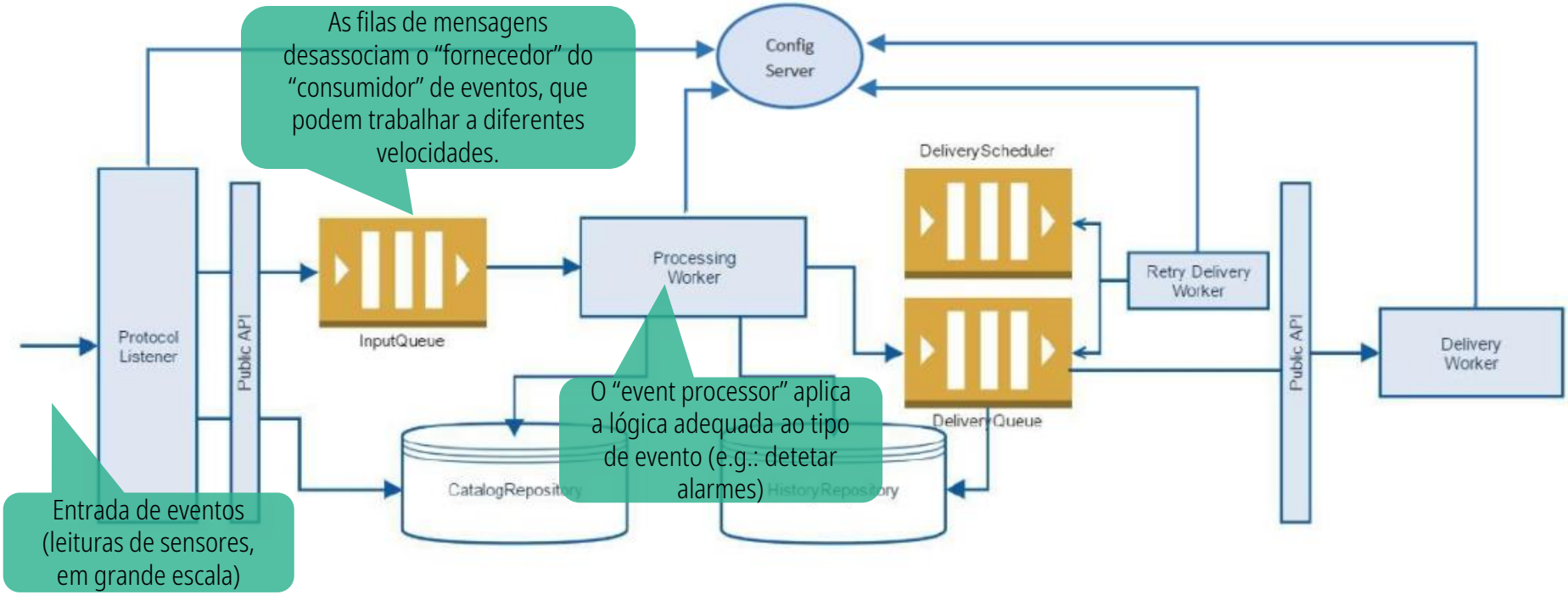


Figura 5.2: Arquitetura e componentes da SmartIOT.

Mais info: https://repositorio-aberto.up.pt/handle/10216/109650

Arquitetura por camadas

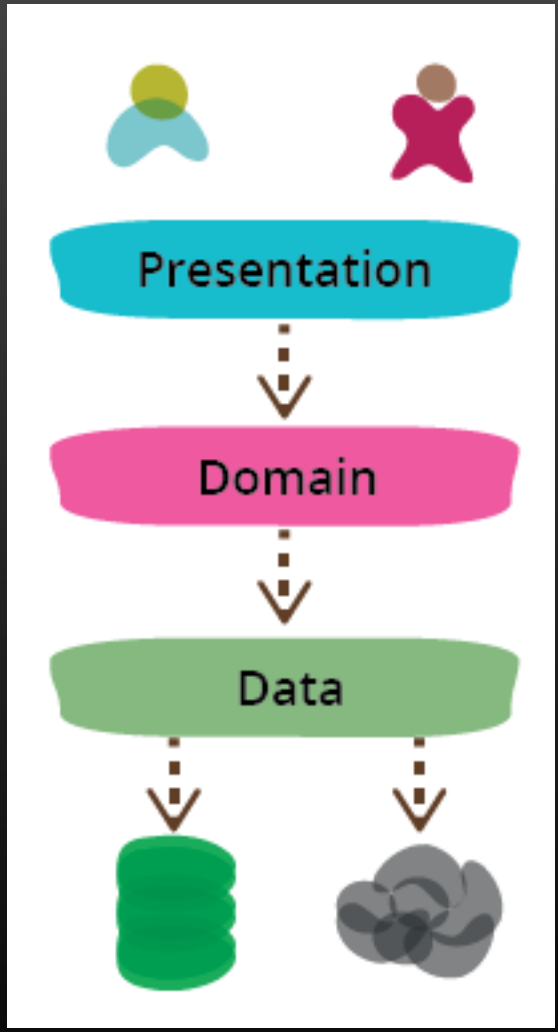
Divisão modular da solução de software em camadas/níveis de abstração.

As camadas são sobrepostas

Cada camada tem uma especialização

Camadas “em cima” pedem serviços às camadas “de baixo”

Não se pode saltar camadas: os componentes, em cada camada, “falam” com as camadas adjacentes.



https://martinfowler.com/bliki/PresentationDomainDataLayering.html

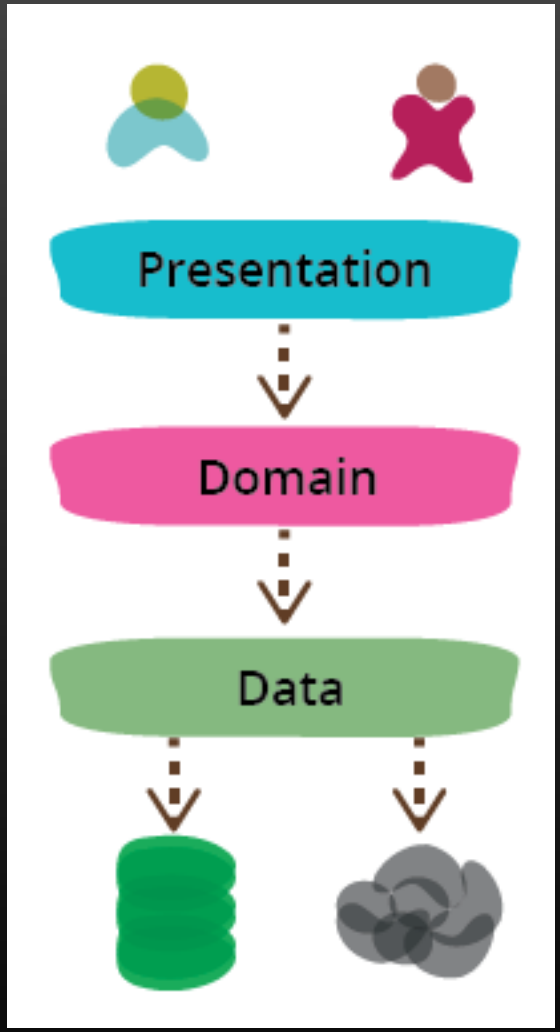
A arquitetura de 3 camadas

Uma das formas mais comuns de modularizar um programa orientado para a gestão de informação é separá-lo em três camadas principais:

1) apresentação (*user interface*, UI),
2) lógica de domínio (a.k.a. *business logic*)
3) acesso a dados.

Desta forma, é normal organizar uma aplicação web em três camadas:

- Uma camada web que sabe lidar com pedidos HTTP e preparar as páginas HTML,
- uma camada com a lógica de negócio, que contém regras (algoritmos), validações e cálculos,
- e uma camada de acesso de dados que assegura como gerir os dados de forma persistente, numa base de dados ou com integração de serviços remotos.



https://martinfowler.com/bliki/PresentationDomainDataLayering.html

GUI windows
reports
speech interface
HTML, XML, XSLT, JSP, Javascript, ...

UI
(AKA Presentation, View)

handles presentation layer requests
workflow
session state
window/page transitions
consolidation/transformation of disparate data for presentation

Application
(AKA Workflow, Process, Mediation, App Controller)

handles application layer requests
implementation of domain rules
domain services (POS, Inventory)
- services may be used by just one application, but there is also the possibility of multi-application services

Domain
(AKA Business, Application Logic, Model)

very general low-level business services used in many business domains
CurrencyConverter

Business Infrastructure
(AKA Low-level Business Services)

(relatively) high-level technical services and frameworks
Persistence, Security

Technical Services
(AKA Technical Infrastructure, High-level Technical Services)

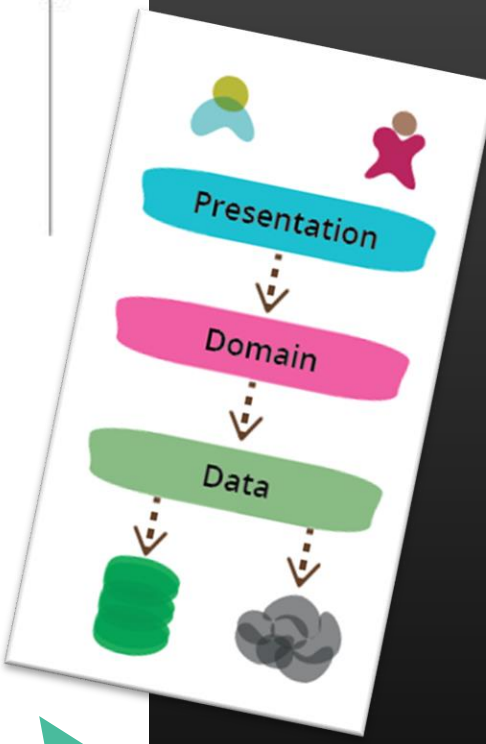
low-level technical services, utilities, and frameworks
data structures, threads, math, file, DB, and network I/O

Foundation
(AKA Core Services, Base Services, Low-level Technical Services/Infrastructure)

width implies range of applicability

more app specific

dependency



Algumas arquiteturas baseiam-se no modelo três camadas que ampliam para mostrar uma maior especialização de responsabilidades

[Larman'04]

Camadas e partições (modularização)

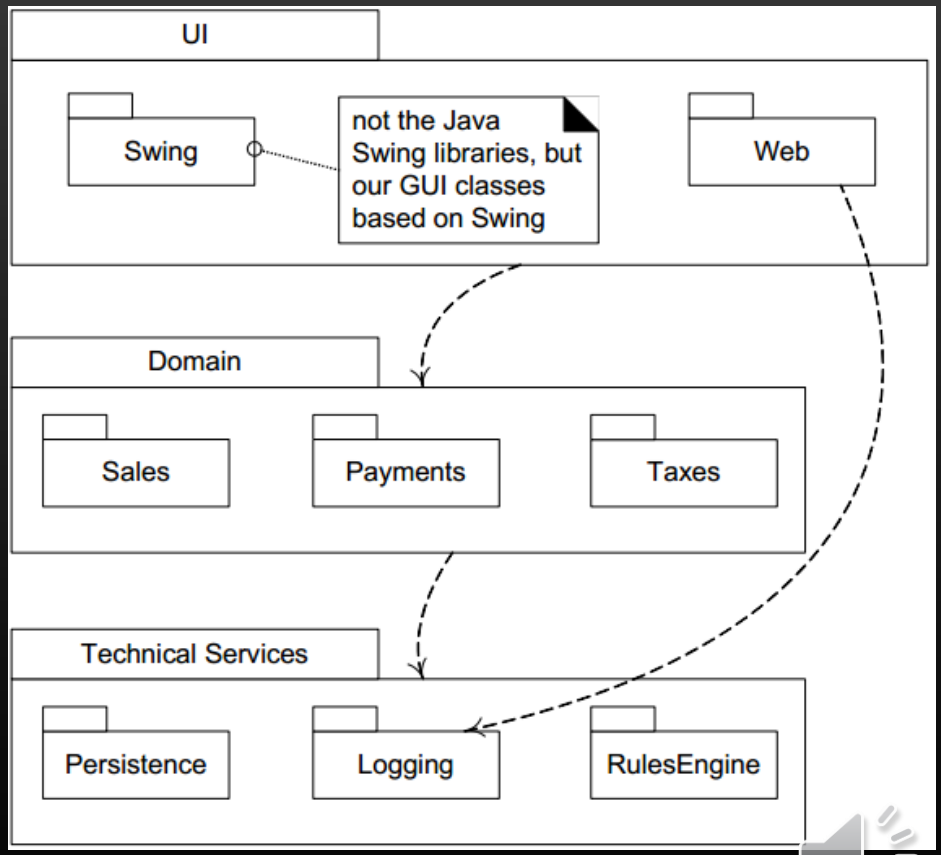
Camadas verticais:

- divisão por níveis de abstração

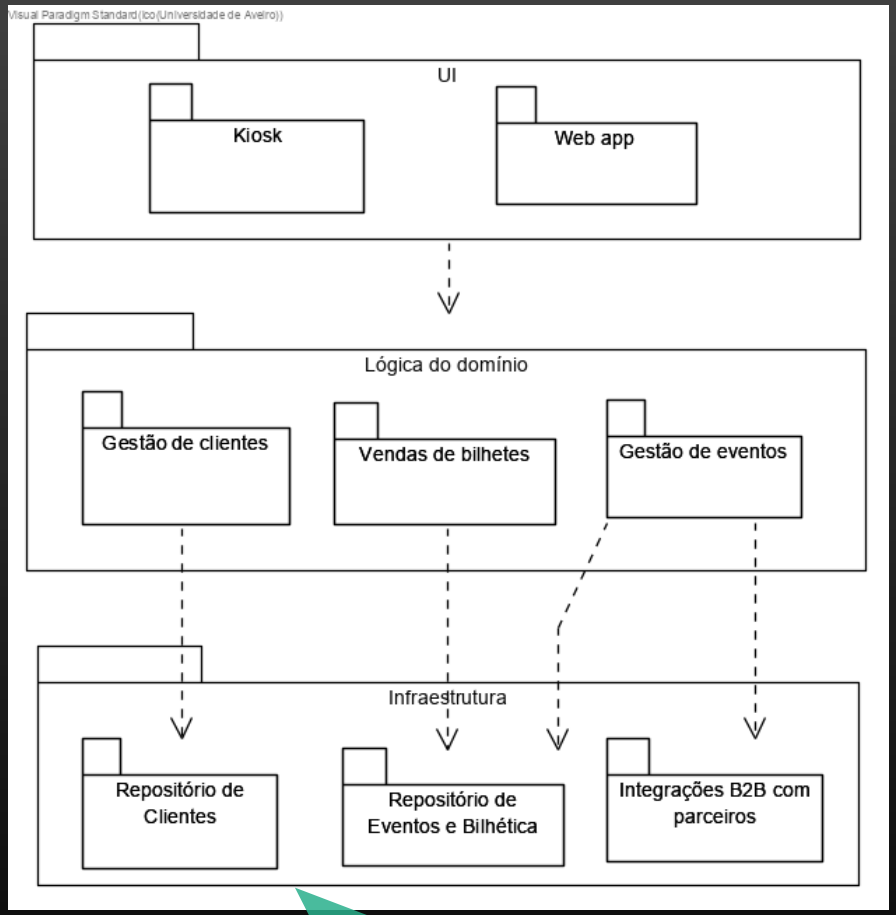
Partições horizontais:

- Módulos dentro de uma camada

E.g.: a lógica do domínio está dividida em grandes módulos funcionais especializados, agrupando as programação relativa às Vendas, aos Pagamentos e à Fiscalidade.



Considerar o uso de arquiteturas lógicas nos projetos de grupo.



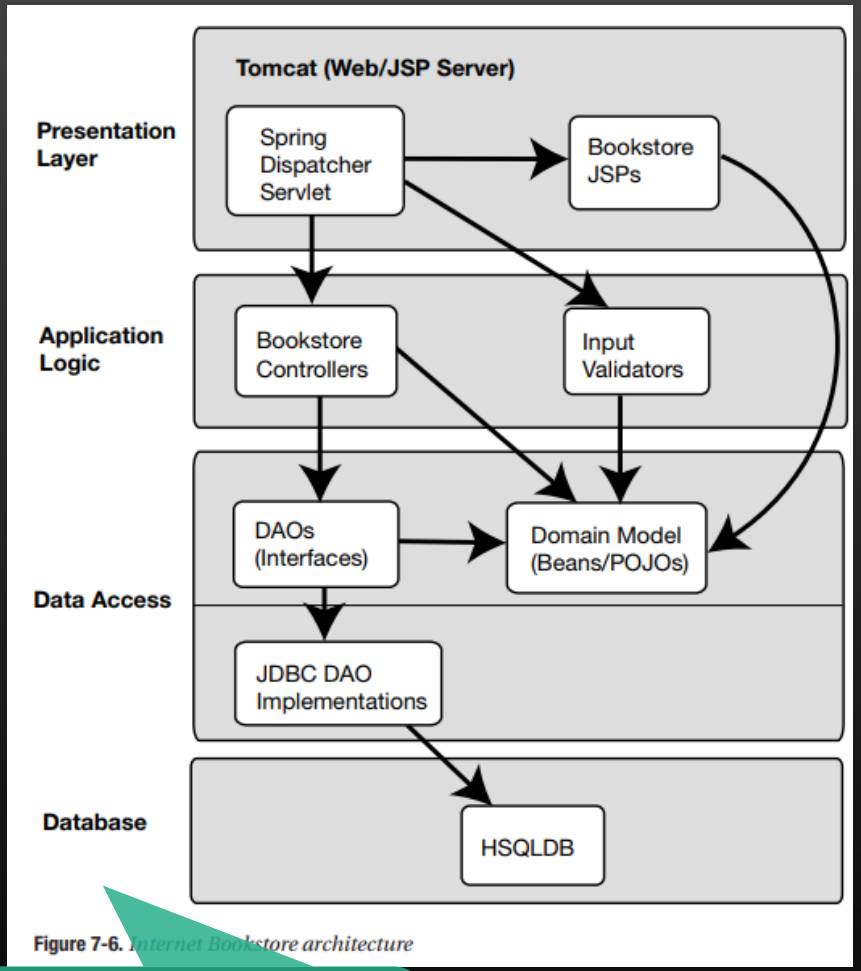


Figure 7-6. *Internet Bookstore architecture*

Opção I: vista puramente lógica, sem elementos de implementação. Destaca a modularização esperada do sistema.

Opção II: arquitetura tecnológica, mostra os componentes da aplicação tendo em conta os elementos da implementação. Explica a forma como as tecnologias/ambientes/bibliotecas serão usadas na construção da "full stack".

Referências

Core readings	Suggested readings
• [Larman04] – Chap. 13	• Fowler, “Software Architecture Guide”