UNIVERSIDADE EDUARDO MONDLANE

FACULDADE DE ENGENHARIA

DEPARTAMENTO DE ENGENHARIA ELECTROTÉCNICA

Programação Orientada a Objectos II

Desenvolvimento em Camadas

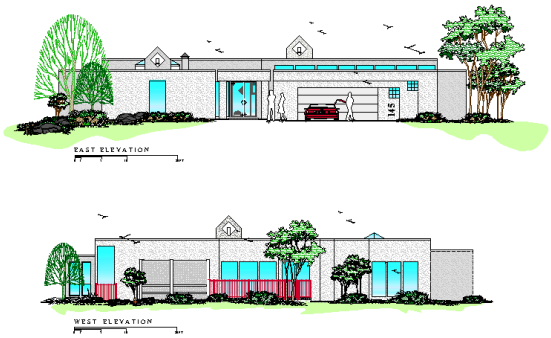
Docente: Ruben Manhiça
Leila Omar
Bhavika Rugnat

Maputo, 23 de janeiro de 2021

Conteúdo da Aula

1. Evolução da Arquitectura de Sistemas;
2. Desenvolvimento em Camadas (1, 2, 3);
3. Principios de Desenvolvimento em 3 Camadas;
4. MVC;

Arquitetura de um Sistema



Definição: Representa a estrutura estática e comportamental de um sistema, compreendendo os serviços do sistema, realizados pelos componentes de software, que correm nos processadores disponíveis

Está associada a detalhes internos do software na medida que esses detalhes manifestam-se externamente

Não pode ser representada por um simples diagrama: é multifacetada

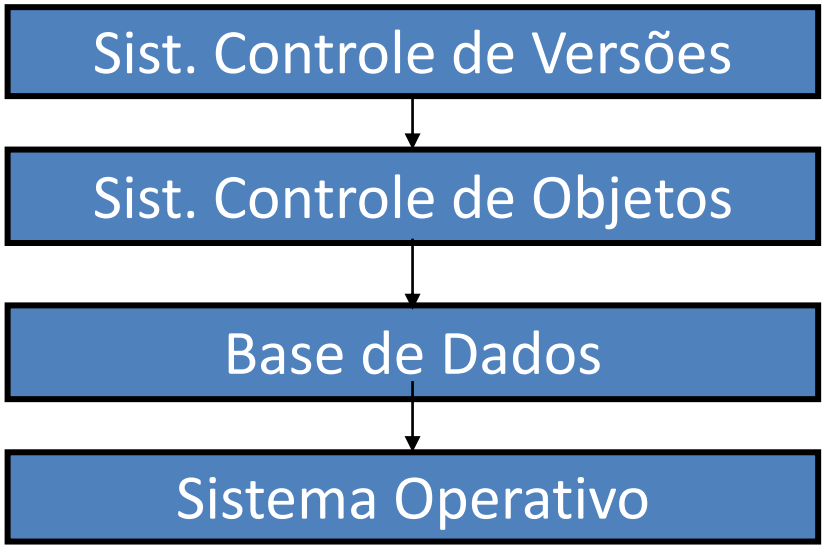
Arquitetura de um Sistema

- Inicialmente proposto por Dijkstra, em 1968
- Divisão do sistema hierarquicamente em camadas

Exemplos

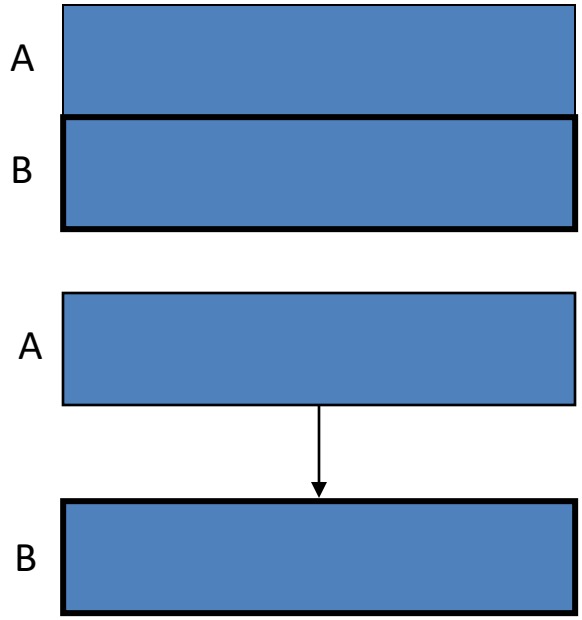
PROTOCOLO OSI

SGBD

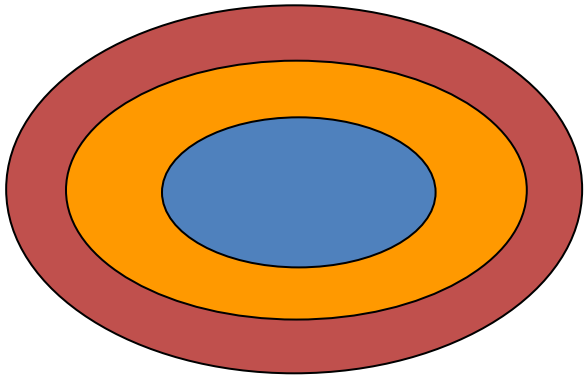


Notação

- Pilha



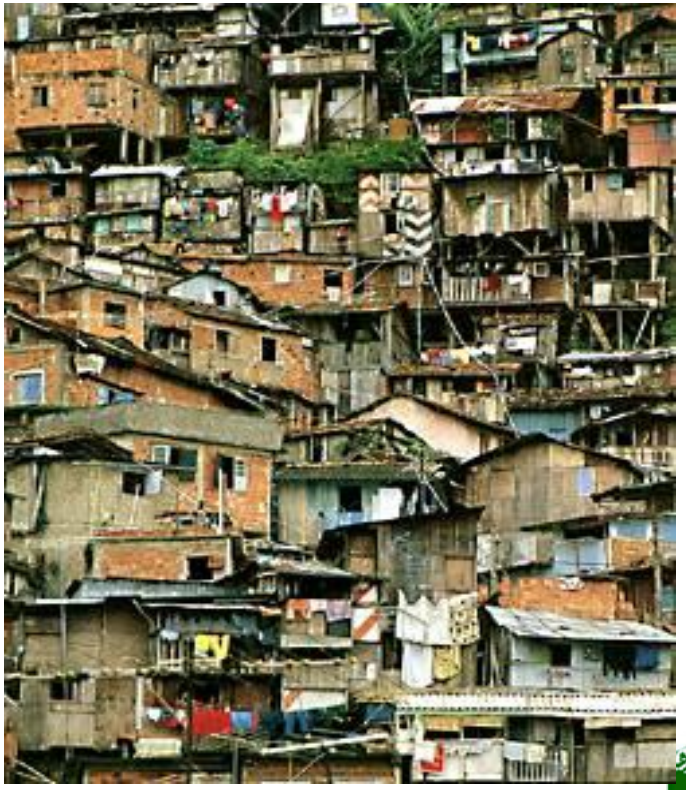
- Anel



- UML - Pacotes

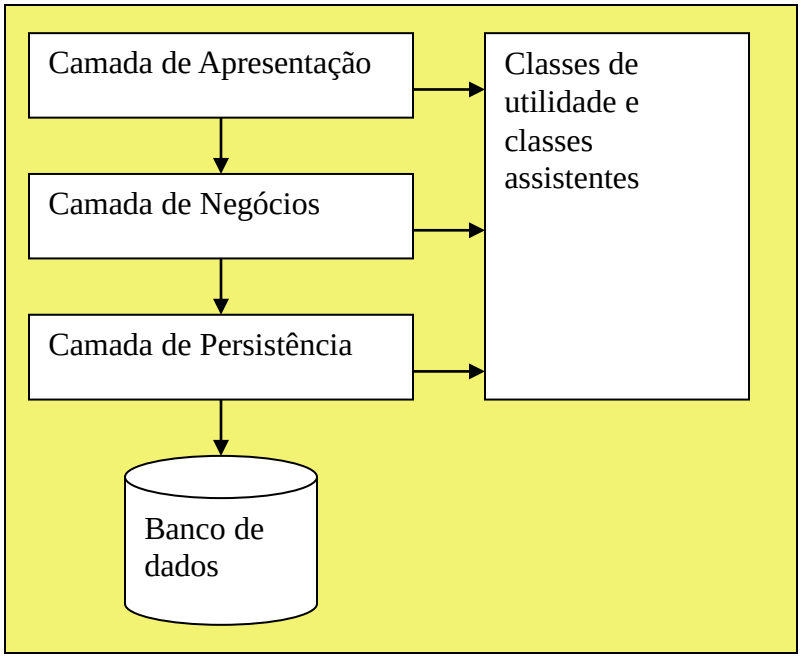
Todo sistema já criado tem sua Arquitetura

Ela existe independente do conhecimento pelo projetista de sistemas



Arquitetura em camadas

- Arquitetura em camadas visa a criação de aplicativos modulares, de forma que a camada mais alta se comunica com a camada mais baixa e assim por diante, fazendo com que uma camada seja dependente apenas da camada imediatamente abaixo.



Arquitetura em camadas

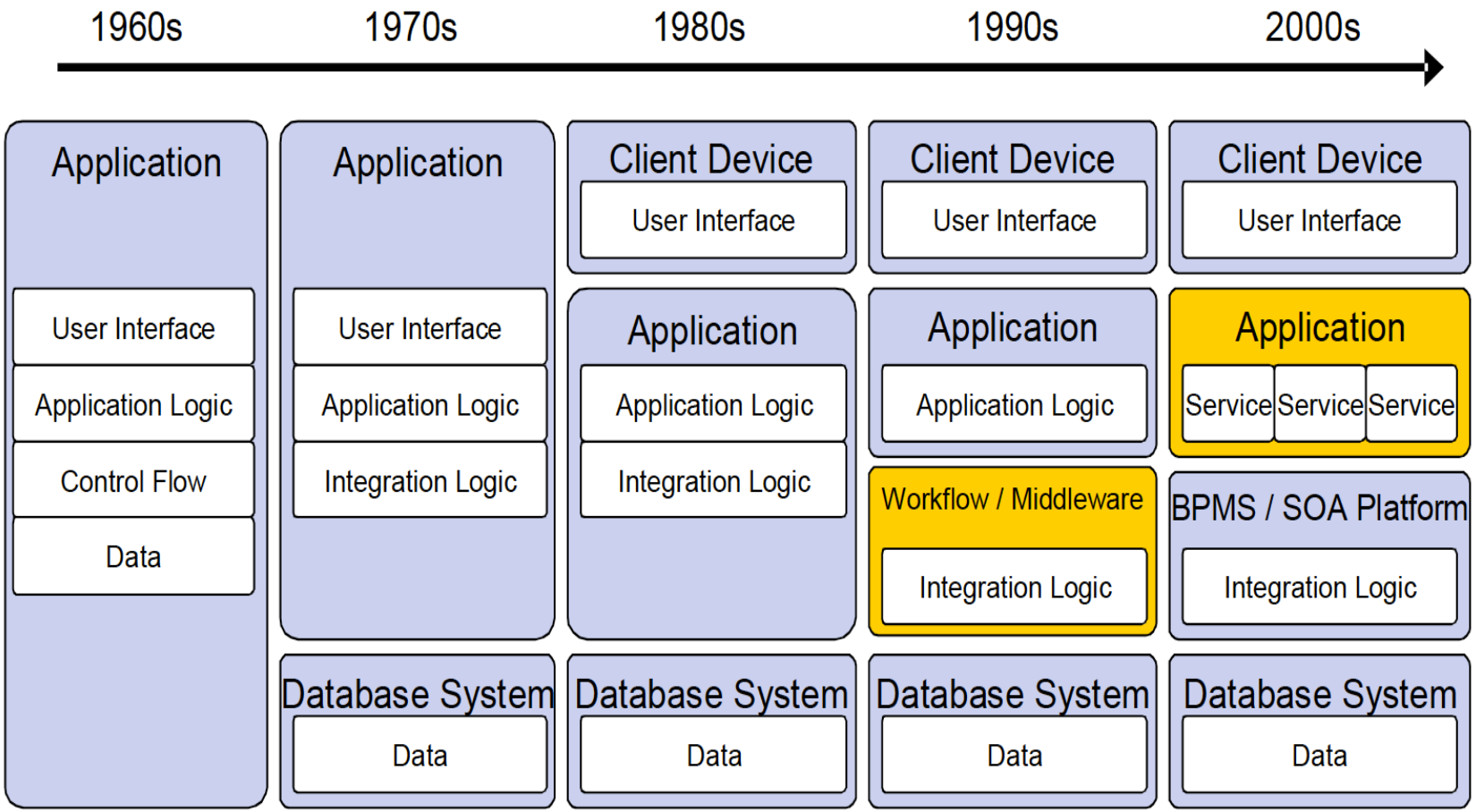
- **Camada de Apresentação:** Lógica de interface do utilizador (GUI). O código responsável pela apresentação e controle da página e tela de navegação forma a camada de apresentação;
- **Camada de Negócios:** Código referente a implementação de regras de negócio ou requisitos do sistema;
- **Camada de persistência:** Responsável por armazenamento e recuperação dos dados quando solicitados. Objetivo é o de garantir uma independência da fonte de dados (arquivos, bases de dados, etc) e ao mesmo tempo manter as informações entre diferentes sessões de uso.

Arquitetura em camadas

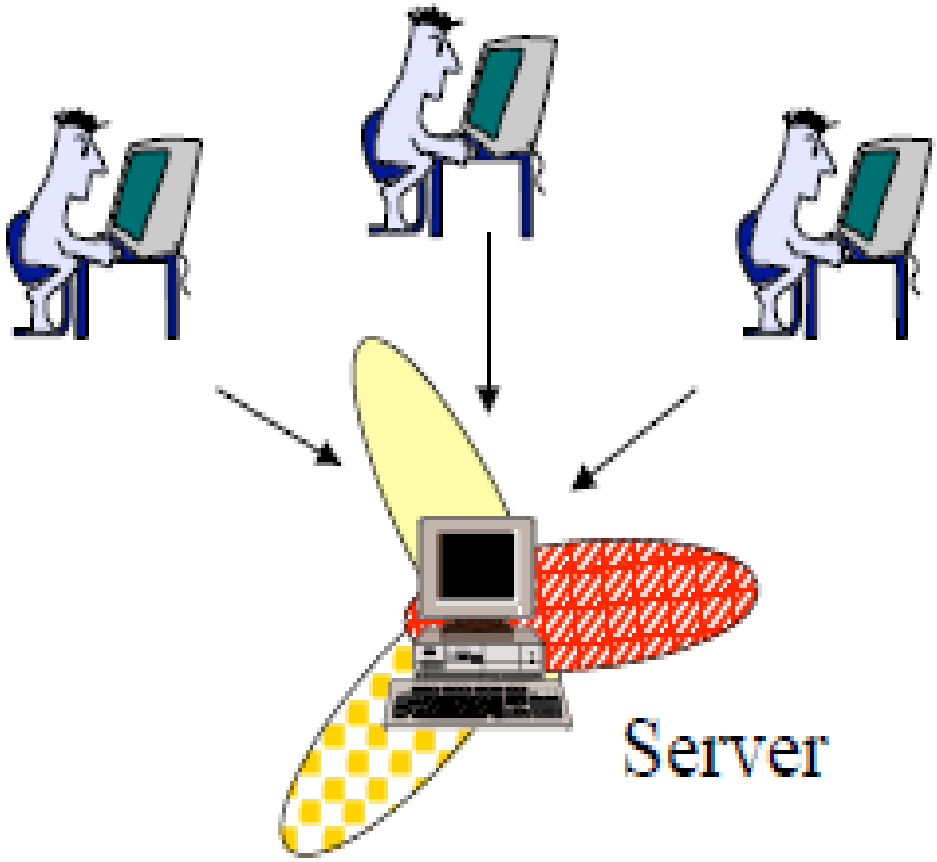
- **Base de dados**: A BD existe fora da aplicação Java, é a actual representação persistente do estado do sistema.

- **Assistentes e Classes de utilidade**: São classes necessárias para o funcionamento ou mesmo o complemento de uma aplicação ou parte dela, como por exemplo o Exception para tratamento de erros.

Evolução da Arquitectura de Sistemas



Aplicações Monoliticas (Stand-Alone Application)



Terminal burro

- Não possui capacidade de processamento
- Conexão a um servidor central
- Compartilhamento de recursos
- Altamente acoplada e centralizada
- Dependência de um único fornecedor de hardware e software

Aplicações Cliente-Servidor

- As aplicações na **Web** funcionam segundo o princípio de comunicação cliente-servidor.
  - O programa cliente exerce basicamente as funções de interface com o utilizador e geração dos comandos de consulta.
  - O programa servidor nunca toma a iniciativa.
- Na comunicação cliente-servidor, a iniciativa é do cliente.

Aplicações Cliente-Servidor

- As aplicações do tipo cliente-servidor são logicamente divididas em três camadas:
  - lógica de apresentação
  - lógica de negócio
  - lógica de acesso aos dados

Aplicações Cliente-Servidor

Lógica de Apresentação:

- Define como o usuário interage com a aplicação
- Usualmente é implementada através de uma interface gráfica com o usuário (GUI).
    - Exemplos: Windows ou Web.

Lógica de Negócio

- Define a mecânica (ou regras de negócio) da aplicação.
    - Exemplo: a transferência do dinheiro de uma conta para outra implica saque em uma conta e depósito em outra.
- A lógica do negócio pode ser executada no cliente ou no servidor.

Lógica de Acesso aos Dados

- Define como os dados são armazenados e recuperados, garantindo que a integridade dos dados seja mantida.
    - Essa função é executada pelo SGDB.

Arquitetura duas-camadas: cliente-gordo/servidor magro

LÓGICA DE APRESENTAÇÃO

LÓGICA DE NEGÓCIO

LÓGICA DE ACESSO AOS DADOS

Nome:	Paula
Sobrenome:	Santos
Data:	17-10-00
UF:	PR

Inserir

Reservar-quarto
- encontrar quarto livre
- se quarto livre, então cadastrar paciente fazer reserva. Senão alertar administração

Nº do paciente	Nome	Sobrenome	Data da Internação	Data da alta	UF
001-01-5491	Pedro	Farias	13-05-68	15-05-68	PR
012-21-1101	José	Pires	23-10-87	29-10-87	PR
201-66-7310	Maria	Antunes	16-12-99	20-12-99	SP
302-19-8329	Ana	Dias	29-01-00	01-02-00	SP
601-20-9962	Paula	Santos	17-10-00		PR

Nº do quarto	Nº do paciente
Q001	601-20-9962
Q002	
Q003	

cliente

servidor

Arquitetura duas camadas: Cliente Magro / Servidor Gordo

LÓGICA DE APRESENTAÇÃO

Nome: Paula
Sobrenome: Santos
Data: 17-10-00
UF: PR
Inserir

cliente

LÓGICA DE NEGÓCIO

Reservar-quarto
- encontrar quarto livre
- se quarto livre, então cadastrar paciente fazer reserva. Senão alertar administração

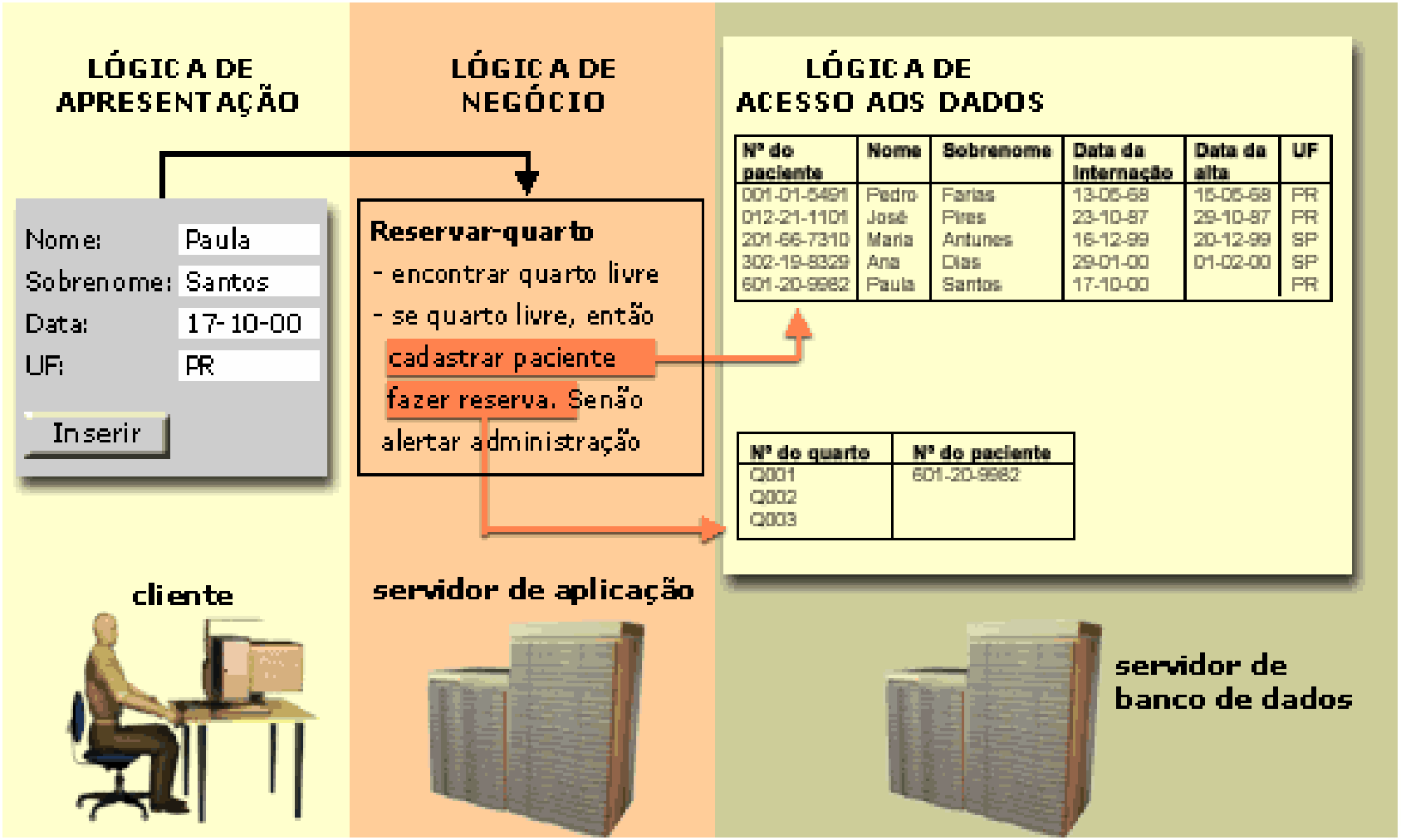
LÓGICA DE ACESSO AOS DADOS

Nº do paciente	Nome	Sobrenome	Data da Internação	Data da alta	UF
001-01-5491	Pedro	Farias	13-05-88	15-05-88	PR
012-21-1101	José	Pires	23-10-87	29-10-87	PR
201-66-7310	Maria	Antunes	16-12-99	20-12-99	SP
302-19-8329	Ana	Dias	29-01-00	01-02-00	SP
601-20-9982	Paula	Santos	17-10-00		PR

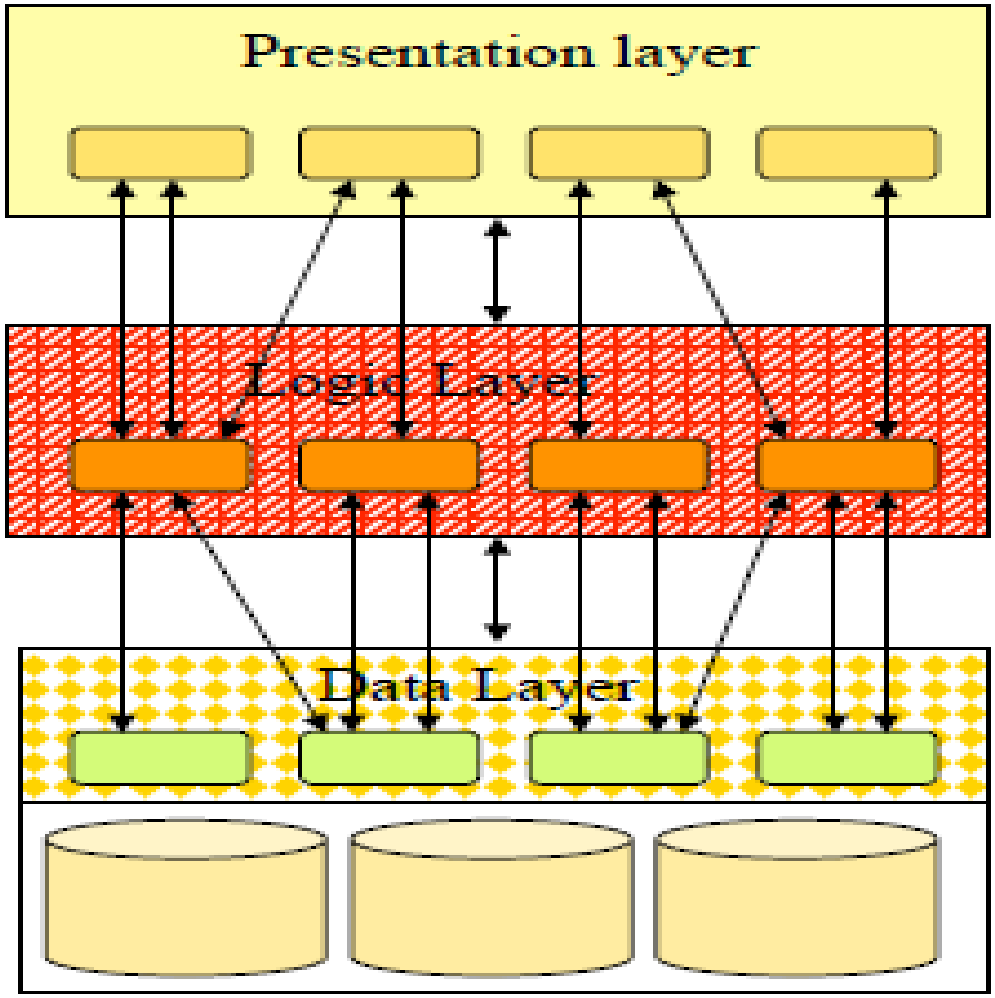
Nº do quarto	Nº do paciente
Q001	601-20-9982
Q002	
Q003	

servidor

Arquitetura três-camadas: cliente magro / servidor magro



Principios de Desenvolvimento em 3 Camadas

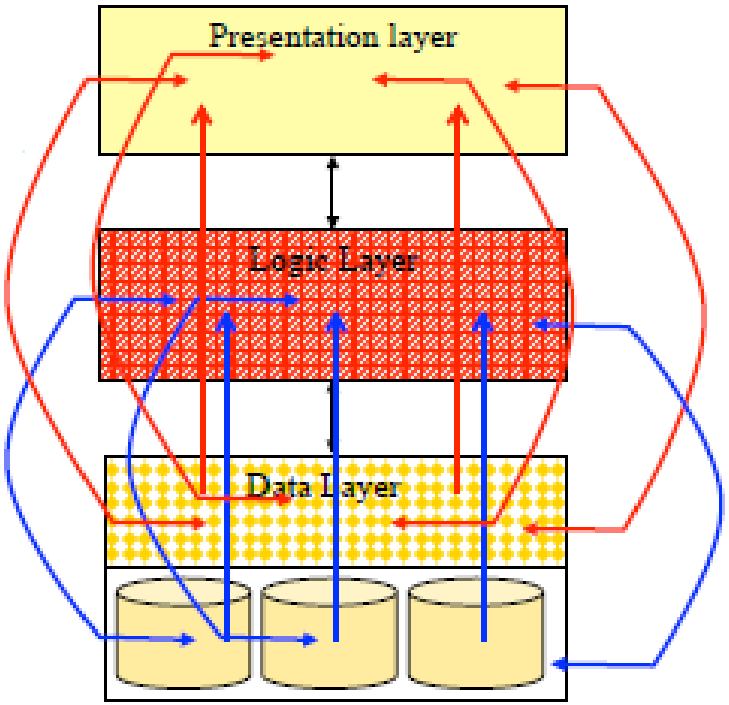


Principios de Desenvolvimento em 3 Camadas

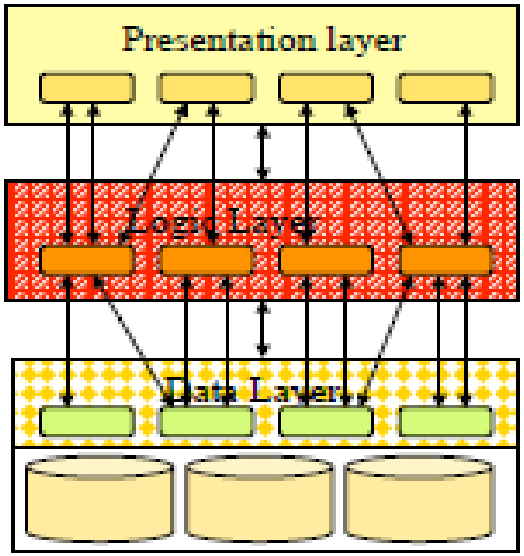
- Cada Camada deve criar suas proprias classes e Objectos relacionados;
- Toda Comunicação/mensagens so pode ser feita por camadas adjacentes;
- Cada camada usa seus proprios Objectos ou somente objectos de camadas adjacentes;

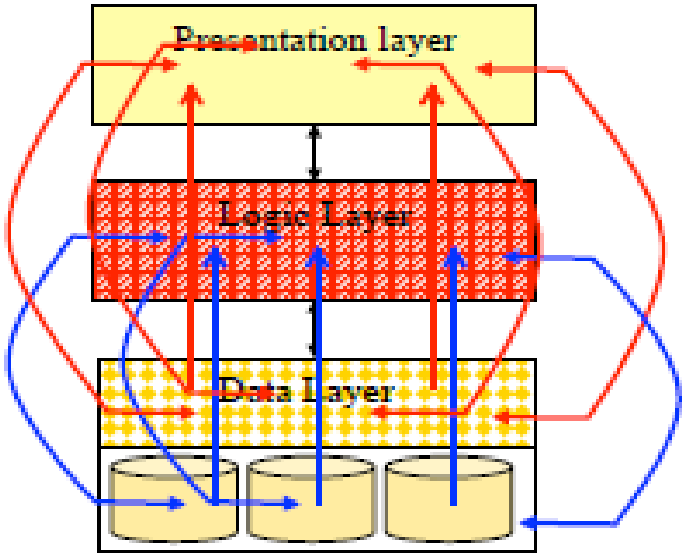
Principios de Desenvolvimento em 3 Camadas

- Em toda aplicação é sempre possível de desviar do desenvolvimento em 3 camadas, quando isso é feito o código é chamado de **Codigo Espargueta** !



Pros & Contras





- Reusabilidade;
- Escalabilidade;
- Legibilidade;
- **Dispendioso;**
- **Muitos Objectos, Chamdas, Linhas de codigo**

- Velocidade;
- Poucos objectos, chamadas e linhas de Codigo;
- **Não Escalavel;**
- **Não reusavel;**

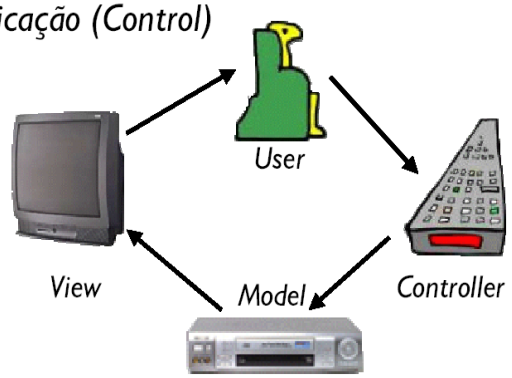
MVC (Model View Controller)

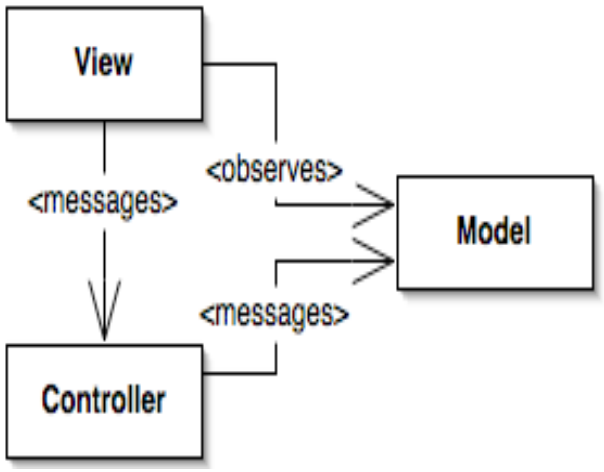
Assim como Bolachas o MVC tem 3 Camadas

Model-View-Controller (MVC) Design Pattern

- O padrão de arquitetura MVC (model-view-controller) surgiu nos anos 80 com a linguagem SmallTalk.
  - Criado por Trygve Reenskaug no fim dos anos 70
  - Usado no desenvolvimento de aplicações desktop por facilitar o desenvolvimento em camadas de aplicações que usam a orientação a objetos

Técnica para separar dados ou lógica de negócios (Model) da interface do usuário (View) e do fluxo da aplicação (Control)





Model-View-Controller (MVC) Design Pattern

- **O que é o MVC**
  - padrão projeto para o desenvolvimento de aplicações,
  - A implementação de aplicações usando este padrão são feitas com recurso a frameworks, apesar de não ser obrigatória a utilização de um para seguir o padrão.
- **Objetivo do MVC**
  - Isolar mudanças na GUI, evitando que estas mudanças acarretem em mudanças na Camada de Negocio da Aplicação (Application's Domain Logic)
- **Vantagens**
  - Facilita a manutenção
  - Facilita o desenvolvimento por times multi-disciplinares:
    - desenvolvedores – *creating robust business code*
    - designers – *building usable and engaging UIs*

Model-View-Controller (MVC) Design Pattern

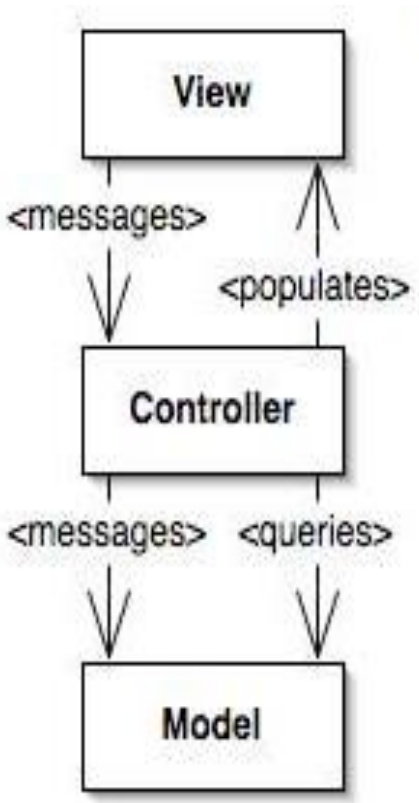
- Camadas e respectivas funções
  - **Model**:
    - Define as regras de acesso e manipulação dos dados Armazenados em bases de dados ou ficheiros, mas nada indica que sirva só para alojamento persistente dos dados.
    - Pode ser usada para dados em memória volátil, p.e.: memória RAM, apesar não se verificar tal utilização com muita frequência. Todas as regras relacionadas com tratamento, obtenção e validação dos dados devem ser implementados nesta camada.
  - **View:**
    - Responsável por criar a forma como a resposta será apresentada, página web, formulário, relatório, etc...
  - **Controller:**
    - Responsável por responder aos pedidos por parte do utilizador. Sempre que um utilizador faz um pedido ao servidor esta camada é a primeira a ser executada.

MVC Model 2

- Com o crescimento das aplicações web baseadas no protocolo HTTP que é sem estado, não temos mais uma sessão permanentemente aberta entre o cliente e o servidor. Além disso o HTTP não prevê uma forma de "enviar" (push) informações do servidor para o cliente.

- Isto impede o trabalho do Controller que não pode mais enviar informações para a View sem ser solicitado. Para contornar o problema a Sun criou o MVC Model 2, baseado no padrão FrontController.

- Agora a camada Controller submete ações tentando acompanhar o processo de request-response do protocolo HTTP ao invés de observar a camada Model, criando um fluxo linear para a arquitetura das aplicações.



Plataformas de Desenvolvimento em MVC em Java

- Mentawai
- VRaptor
- Neo Framework
- Brutos Framework
- Apache Struts
- Tapestry
- WebWork
- Play! Framework
- Spring MVC
- JSF
- Click Framework

TPC

- Aprofundar os conhecimentos sobre desenvolvimento em camadas

FIM!!!

Duvidas e Questões?