Laboratório de Engenharia de Software

Model-View-Controller (MVC)

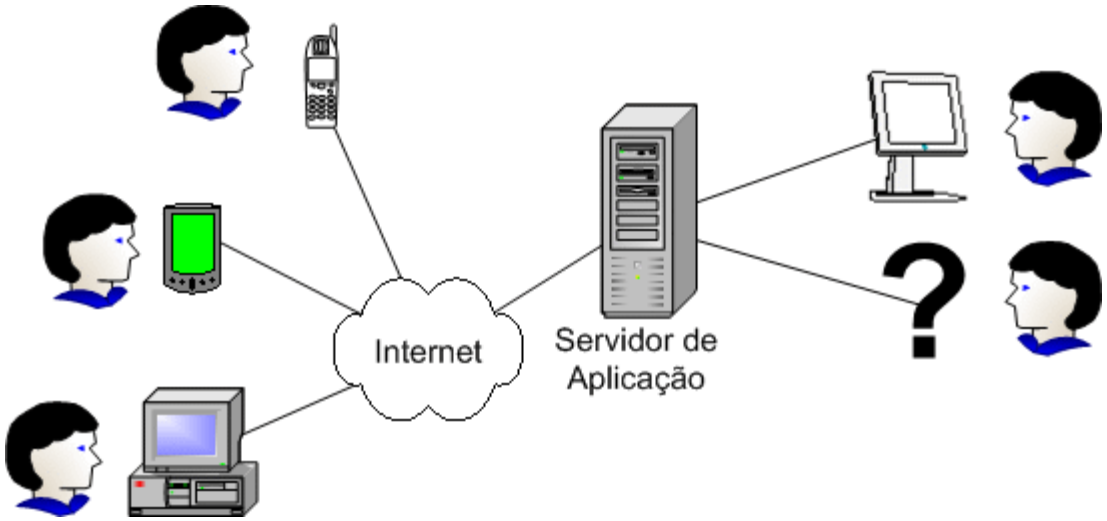
Fernando de Freitas Silva
fernd.ffs@gmail.com

Arquitetura MVC

Objetivo:

- Separar dados ou lógica de negócios (Model) da interface do usuário (View) e do fluxo da aplicação (Control)
- A idéia é permitir que uma mesma lógica de negócios possa ser acessada e visualizada através de várias interfaces.
- Na arquitetura MVC, a lógica de negócios (chamaremos de Modelo) não sabe de quantas nem quais interfaces com o usuário estão exibindo seu estado.
- Com as diversas possibilidades de interfaces que conhecemos hoje, a MVC é uma ferramenta indispensável para desenvolvermos sistemas (Figura 1).

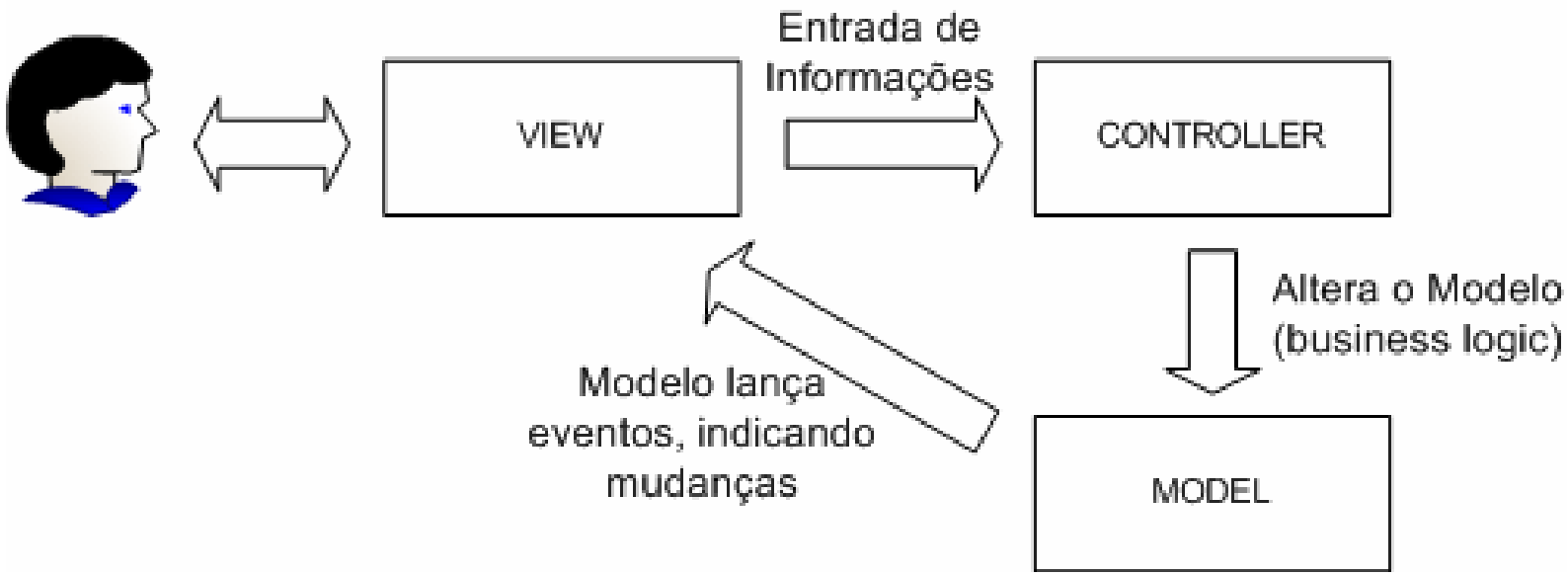
Arquitetura MVC



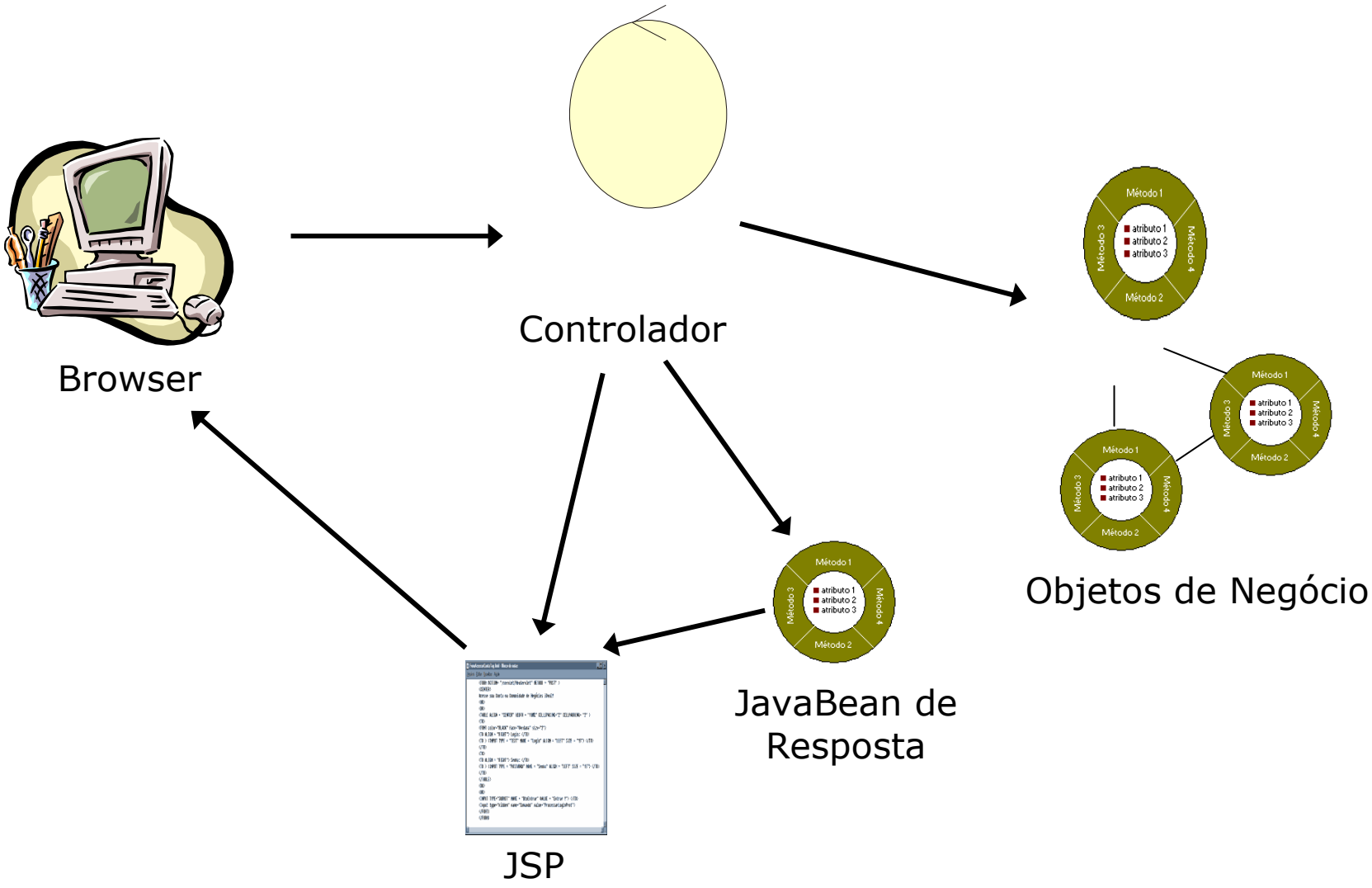
Arquitetura MVC

- Model – Representa o modelo da sua aplicação, com as regras de negócio (business logic) e todo o processamento da aplicação
- View – Representa a informação e recolhe os dados fornecidos pelo usuário
- Controller – Recebe as informações da entrada e as transmite para o modelo

Arquitetura MVC



MVC X Servlet

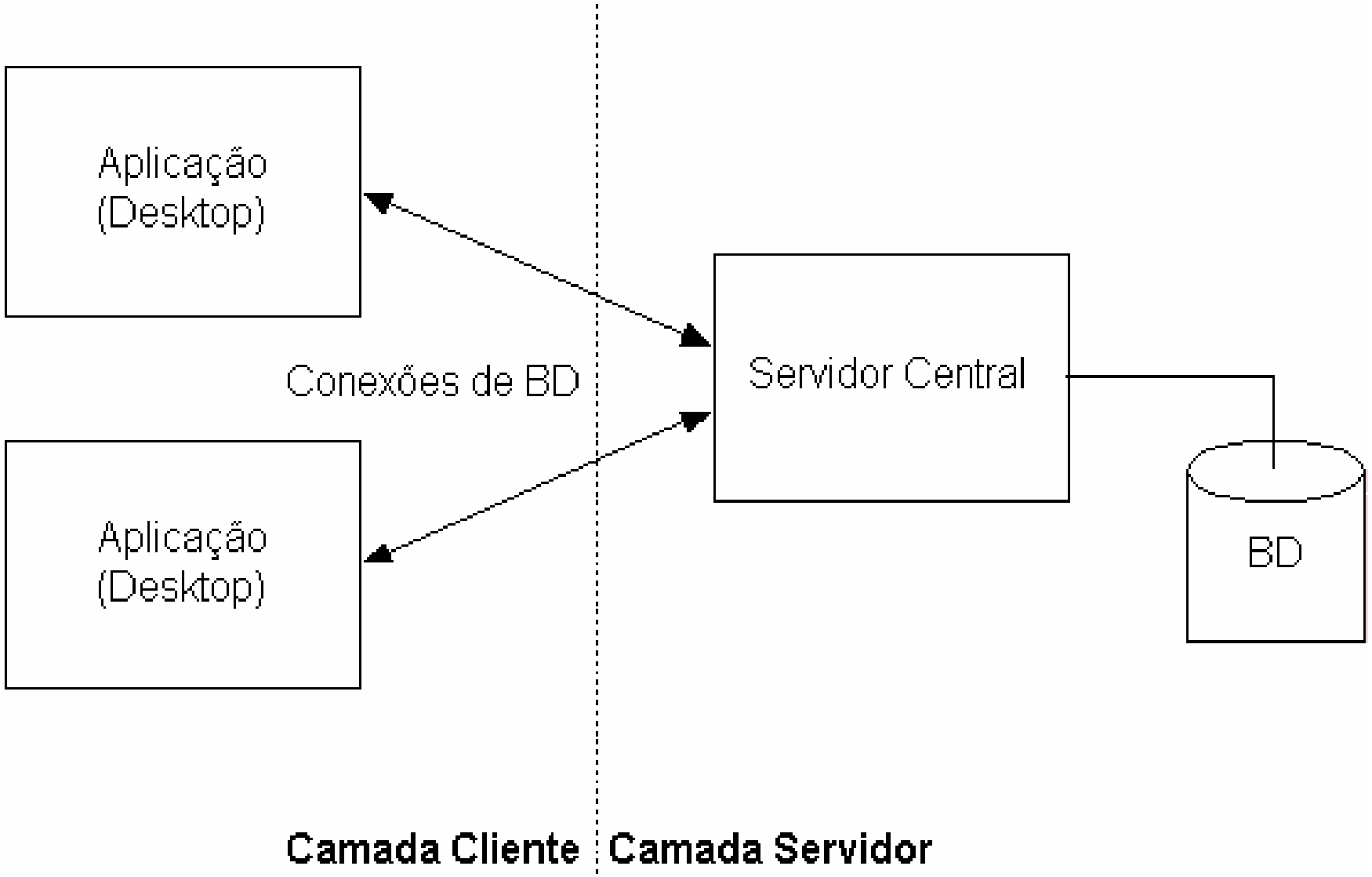


Arquitetura em Camadas

Arquitetura em Camadas

- **Arquitetura centralizada**
  - Dominantes até década de 80 como arquitetura corporativa
  - Problema básico: interface não amigável
- **Arquitetura em 2 camadas**
  - Sistemas em camadas surgiram para:
    - Melhor aproveitar os PCs da empresa
    - Oferecer sistemas com interfaces gráficas amigáveis
    - Integrar o desktop e os dados corporativos
  - Em outras palavras, permitiram aumentar a escalabilidade de uso de Sistemas de Informação
  - Os primeiros sistemas cliente-servidor eram de duas camadas
  - Camada cliente trata da lógica de negócio e da UI
  - Camada servidor trata dos dados (usando um SGBD)

Arquitetura em Camadas



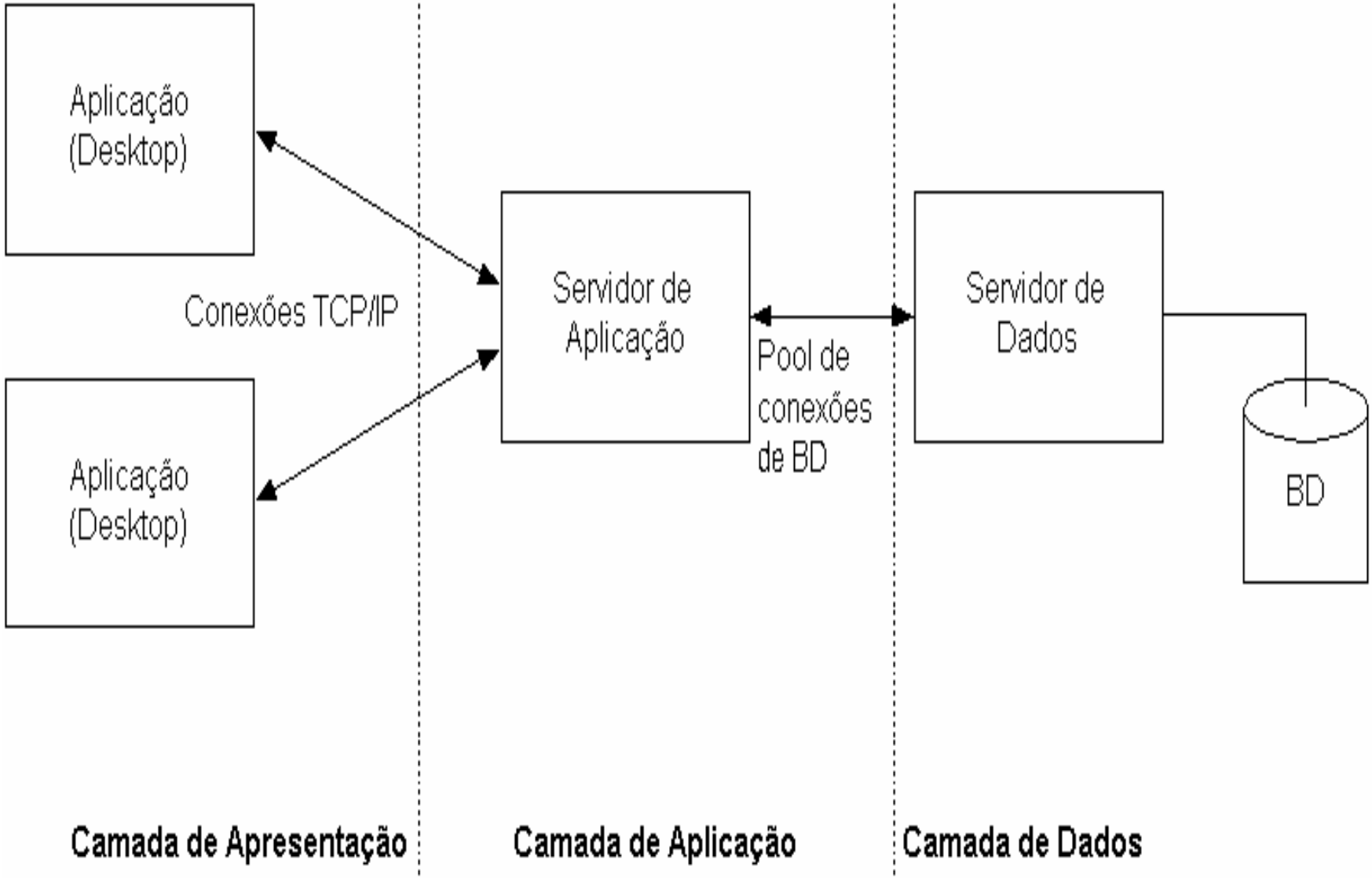
Arquitetura em Camadas

- **Arquitetura em 3 camadas**
  - A arquitetura cliente/servidor em 2 camadas sofria de vários problemas:
    - Falta de escalabilidade (conexões a bancos de dados)
    - Enormes problemas de manutenção (mudanças na lógica de aplicação forçava instalações)
    - Dificuldade de acessar fontes heterogêneas (legado CICS, 3270, ...)
  - Inventou-se a arquitetura em 3 camadas
    - Camada de apresentação (UI)
    - Camada de aplicação (business logic)
    - Camada de dados

Arquitetura em 3 Camadas

- **Apresentação:** Continua no programa instalado no cliente.
- **Lógica:** São as regras do negócio, as quais determinam de que maneira os dados serão utilizados.
- **Dados:** Nesta camada temos o servidor de Banco de dados, no qual reside toda a informação necessária para o funcionamento da aplicação.

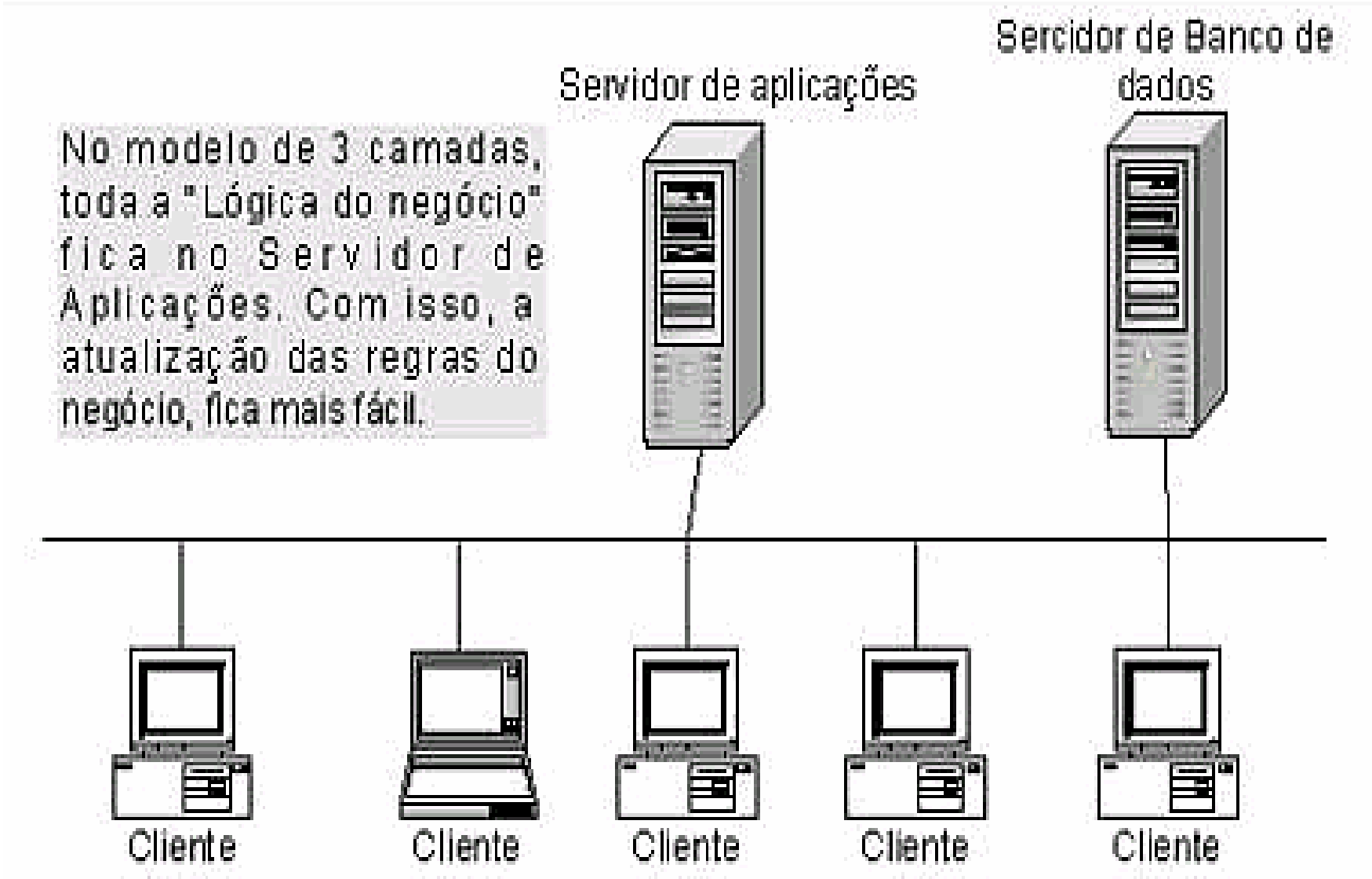
Arquitetura em Camadas



Arquitetura em Camadas

- Problemas de manutenção foram reduzidos, pois mudanças às camadas de aplicação e de dados não necessitam de novas instalações no desktop
- Observe que as camadas são lógicas
- Fisicamente, várias camadas podem executar na mesma máquina
- Quase sempre, há separação física de máquinas
- Em inglês, usa-se "layer" e "tier"
  - Para certas pessoas, não tem diferença
  - Para outras, "tier" indicaria camada física

Arquitetura em 3 camadas



Boas Práticas

O padrão Comando (I)

- Como implementar?
  - Atributo *hidden*

```
<form method="POST" action="/servlet/ServletCarrinho">
  <p>Carrinho de Brinquedo <br> R$ 10,00</p>
  <input type="hidden" name="Comando" value="Comprar">
  <p><input type="submit" value="Comprar" name="B1"></p>
</form>

<form method="POST" action="/servlet/ServletCarrinho">
  <input type="hidden" name="Comando" value="Fechar">
  <p><input type="submit" value="Fechar Compras" name="B1"></p>
</form>

<form method="POST" action="/servlet/ServletCarrinho">
  <input type="hidden" name="Comando" value="Logout">
  <p><input type="submit" value="Logout" name="B2"></p>
</form>
```

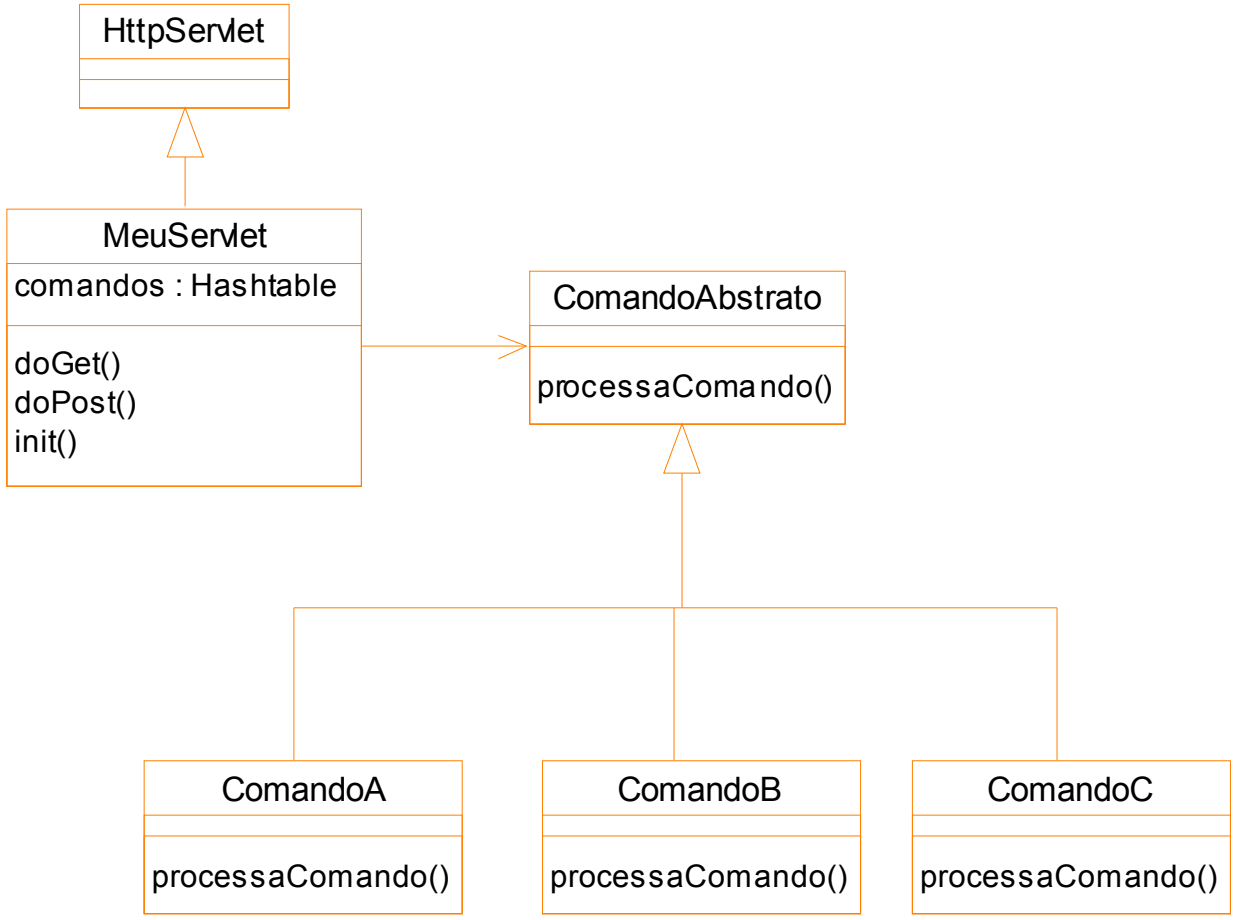
O padrão Comando (II)

- Como implementar?
  - Atributo *hidden* – Ex:

```
response.setContentType("text/html");
PrintWriter out = response.getWriter();
String comando = request.getParameter( "Comando" );
System.out.println( comando );
if ( comando.equalsIgnoreCase( "Login" ) )
{
   //...
}
if ( comando.equalsIgnoreCase( "Comprar" ) )
{
   //...
}
if ( comando.equalsIgnoreCase( "Fechar" ) )
{
   //...
}
if ( comando.equalsIgnoreCase( "Logout" ) )
{
   //....
}
```

  - Atributo hidden + Comando

O padrão Comando (III)



O padrão Comando (IV)

- O Comando Genérico:

```
public abstract class GenericCommand
{

    public abstract Vector execute( HttpServletRequest req ,
                                    HttpServletResponse res );

}
```

  - Passa o comando a ser instanciado (input do formulário)
- HttpServletResponse
  - Contém o PrintWriter que deverá receber as respostas do Comando

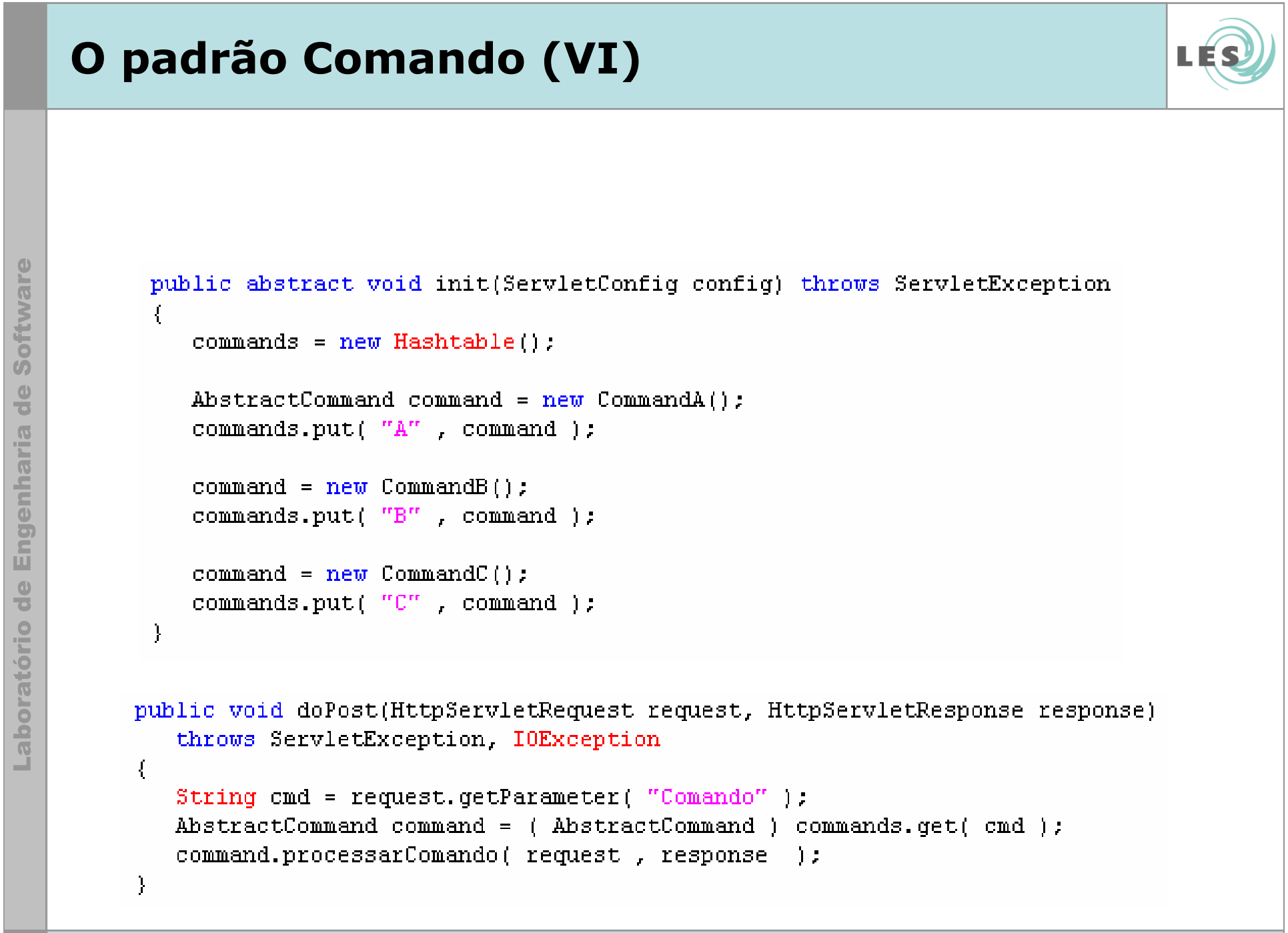
O padrão Comando (V)

- O Servlet deve possuir um atributo HashTable com todos os Commands

- Ao ser iniciado, cada Comando deve ser instanciado e inserido na HashTable

```
public class MeuServlet extends HTTPServlet
{
   private Hashtable commands;
```

O padrão Comando (VI)



```
public abstract void init(ServletConfig config) throws ServletException
{
    commands = new Hashtable();

    AbstractCommand command = new CommandA();
    commands.put( "A" , command );

    command = new CommandB();
    commands.put( "B" , command );

    command = new CommandC();
    commands.put( "C" , command );
}
```

```
public void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException
{
    String cmd = request.getParameter( "Comando" );
    AbstractCommand command = ( AbstractCommand ) commands.get( cmd );
    command.processarComando( request , response  );
}
```

O padrão Comando (VII)

FormAcessarContaTag.html - Bloco de notas

Arquivo Editar Localizar Ajuda

```
<FORM ACTION= "/servlet/MeuServlet" METHOD = "POST" >
<CENTER>
Acesse sua Conta na Comunidade de Negócios iDeal!
<HR>
<BR>
<TABLE ALIGN = "CENTER" WIDTH = "100%" CELLSPACING="2" CELLPADDING= "2" >
<TR>
<FONT color="BLACK" face="Verdana" size="2">
<TD ALIGN = "RIGHT"> Login: </TD>
<TD > <INPUT TYPE = "TEXT" NAME = "Login" ALIGN = "LEFT" SIZE = "15"> </TD>
</TR>
<TR>
<TD ALIGN = "RIGHT"> Senha: </TD>
<TD > <INPUT TYPE = "PASSWORD" NAME = "Senha" ALIGN = "LEFT" SIZE = "15"> </TD>
</TR>
</TABLE>
<BR>
<HR>
<INPUT TYPE="SUBMIT" NAME = "BtnEntrar" VALUE = "Entrar !"> </TD>
<Input type="hidden" name="Comando" value="ProcessarLoginProt">
</FONT>
</FORM>
```