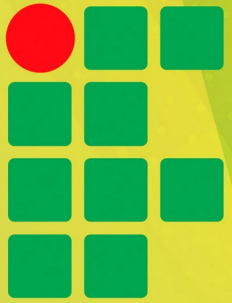


INSTITUTO FEDERAL

Sertão Pernambucano

Sistemas Distribuídos

Sistemas Distribuídos Baseados na Web.

Prof. Heraldo Gonçalves Lima Junior

INSTITUTO FEDERAL
Sertão Pernambucano

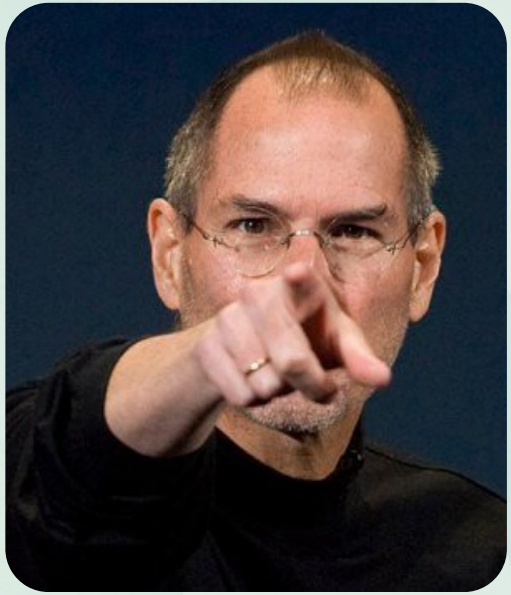
1. Paradigmas de Sistemas Distribuídos

- Troca de Mensagens (Messaging)
- Remote Procedure Call (RPC)
- Objetos Distribuídos
  - CORBA,DCOM e Java RMI
- Componentes Distribuídos
  - J2EE e .Net
- Web Services

2. Problemas

- Pressões do negócio
- Pressões tecnológicas

2.1. Problemas - Pressões do NEGÓCIO



- **Steve Jobs: "Internet time"**
- É um novo conceito (1997-1998) e significa que as empresas devem implementar novos sistemas muito rapidamente.

2.1. Problemas – Pressões do NEGÓCIO

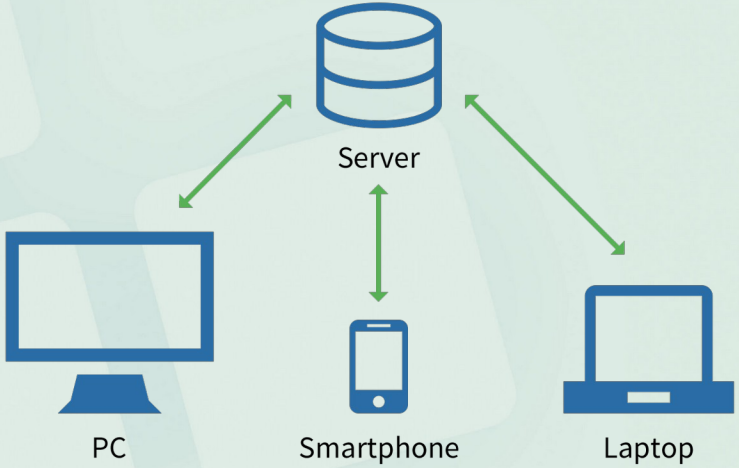
- Muitos sistemas de empresas devem **migrar para Internet**/Intranet/Extranet
- Novos tipos de sistemas devem ser desenvolvidos para usar a tecnologia como **business advantage**, dando um diferencial nos negócios

2.1. Problemas - Pressões do NEGÓCIO

- Muitos sistemas devem mudar devido a mudanças nos negócios
  - Fusões e aquisições
- **Resultado: tem muito mais software a fazer, muito mais rapidamente**
  - Mas há pressões para manter custos de IT (Information Technology) baixos

2.2. Problemas - Pressões TECNOLÓGICAS

- **Evolução tecnológica força mudanças de sistemas**
  - Muitas empresas acabam de fazer a **transição para cliente/servidor** e agora a palavra de ordem é "**client/server is dead!"**
- – Isto é, client/server em 2 camadas.



2.1. Problemas - Pressões do NEGÓCIO

- "O Cliente/Servidor foi um tremendo erro. Lamentamos tê-los vendido a vocês“
- Larry Ellison, CEO da Oracle em 1999.

2.2. Problemas - Pressões TECNOLÓGICAS

- **Sistemas devem migrar para arquiteturas distribuídas (N-tier)**
  - Para ter escala
  - Para ter acesso a Internet mas com acesso a sistemas legados
    - **Não podemos jogar o legado fora**
  - Para juntar sistemas operando em plataformas diferentes

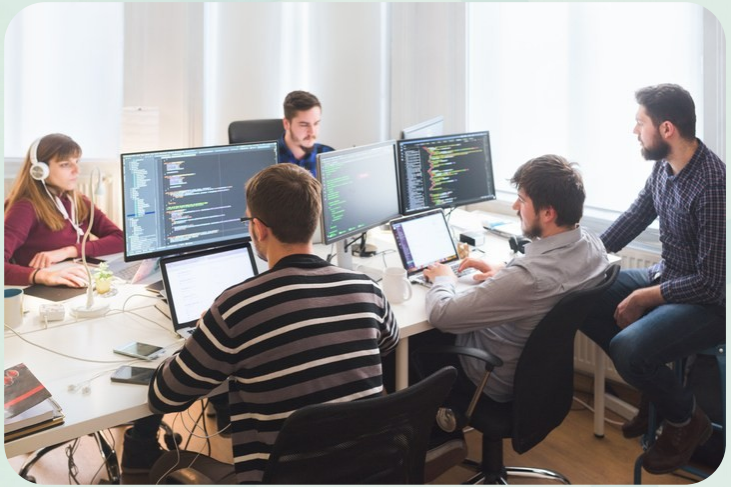
2.2. Problemas - Pressões TECNOLÓGICAS

- **O desenvolvimento de software ficou muito mais complexo nos últimos anos**
  - Pelos motivos acima
  - Porque usuários querem funcionalidade mais sofisticada
- **Problemas de complexidade**
  - O desenvolvimento de muitos grandes sistemas tem fracassado recentemente

2.2. Problemas - Pressões TECNOLÓGICAS

- **Resumindo: fazer sistemas de produção customizados do zero:**
  - É muito caro
  - Demora muito tempo
  - Não produz boa qualidade

2.3. Problemas – O que faremos?



- O que grandes empresas, com grandes problemas de desenvolvimento de sistemas, querem?

2.3. Problemas – O que faremos?

- **Na visão do usuário, software de qualidade:**
- Tem que **satisfazer** as necessidades do usuário
- Tem que ser feito no **prazo** combinado
- Tem que ser feito dentro do **custo** combinado
- Tem que ter nível aceitável de **defeitos**

2.3. Problemas – O que faremos?

- **Mas queremos ver a visão do desenvolvedor...**
- **Melhor flexibilidade**
  - Possibilitando satisfazer novos requisitos do negócio (=funcionalidade) rapidamente.

2.3. Problemas – O que faremos?

- **Melhor adaptabilidade**
  - Possibilitando customizar uma aplicação para vários usuários, usando várias alternativas de delivery dos serviços da aplicação com impacto mínimo ao núcleo da aplicação.

2.3. Problemas – O que faremos?

- **Melhor manutenibilidade**
  - Possibilitando atualizar uma aplicação, mas minimizando o impacto da maioria das mudanças;
  - Melhor reusabilidade;
  - Possibilitando rapidamente montar aplicações únicas e dinâmicas.

2.3. Problemas – O que faremos?

- **Melhor aproveitamento do legado**
  - Possibilitando reusar a funcionalidade de sistemas legados em novas aplicações
- **Melhor interoperabilidade**
  - Possibilitando integrar 2 aplicações executando em plataformas diferentes

2.3. Problemas – O que faremos?

- **Melhor escalabilidade**
  - Possibilitando distribuir e configurar a execução da aplicação para satisfazer vários volumes de transação
- **Menor tempo de desenvolvimento**
  - Possibilitando viver em "Internet time" e com baixo orçamento

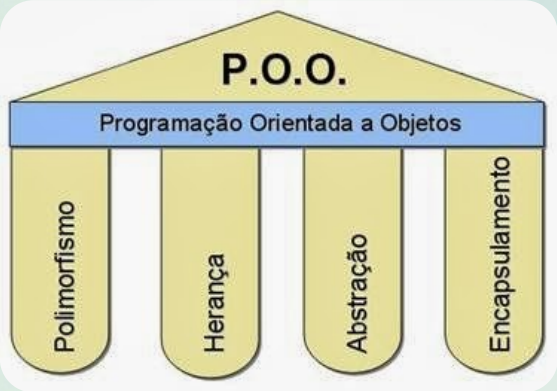
2.3. Problemas – O que faremos?

- **Melhor robustez**
  - Possibilitando ter soluções com menos defeitos
- **Menor risco**
  - Possibilitando tudo que falamos acima e ainda não se arriscar a ter projetos fracassados

2.4. Problemas – Orientação a Objetos não está resolvendo?

- Muitos grandes projetos baseados em OO estão fracassando recentemente.
- Aparentemente, OO **não está à altura**.
- **OO é apenas uma Enabling Technology**
  - Benefícios não são automáticos mas dependem do uso correto da tecnologia.

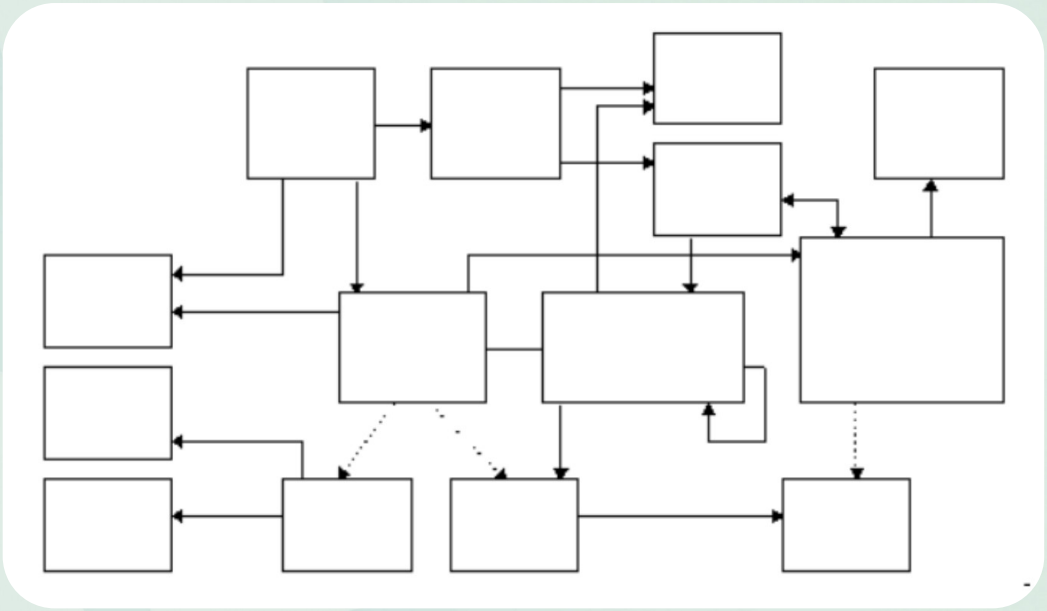
2.4. Problemas – Orientação a Objetos não está resolvendo?



- **Onde OO peca?**
  - OO (sozinha) não produz software reutilizável automaticamente
  - OO (sozinha) não tem boa escalabilidade
  - OO (sozinha) não provê boa encapsulação (esconder informação)
- **Isto pode ser visto com o resultado do uso de OO sem disciplina**

2.4. Problemas – Orientação a Objetos não está resolvendo?

- **Hyperspaghetti Objects**



2.4. Problemas – Orientação a Objetos não está resolvendo?

- **Ocorre mais quando software cresce em tamanho**
  - Um objeto pode referenciar qualquer outro
  - Referências demais não permitem arrancar um objeto e reutilizá-lo sozinho
  - Manutenção difícil, causando instabilidade
  - Tirar bugs introduz novos bugs
- **Resultado: o programa está muito complexo e não gerenciável**

2.4. Problemas – Orientação a Objetos não está resolvendo?

- **Onde OO peca mais?**
  - **OO (sozinha) não permite a construção de sistemas verdadeiramente extensíveis**
  - Para ser verdadeiramente extensível, um sistema deve permitir a adição tardia de uma extensão sem necessitar de uma verificação global de integridade
  - Em outras palavras, para adicionar algo, somos obrigados a passar por compilação e/ou link edição, testes de integração, etc.

2.4. Problemas – Orientação a Objetos não está resolvendo?

- **Queremos fazer mais ou menos como num sistema operacional**
- Aplicações estendem o SO mas não precisamos de um teste total de sistema
- Temos problemas de instalação e configuração, apenas

2.4. Problemas – Orientação a Objetos não está resolvendo?

- **O uso de herança em OO cria um acoplamento muito forte entre os pedaços**
  - Por este motivo, a extensão de uma classe usando herança (de implementação) requer frequentemente que se tenha o código fonte da classe sendo estendida
  - Muitos fornecedores de bibliotecas OO fornecem código fonte por este motivo

2.5. Soluções

- **Precisamos de novas abordagens:**
  - **Aplicações Multicamadas Distribuídas**
  - **Desenvolvimento Baseado em Componentes**

2.6. Soluções – Resumo Histórico

- **Arquitetura centralizada**
  - – Dominantes até década de 80 como arquitetura corporativa
  - – Problema básico: interface não amigável

2.6. Soluções – Resumo Histórico

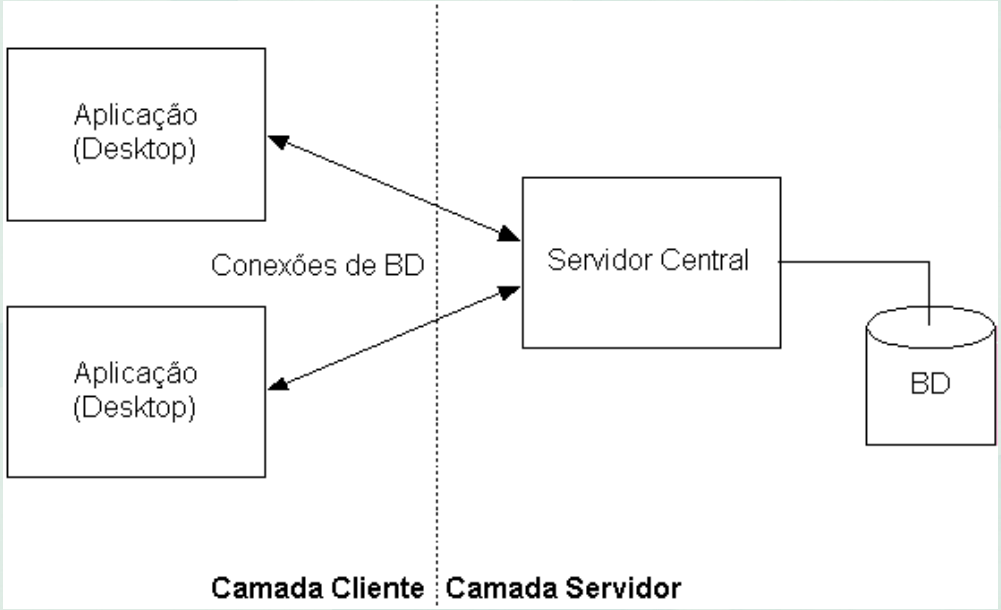
- **Arquitetura em 2 camadas**
  - **Sistemas em camadas surgiram para:**
    - Melhor aproveitar os PCs da empresa
    - Oferecer sistemas com interfaces gráficas amigáveis
    - Integrar o desktop e os dados corporativos
  - **Em outras palavras, permitiram aumentar a escalabilidade de uso de Sistemas de Informação**

2.6. Soluções – Resumo Histórico

- **Arquitetura em 2 camadas**
  - **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)

2.6. Soluções – Resumo Histórico

- **Arquitetura em 2 camadas**



2.6. Soluções – Resumo Histórico

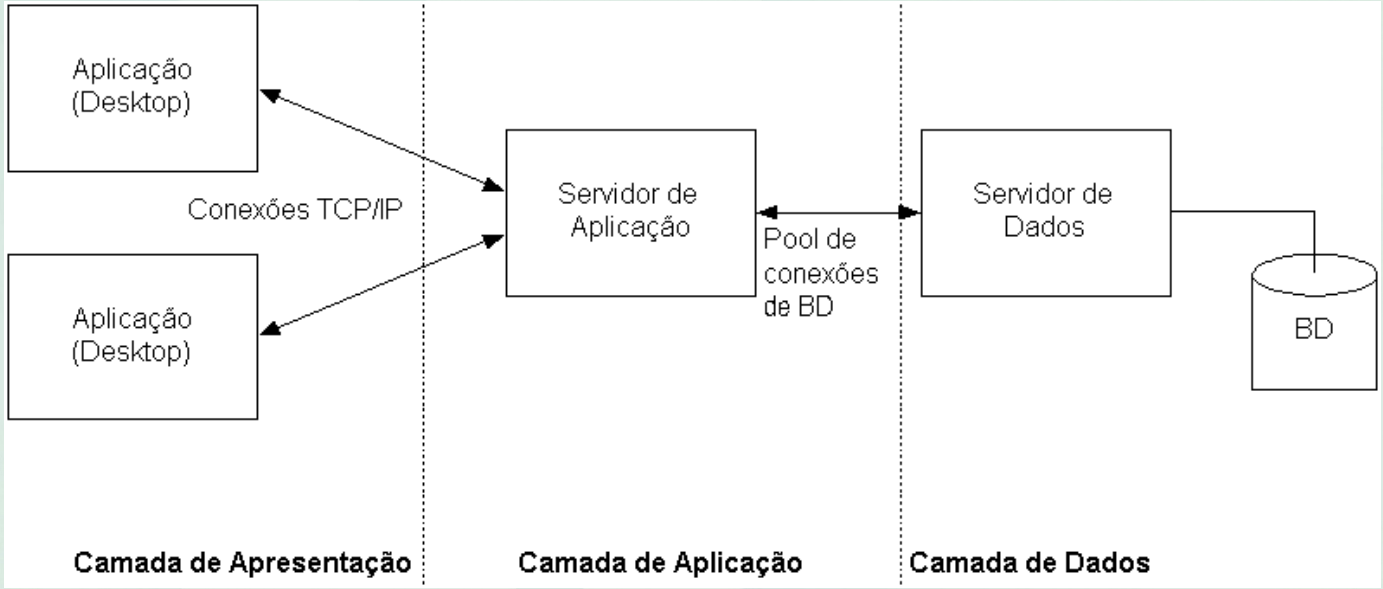
- **Arquitetura em 3 camadas**
  - **A arquitetura cliente/servidor em 2 camadas sofria de vários problemas:**
    - Falta de escalabilidade (conexões a bancos de dados)
    - Enormes problemas de manutenção (mudanças na lógica de aplicação forçava instalações)
    - Dificuldade de acessar fontes heterogêneas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

2.6. Soluções – Resumo Histórico

- **Arquitetura em 3 camadas**
  - **Inventou-se a arquitetura em 3 camadas**
    - Camada de apresentação (UI)
    - Camada de aplicação (business logic)
    - Camada de dados

2.6. Soluções – Resumo Histórico

- **Arquitetura em 3 camadas**



2.6. Soluções – Resumo Histórico

- **Arquitetura em 3 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

2.6. Soluções – Resumo Histórico

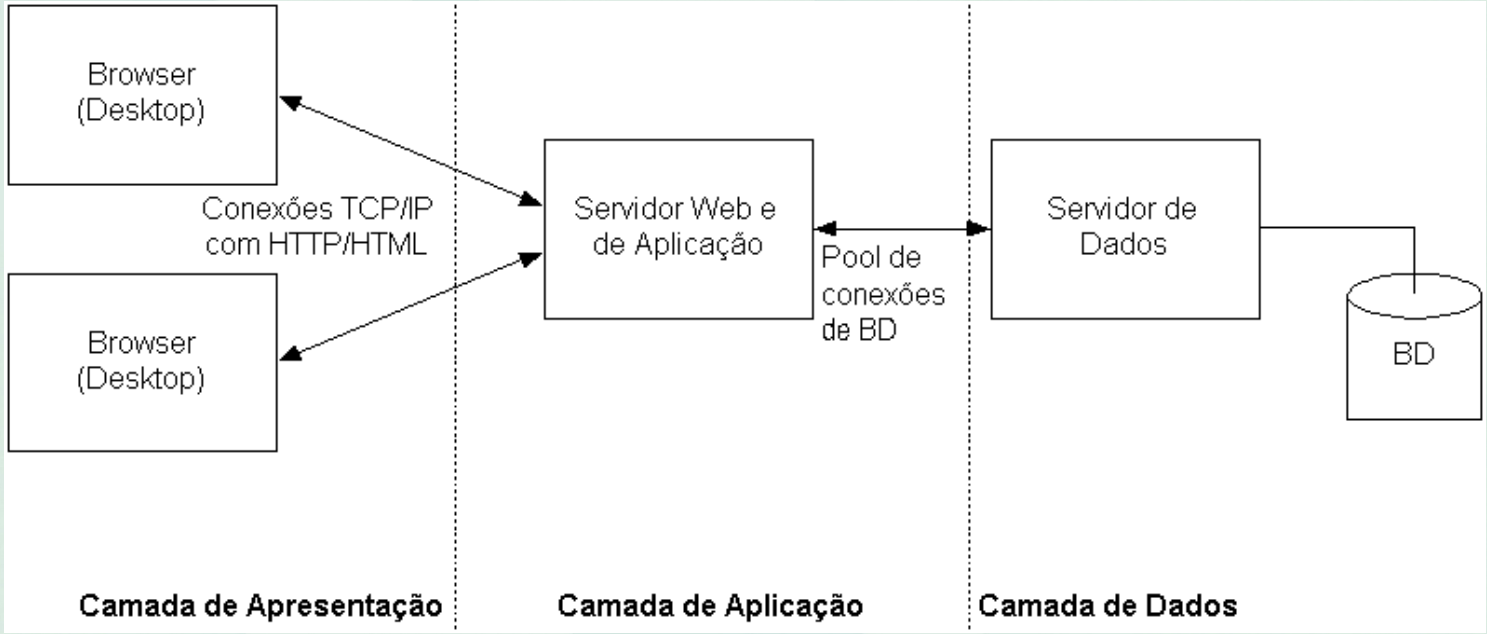
- **Arquitetura em 3/4 camadas Web-Based**
  - **A arquitetura em 3 camadas original sofre de problemas:**
    - A instalação inicial dos programas no desktop é cara
    - O problema de manutenção ainda persiste quando há mudanças à camada de apresentação
    - Não se pode instalar software facilmente num desktop que não está sob seu controle administrativo
      - Em máquinas de parceiros, em máquinas de fornecedores, em máquinas de grandes clientes, em máquinas na Internet

2.6. Soluções – Resumo Histórico

- **Arquitetura em 3/4 camadas Web-Based**
  - **Então, usamos o Browser como Cliente Universal**
    - Conceito de Intranet
    - A camada de aplicação se quebra em duas: Web e Aplicação
    - Evitamos instalar qualquer software no desktop e portanto, problemas de manutenção
    - Evitar instalação em computadores de clientes, parceiros, fornecedores, etc.

2.6. Soluções – Resumo Histórico

- **Arquitetura em 3/4 camadas Web-Based**



2.6. Soluções – Resumo Histórico

- **Arquitetura em 3/4 camadas Web-Based**
  - Às vezes, continua-se a chamar isso de 3 camadas porque as camadas Web e Aplicação frequentemente rodam na mesma máquina (para pequenos volumes)

2.6. Soluções – Resumo Histórico

- **Arquitetura distribuída em n camadas**
  - **Os problemas remanescentes:**
    - Não há suporte a Thin Clients (PDA, celulares, smart cards, quiosques, ...) pois preciso usar um browser (pesado) no cliente
    - Dificuldade de criar software reutilizável: cadê a componentização?
    - Fazer aplicações distribuídas multicamadas é difícil. Tem que:
      - Implementar persistência (impedance mismatch entre o mundo OO e o mundo dos BDs relacionais)
      - Implementar tolerância a falhas com fail-over

2.6. Soluções – Resumo Histórico

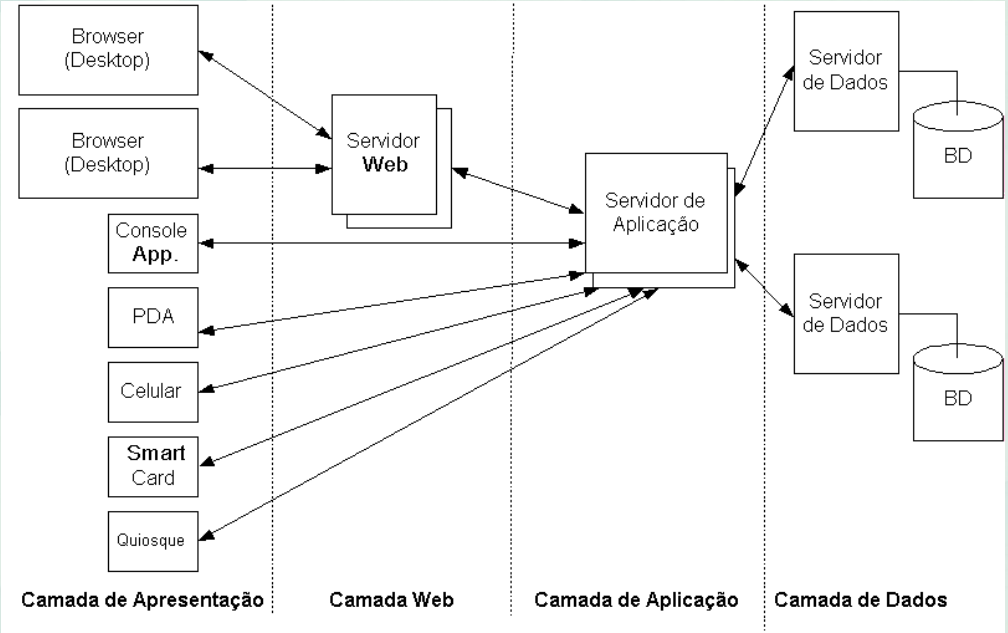
- Implementar gerência de transações distribuídas
- Implementar balanceamento de carga
- Implementar resource pooling
- etc.

2.6. Soluções – Resumo Histórico

- As empresas não querem contratar PhDs para implementar sistemas de informação!
- O truque é introduzir middleware num servidor de aplicação que ofereça esses serviços automaticamente
- Além do mais, as soluções oferecidas (J2EE, .Net) são baseadas em componentes

2.6. Soluções – Resumo Histórico

- **Arquitetura distribuída em n camadas**



2.6. Soluções – Resumo Histórico

- **Arquitetura distribuída em n camadas**
  - **As camadas podem ter vários nomes:**
    - Apresentação, interface, cliente
    - Web
    - Aplicação, Business
    - Dados, Enterprise Information System (EIS)
    - intro-1 programa

Obrigado!
Vlw! Flw!

INSTITUTO FEDERAL
Sertão Pernambucano