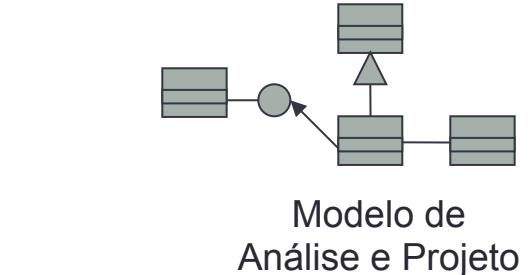
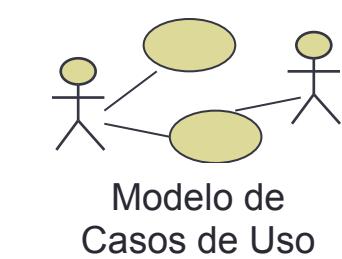


ARQUITETURA DE SISTEMAS

Roteiro

- Definição
- Documento de arquitetura
- Modelos de representação da arquitetura
- Estilos arquiteturais
- Arquitetura de sistemas web
- Arquitetura de sistemas mobile

Arquitetura e Projeto

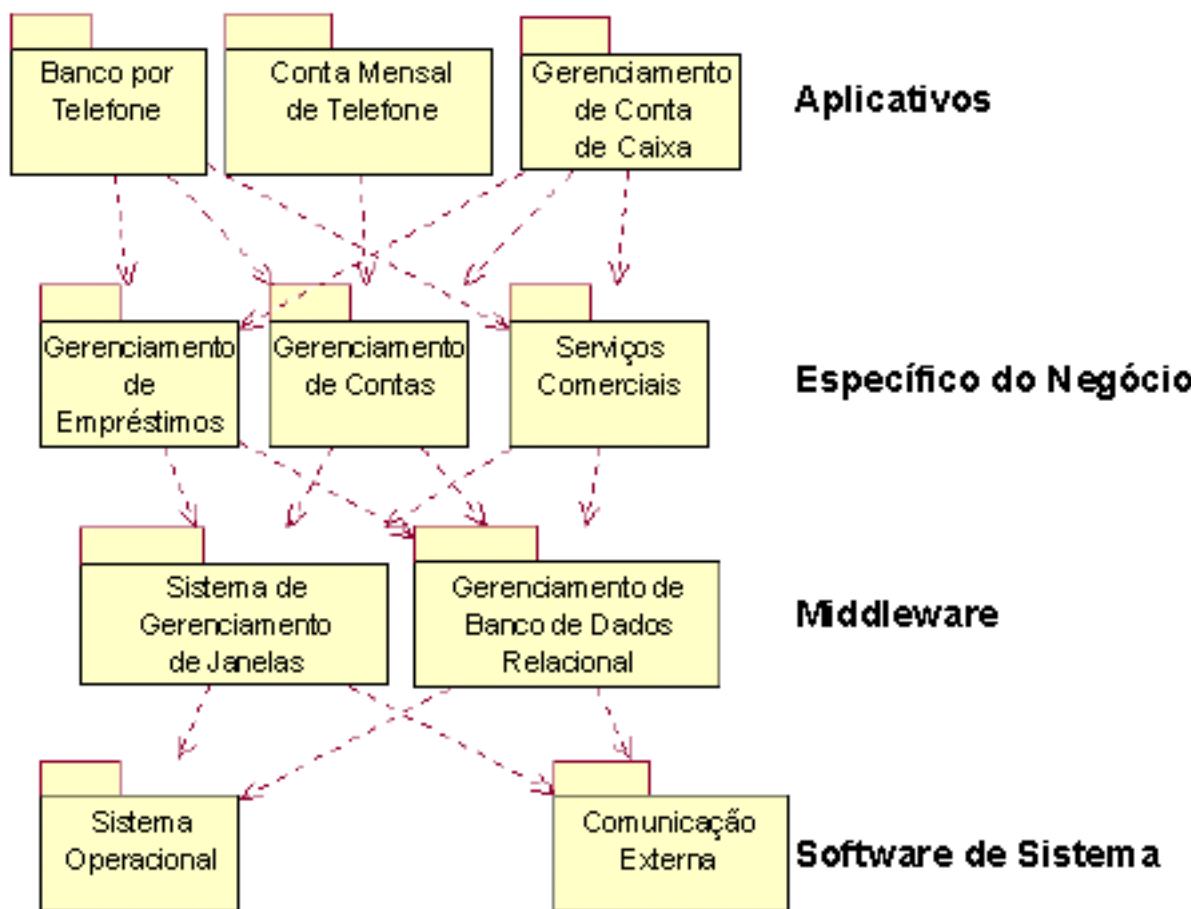


Arquitetura de software

- Sistemas grandes são decompostos em subsistemas
- Subsistemas
 - Fornecem um conjunto de serviços relacionados
 - Ex:
 - Autenticação
 - Comunicação
 - Suporte ao acesso a dados
 - Biblioteca de utilitários ([como usar no projeto da disciplina?](#))
 - Subsistemas diretamente relacionados ao negócio (Gestão Financeira)
 - Log de erros ([como usar no projeto da disciplina?](#))

Arquitetura de software

- Exemplo



O que é arquitetura de software?

“É a estrutura que engloba os subsistemas e componentes do software, definindo as suas propriedades visíveis externamente, o relacionamento entre eles e os mecanismos de controle” (Pressman, 2014)

Arquitetura de software

Além da estrutura:

- Decisões dos engenheiros para atender aos requisitos
(Martin Fowler, 2002; Pressman, 2014)
 - Exemplo da arquitetura distribuída com WCF
- Protocolos utilizados
 - Qual protocolo de comunicação Servidor-App será usado no projeto?
- Funcionalidades dos componentes
- Distribuição física dos componentes
- Capacidade de evolução
 - Como vocês vão incorporar interface web?

Arquitetura de software

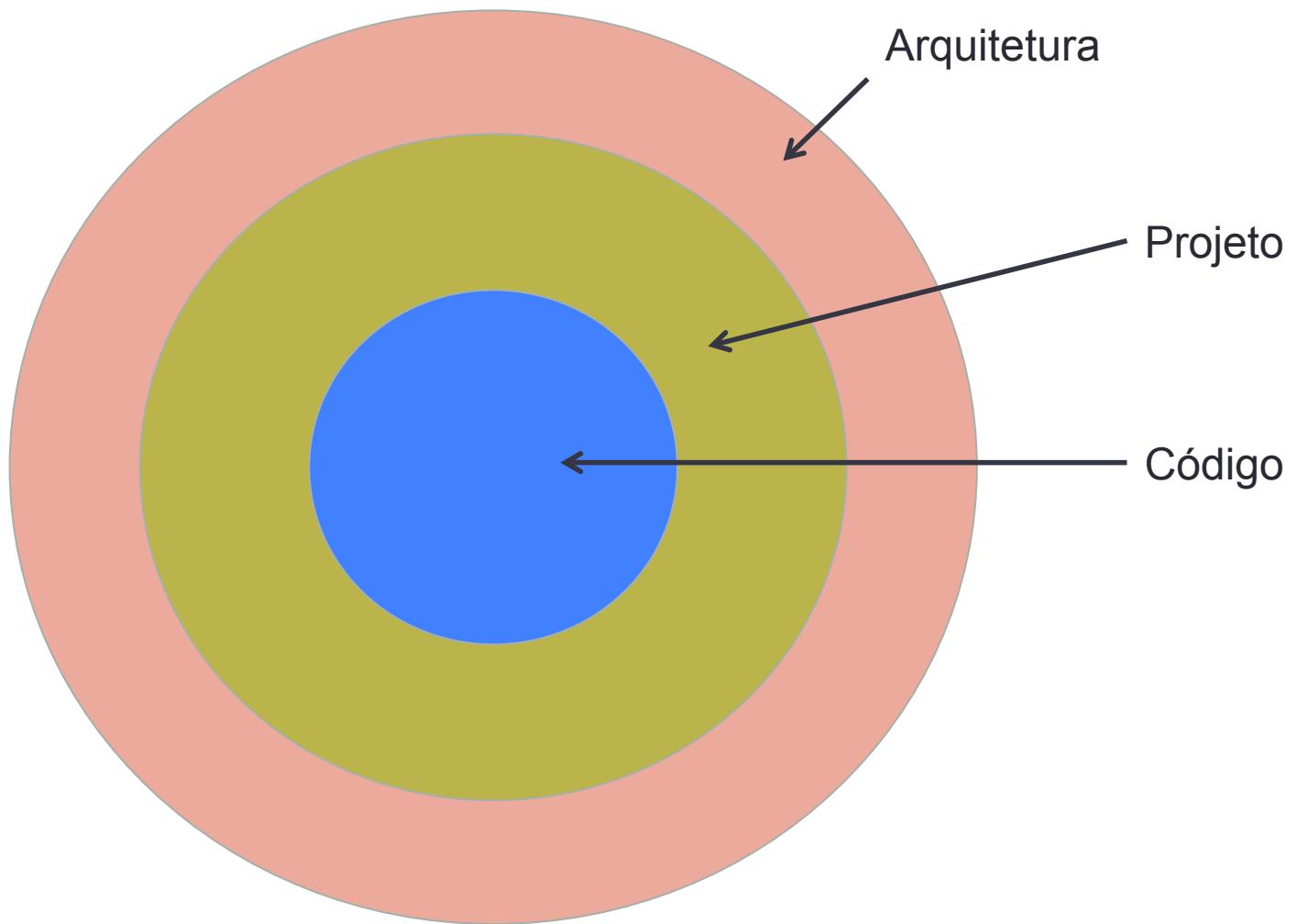
- Foca em questões como:

- Performance
- Confiabilidade
- Escalabilidade
- Testabilidade
- Manutenibilidade



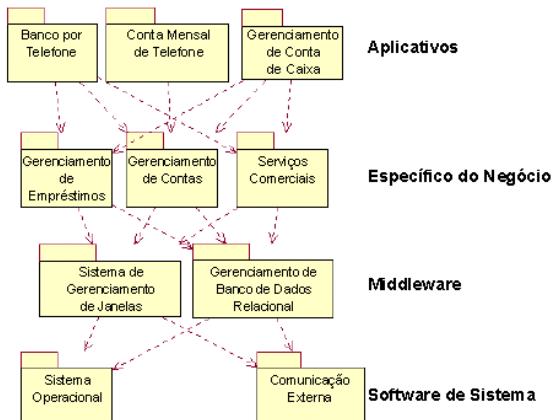
Requisitos não-funcionais!

Arquitetura de software

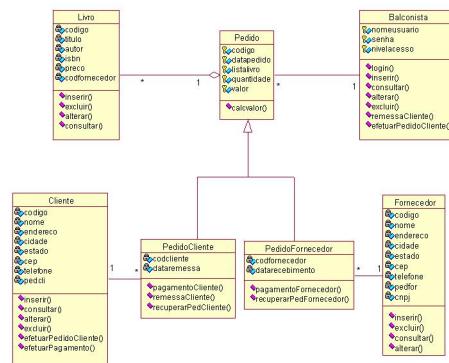


Níveis de abstração

Arquitetura



Projeto



Código

```

private final Command sairCommand = new Command("Sair");
private final TextBox textBox;

public HelloMasters() {
    textBox = new TextBox("Título do TextBox", "HelloMasters");
    textBox.addCommand(sairCommand);
    textBox.setCommandListener(this);
}

public void startApp() {
    Displayable telaAtual = Display.getDisplay(this).getCurrent();
    if(telaAtual == null) {
        Display.getDisplay(this).setCurrent(textBox);
    }
}

public void pauseApp() { }

```

Arquitetura de software

Vantagens de projetar e documentar a arquitetura:

- Facilitar a comunicação com os stakeholders
- Promover uma análise criteriosa sobre os requisitos críticos do projeto já no início do projeto, como:
 - Desempenho, confiabilidade e facilidade de manutenção
 - Projeto: manter dados off-line no mobile?
- Promover o reuso de software em larga escala, pois sistemas com requisitos semelhantes podem ter a mesma arquitetura
 - É possível elaborar um projeto de arquitetura para uma linha de produto

Impactos da escolha da arquitetura

Exemplos:

- Em um sistema onde a disponibilidade é um requisito crítico:
 - Inclua componentes redundantes
 - Projete de modo que possa ser possível substituir componentes sem parar o sistema
- Por outro lado, se o desempenho for um requisito crítico:
 - Utilize uma estrutura menos granular
 - Redução da comunicação entre os subsistemas

Conflitos na escolha da arquitetura

Exemplos:

- Desempenho:

- Estruturas menos granulares

- Manutenção

- Estruturas mais granulares

E se for preciso desempenho e facilidade de manutenção?

Arquiteto vai ter que escolher!

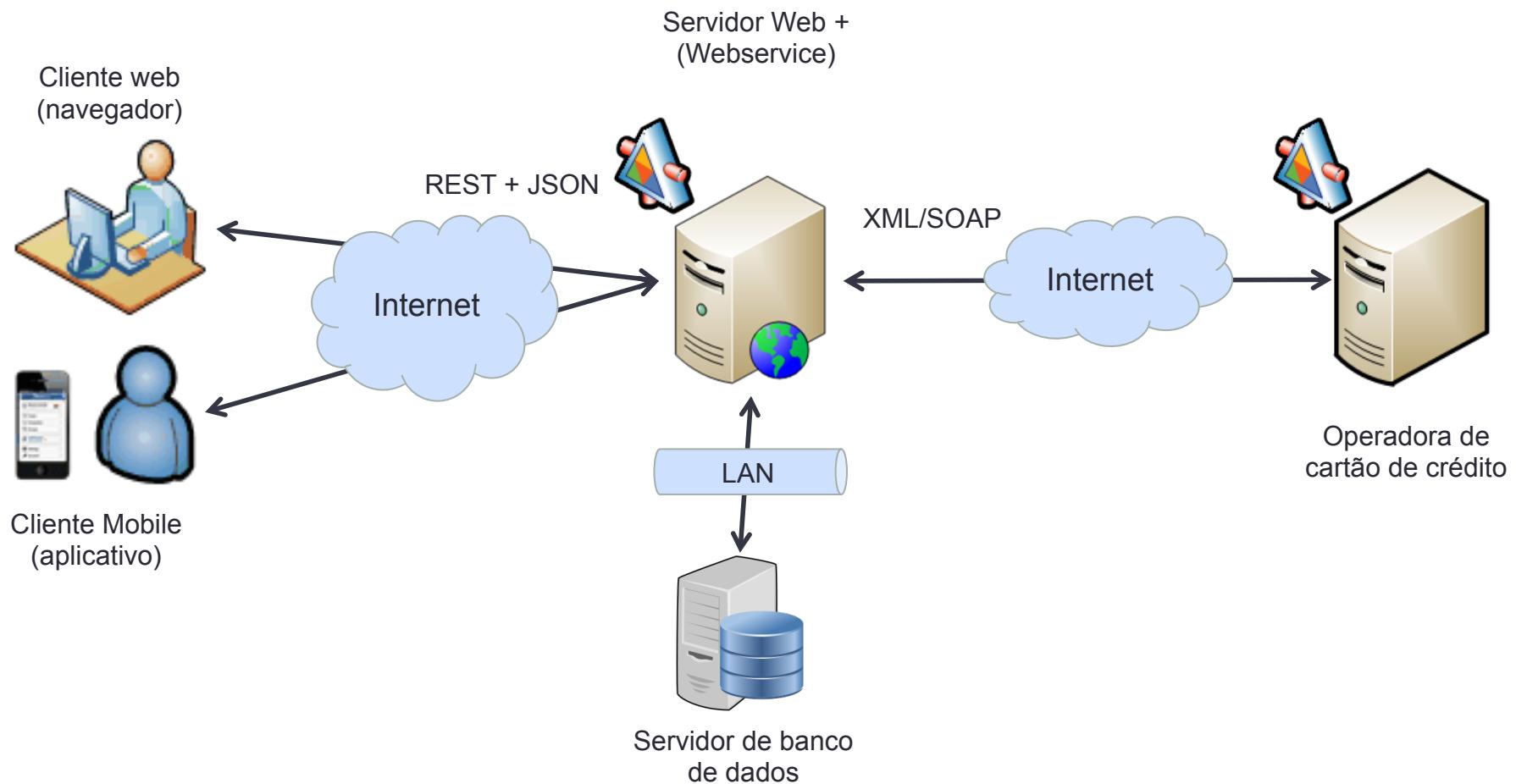
E documentar a decisão!

Documento de arquitetura

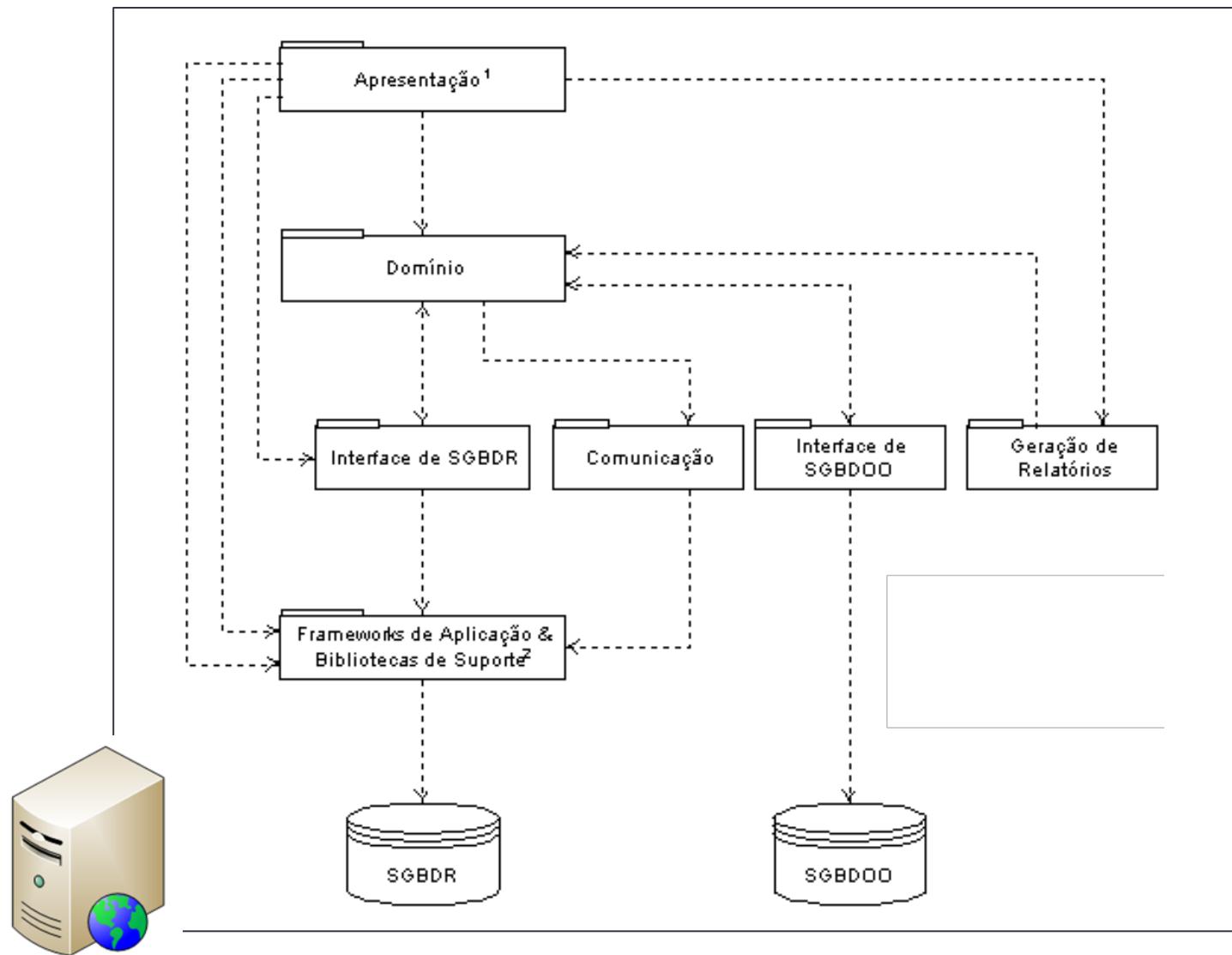
O Produto da fase de projetar arquitetura

- Stakeholders (usuários, sistemas externos, desenvolvedores)
- Representações gráficas da arquitetura
 - Componentes da arquitetura (servidores, clientes, banco de dados, subsistemas, camadas, padrões de projeto)
- Conexão entre os componentes
 - Chamada de rotinas, variáveis compartilhadas, protocolos cliente-servidor, broadcasting, eventos
- Decisões tomadas
 - Restrições

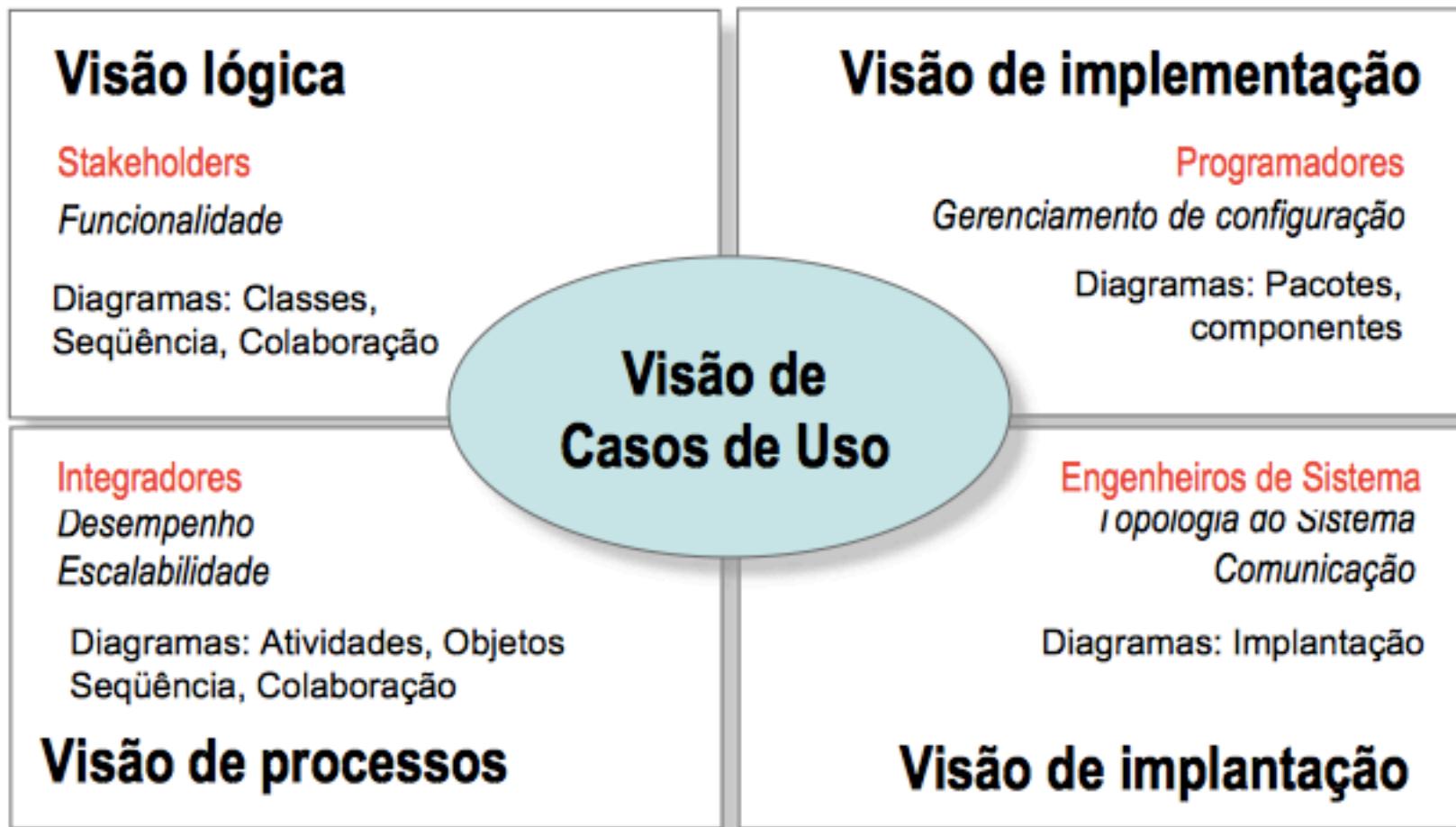
Exemplo de representação gráfica



Exemplo de representação gráfica

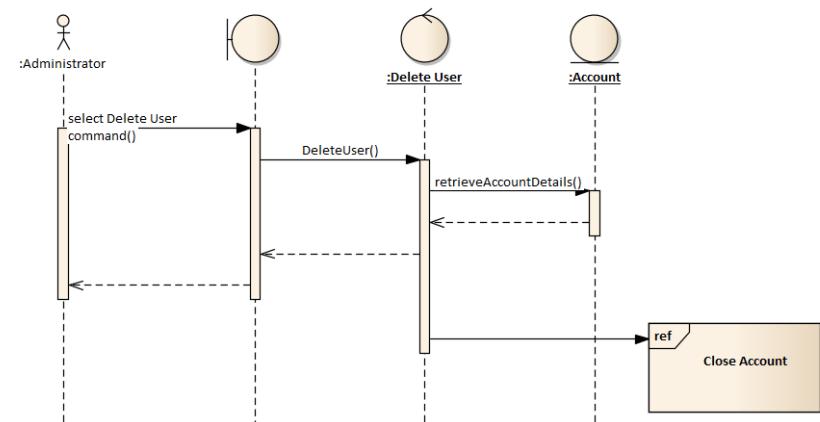
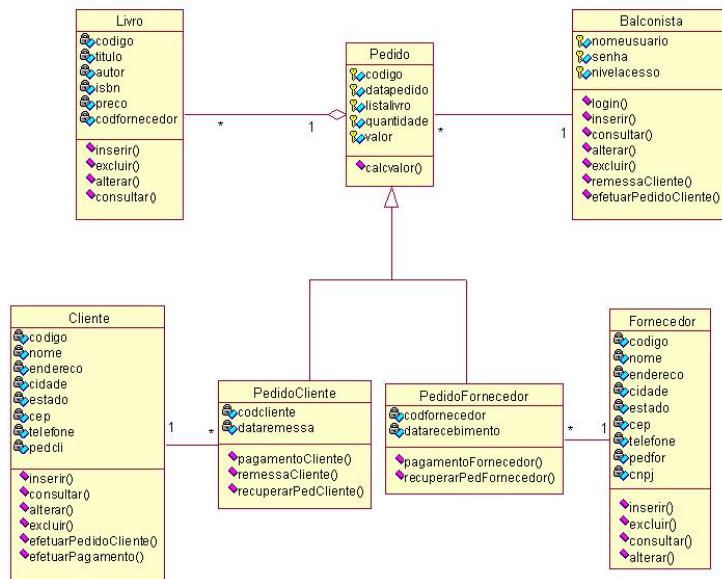


4 + 1 visões (P. Krutchen)



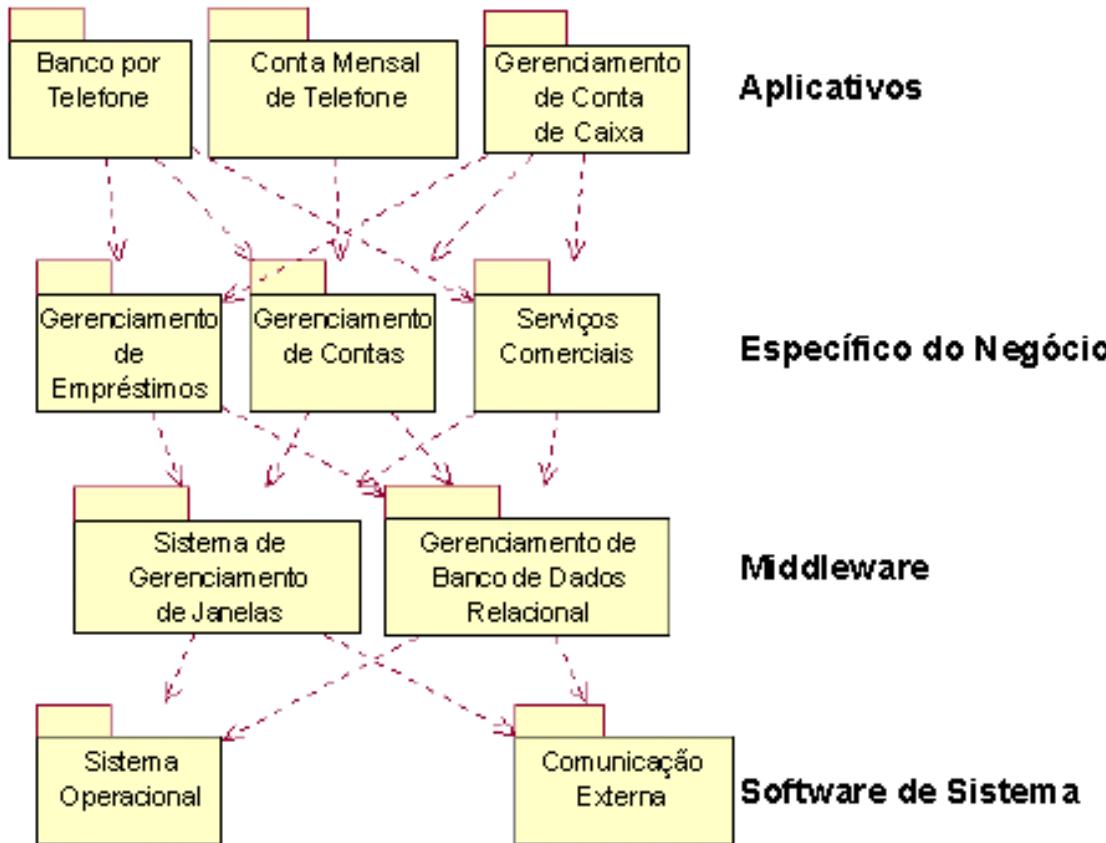
Visão lógica

- Descreve requisitos comportamentais e a decomposição do sistema em um conjunto de abstrações
- Diagramas de classes, sequência e colaboração



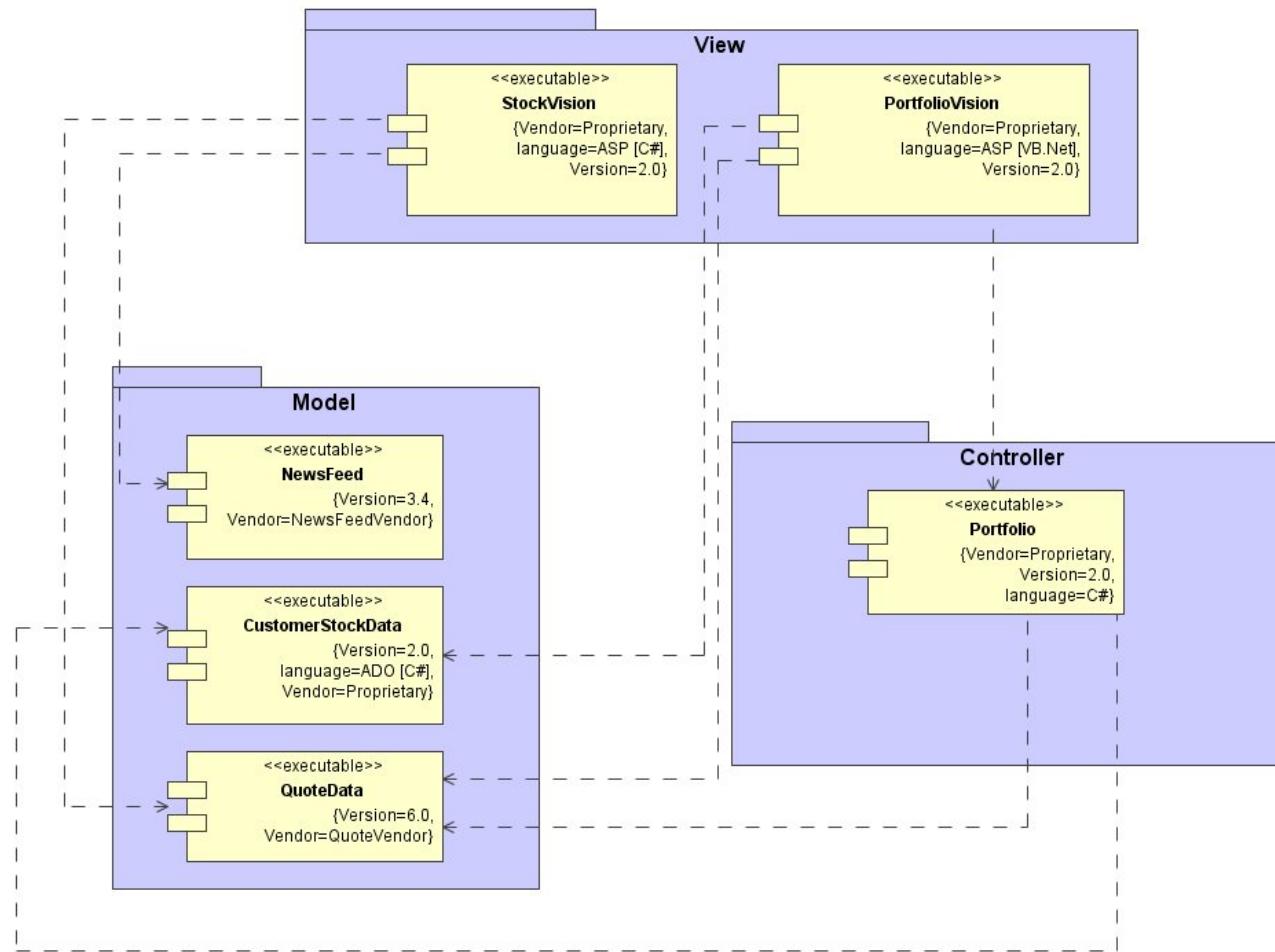
Visão de implementação

- Descreve os módulos do sistema
 - Módulos são mais abstratos que classes e objetos



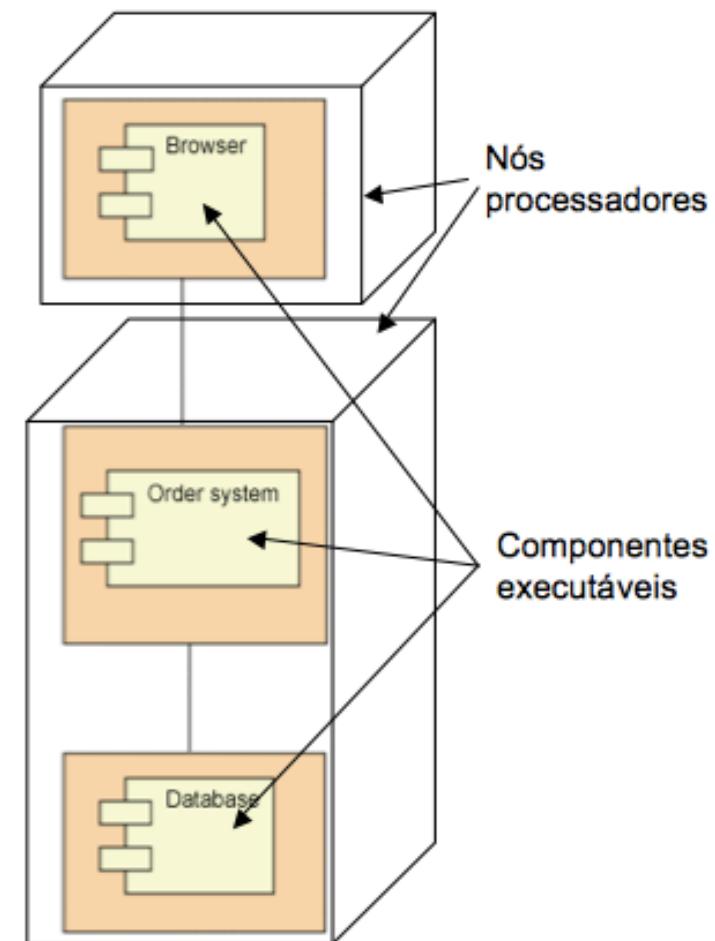
Visão de implementação

- Exemplo de um sistema usando MVC



Visão de implantação (física)

- Descreve como a aplicação é instalada e como executa em uma rede de computadores
 - Onde é executado (computador) cada componente da arquitetura?
- Visão usada para avaliar requisitos não-funcionais, como: desempenho, disponibilidade, confiabilidade

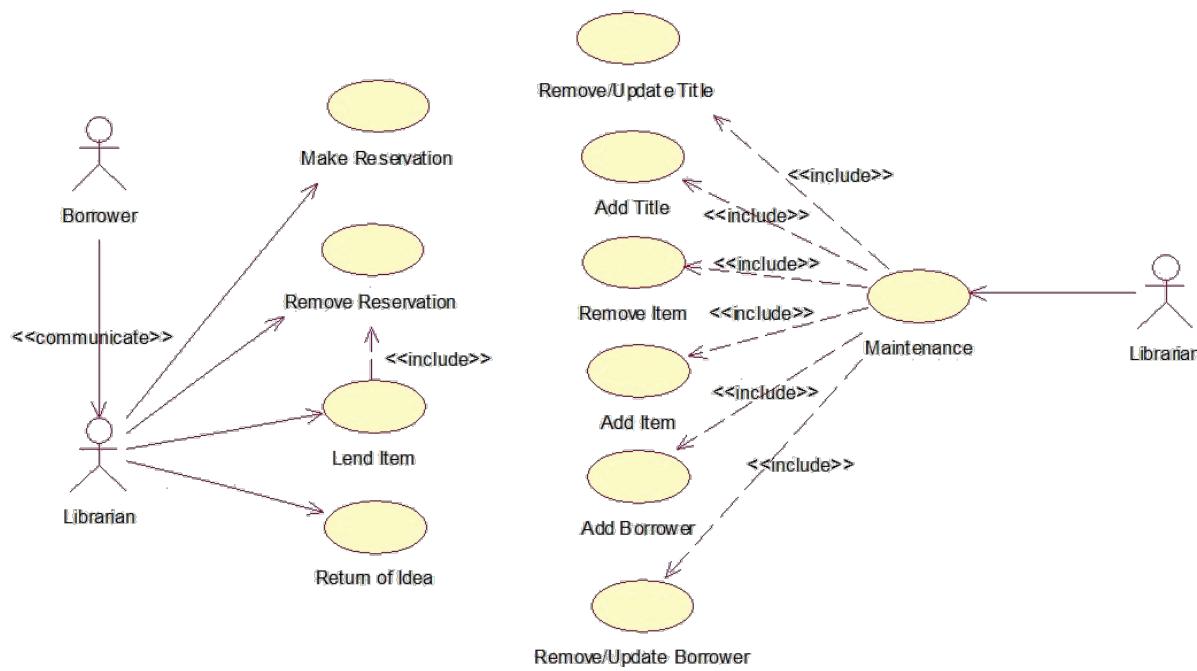


Visão de processo

- Descreve os **processos do sistema** e como eles se comunicam
- Útil quando se tem múltiplos processos ou threads concorrentes
- Diagramas de atividades podem ser usados

Visão de casos de uso

- Descreve a funcionalidade do sistema em termos de casos de uso



Documento de arquitetura

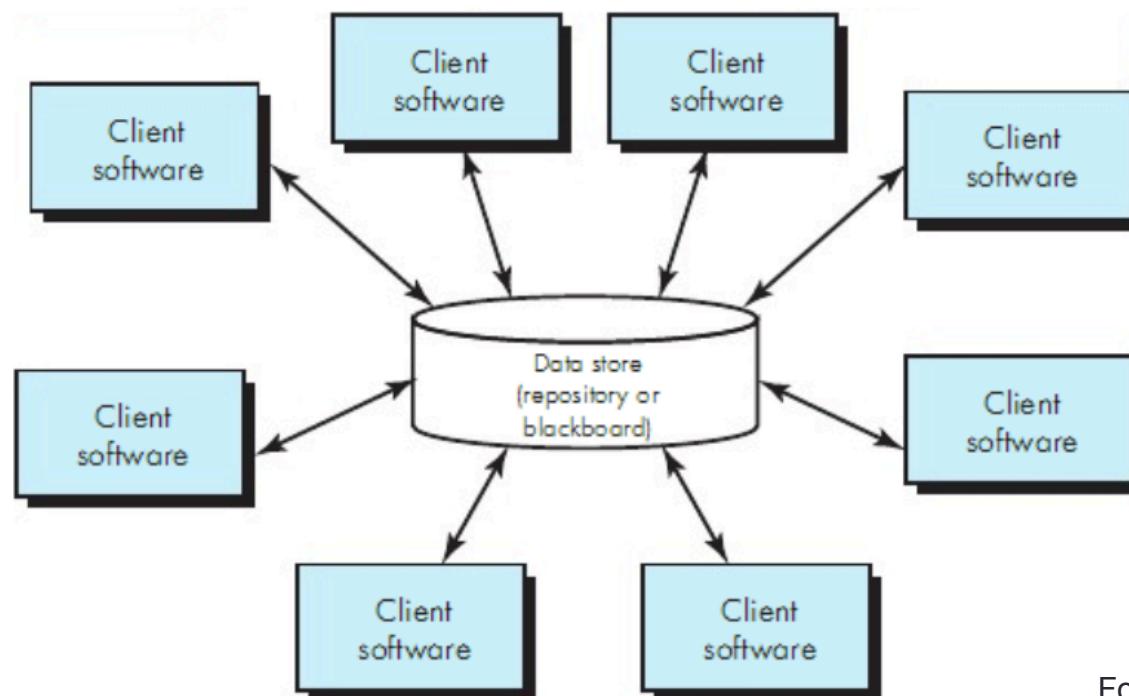
Qual modelo de documento de arquitetura usar no projeto?

Estilos arquiteturais

- Problemas se repetem na engenharia de software
- Documentou-se grupos de estilos capazes de tratar problemas recorrentes:
 - Arquiteturas centrada em dados
 - Arquiteturas de fluxo de dados
 - Sistemas distribuídos
 - Arquiteturas de chamada e retorno

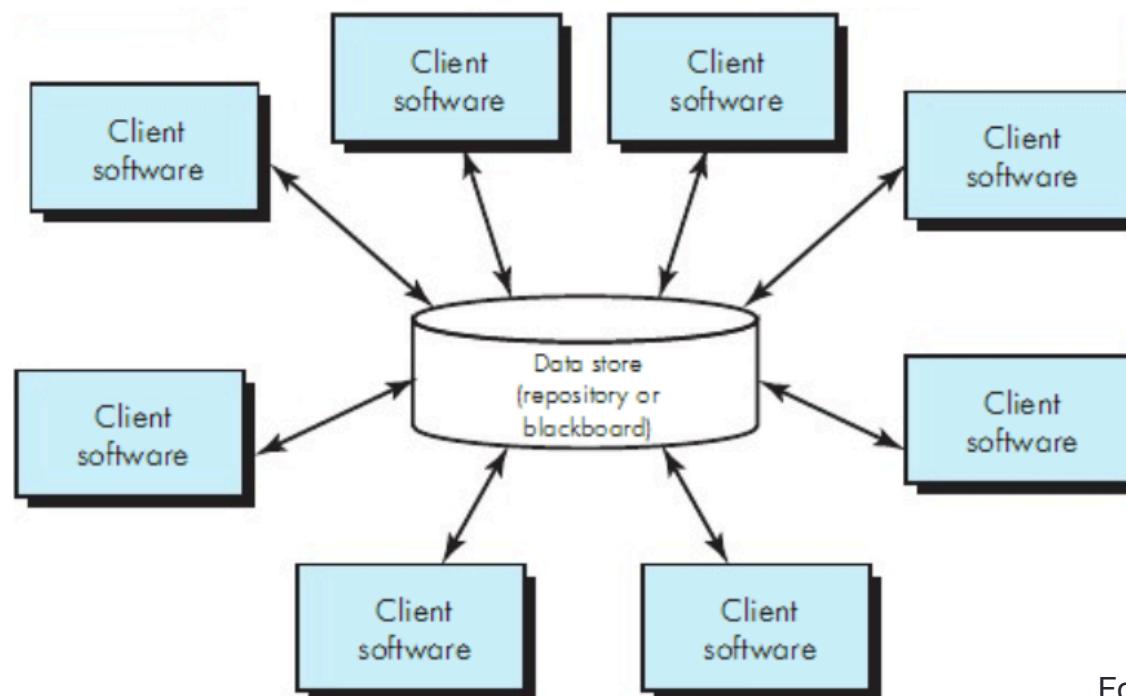
Arquitetura centrada em dados

- Repositório de dados é o centro da arquitetura
- Clientes (subsistemas, sistemas) acessam o repositório para inserir, recuperar, alterar, remover e também trocar dados entre eles



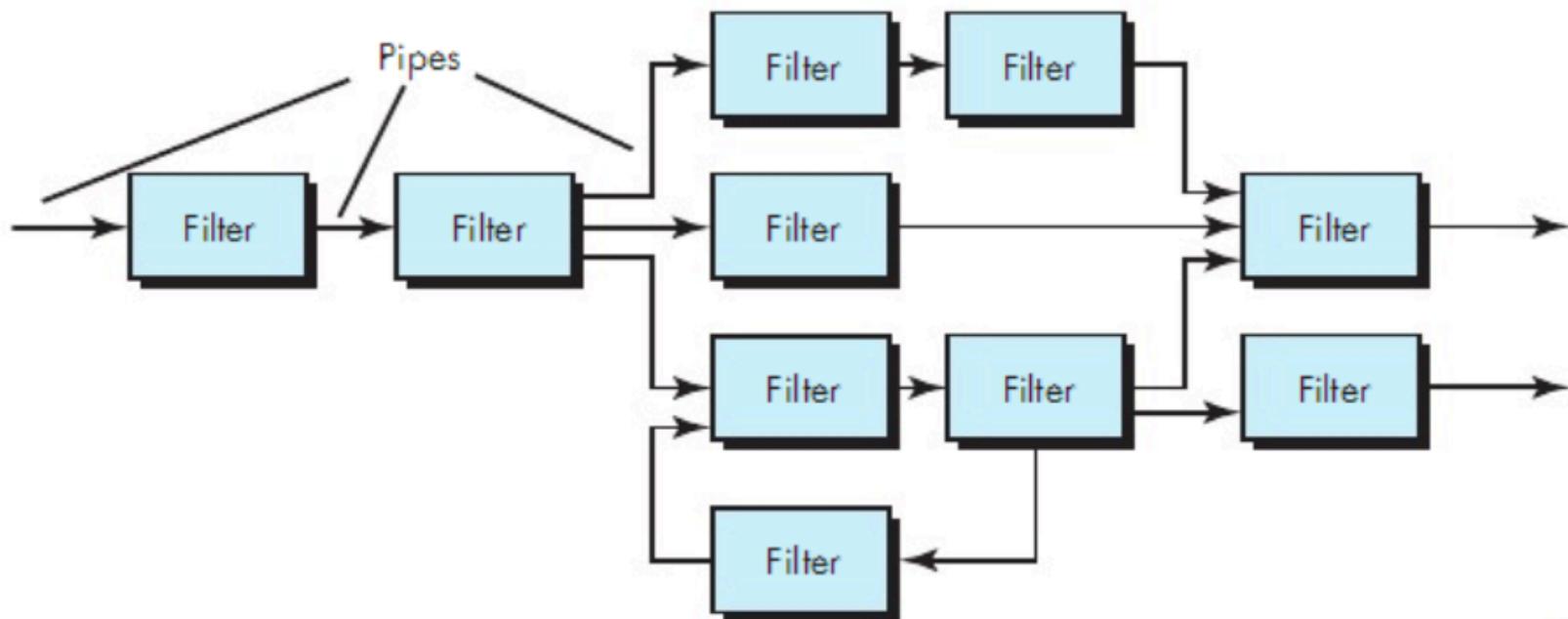
Arquitetura centrada em dados

- Promove a integração entre componentes e sistemas
- Componentes e sistemas podem mudar, ou serem adicionados novos, sem impactar nos demais



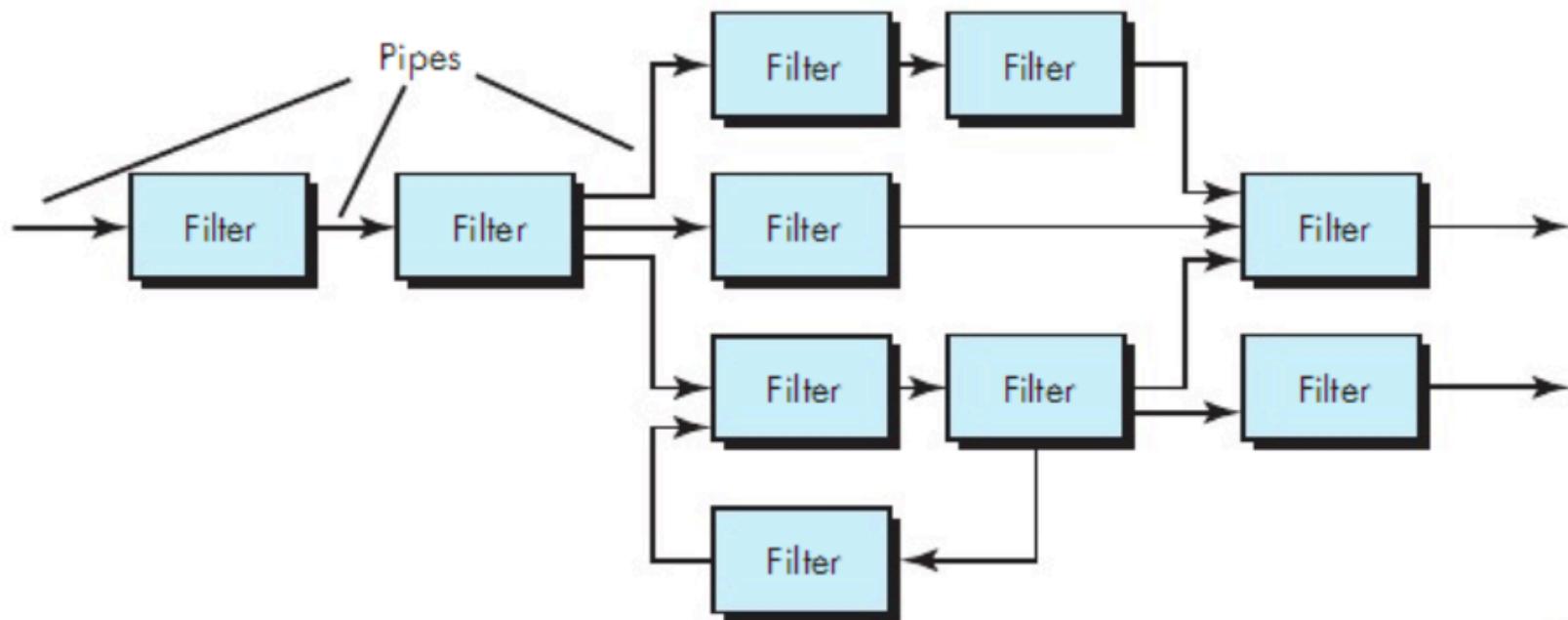
Arquitetura de fluxo de dados

- Dados de entrada são transformados em dados de saída através de uma série de cálculos e transformações
Ex: uso de filtros



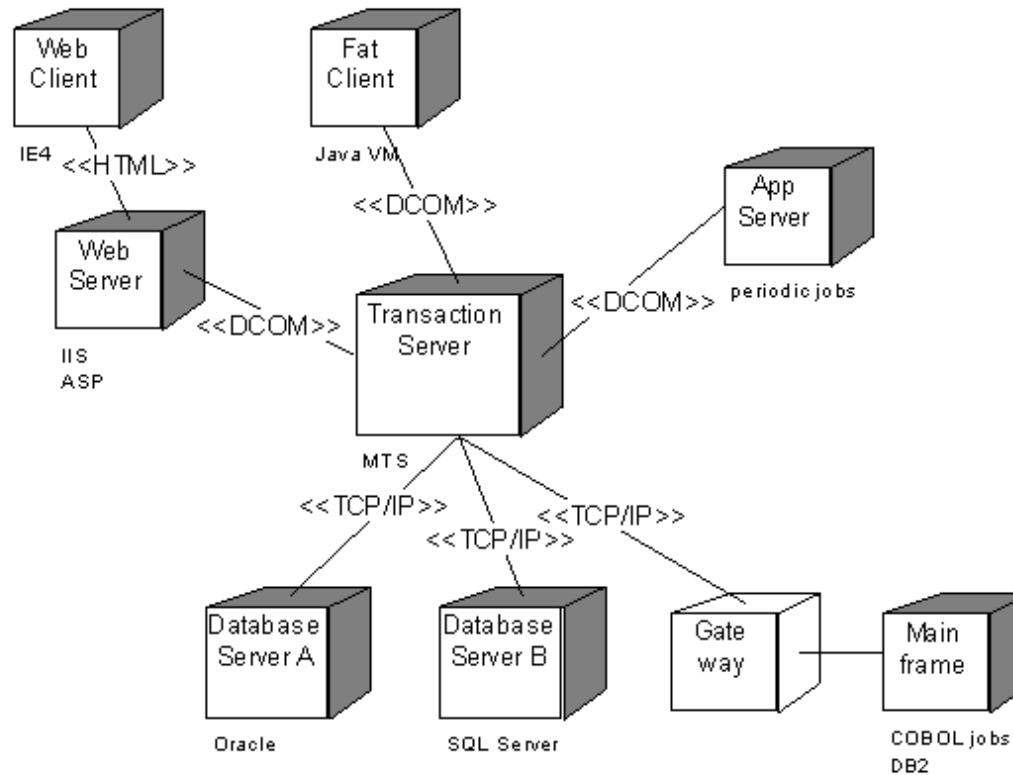
Arquitetura de fluxo de dados

- As entradas e saídas têm formato pré-definido
- Um componente não precisa saber como os outros funcionam



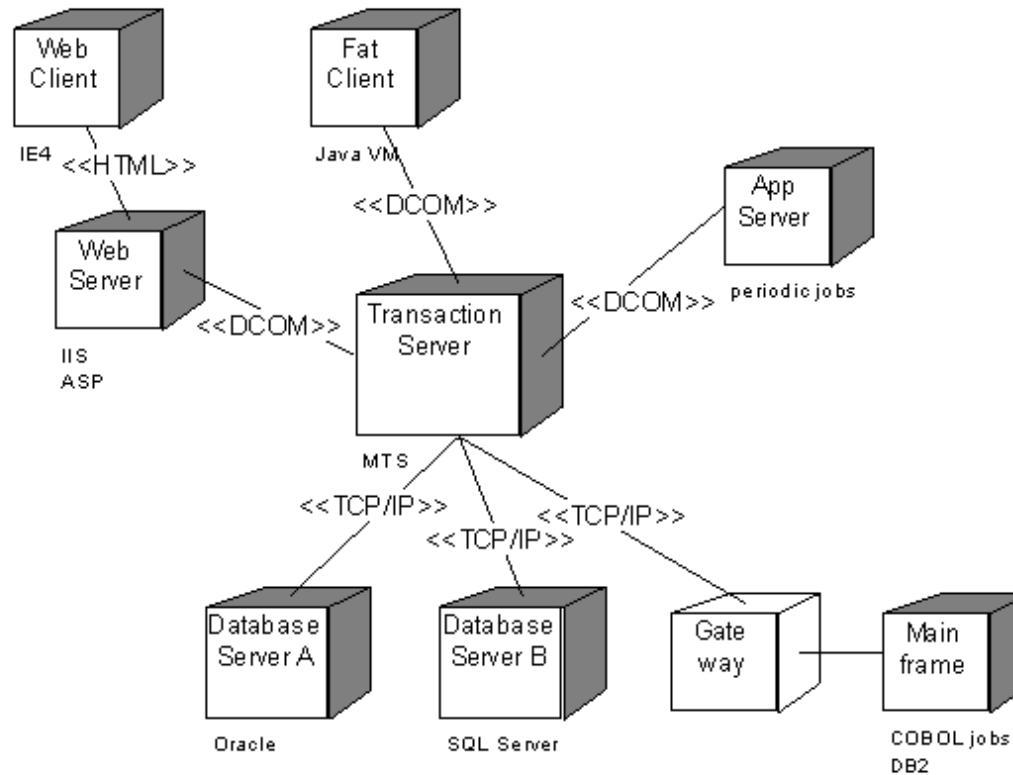
Sistema distribuído

- Módulos distintos cooperam
- Podem estar no mesmo nó ou em nós separados



Sistema distribuído

- Comunicação por protocolos abertos ou proprietários
- Aplicação em sistemas tolerante a falhas



Sistema distribuído

- Vantagens
 - Distribuição de carga
 - Reuso
 - Especialização
- Desvantagem
 - Performance/disponibilidade
 - Sujeito à conectividade
 - Segurança

Sistema distribuído

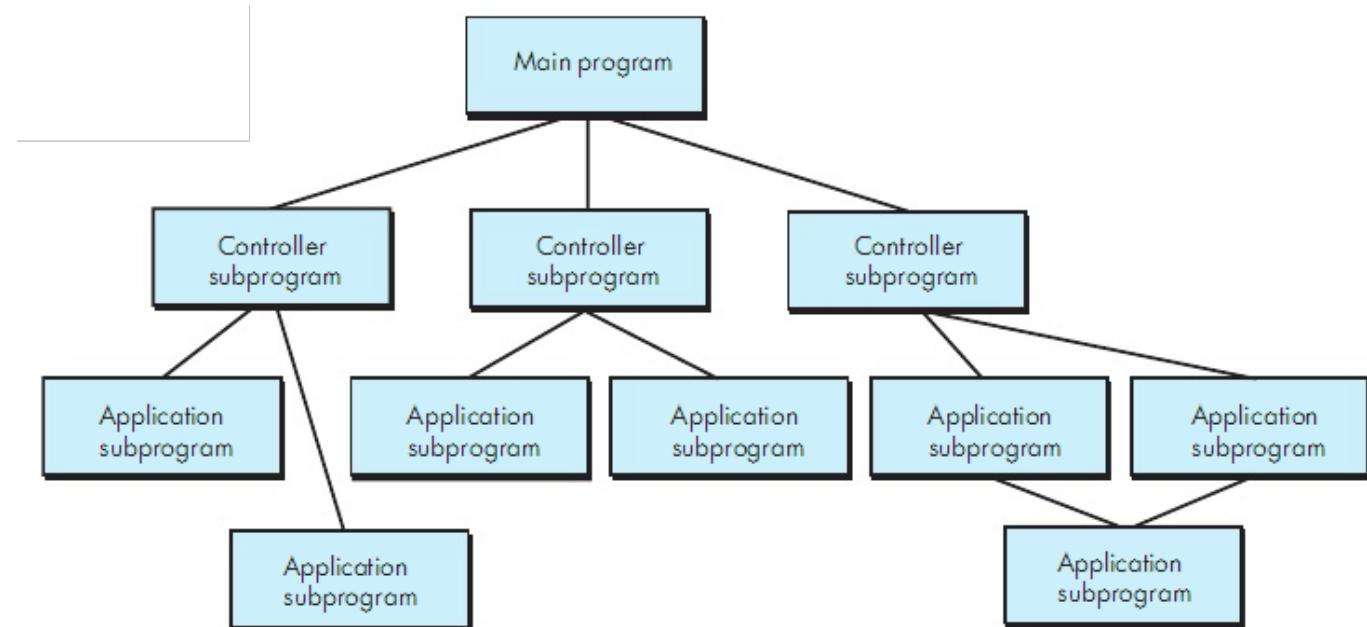
- Variação importante: Cliente-Servidor
 - Cliente: interface com o usuário (solicita informação ao servidor)
 - Servidor: serviço especializado (processamento, acesso a dados, regras de negócios)
 - Prove serviços para vários clientes
- Web
 - Servidor Web
 - IIS, Apache, Tomcat, etc.
 - Cliente
 - Navegador, Aplicativos, Softwares desktop

Arquitetura de chamada e retorno

- Possibilitam uma estrutura relativamente fácil de modificar e escalar
- Exemplos:
 - Arquitetura de Programa principal/Subprogramas
 - Arquitetura em camadas (OO)

Arquitetura Programa principal / subprograma

- Arquitetura hierarquizada
- Programa principal utiliza subprogramas
- Os subprogramas pode utilizar outros subprogramas (e assim por diante)

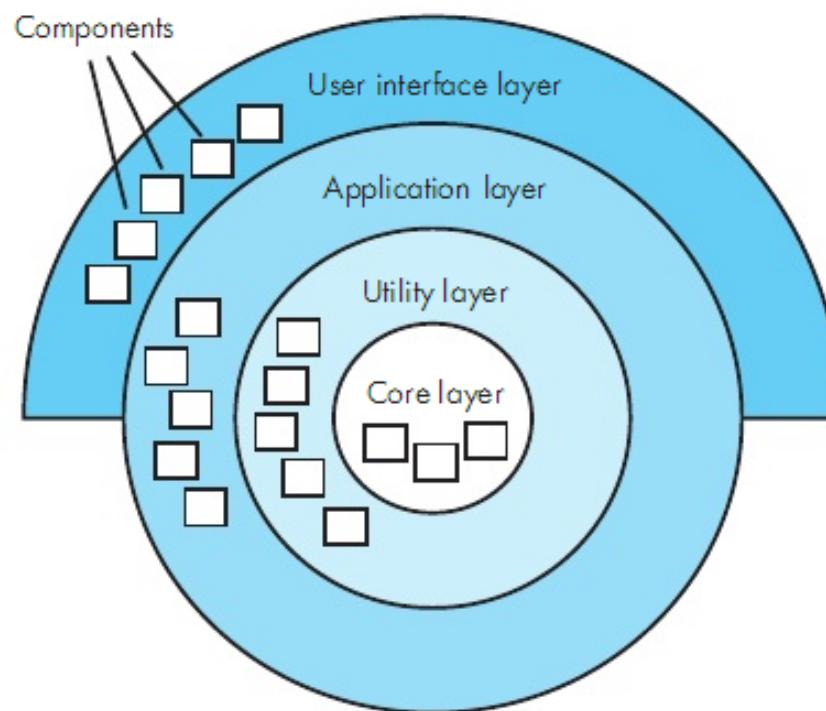


Arquitetura orientada a objetos

- Componentes do sistema são encapsulados em objetos
- Objetos contêm os dados (atributos) e fornecem operações para manipulá-los (métodos, funções)
- A comunicação e coordenação entre os componentes é realizado através da troca de mensagens (métodos dos objetos)

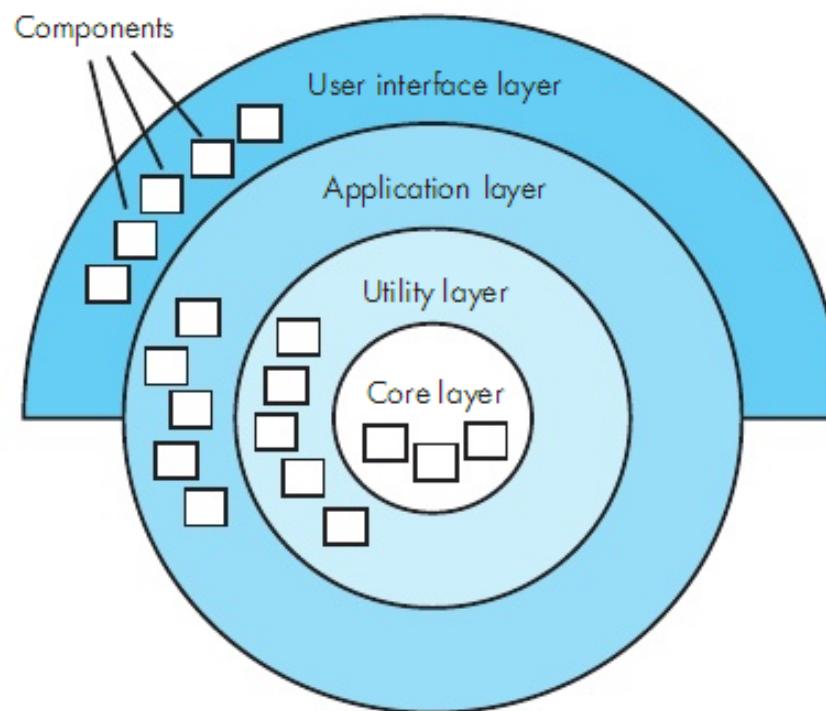
Arquitetura dividida em camadas

- Bastante utilizada em conjunto com a arquitetura orientada a objetos
- A camada mais externa possui componentes de interface com o usuário



Arquitetura dividida em camadas

- As camadas intermediárias possuem as regras de negócio do sistema e os serviços utilitários
- As camadas mais internas possuem componentes de interface com o sistema operacional / repositório de dados



Uso de estilos arquiteturais

- Pode-se utilizar mais de um estilo no mesmo sistema

Ex:

- Arquitetura em camadas + centradas em dados

Uso de estilos arquiteturais

Escolhas comuns

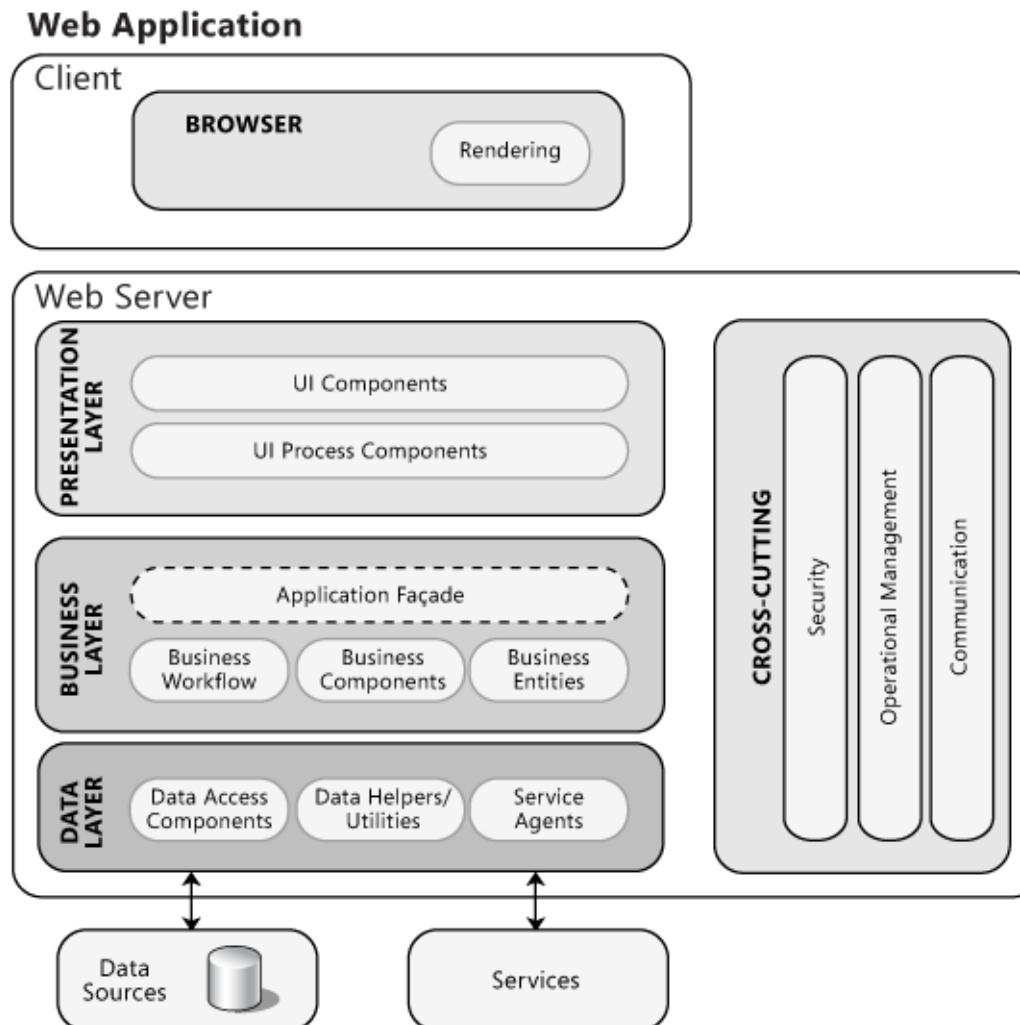
- Sistemas onde existem apenas requisitos funcionais e simples -> **arquitetura simples / monolítica**
- Sistemas que exigem confiabilidade -> **Redundância**
- Sistemas de alto desempenho -> **Concorrência**
- Sistemas fáceis de modificar -> **Camadas**

ARQUITETURA DE SISTEMAS WEB

Arquitetura de sistemas web

- Sistemas web têm sido bastante utilizados
 - Cliente (browser) – Servidor (servidor web – IIS, Apache, Tomcat)
- Alguns estilos têm se consolidado
 - Arquitetura em camadas
 - MVC
 - Service Oriented Architecture (SOA)

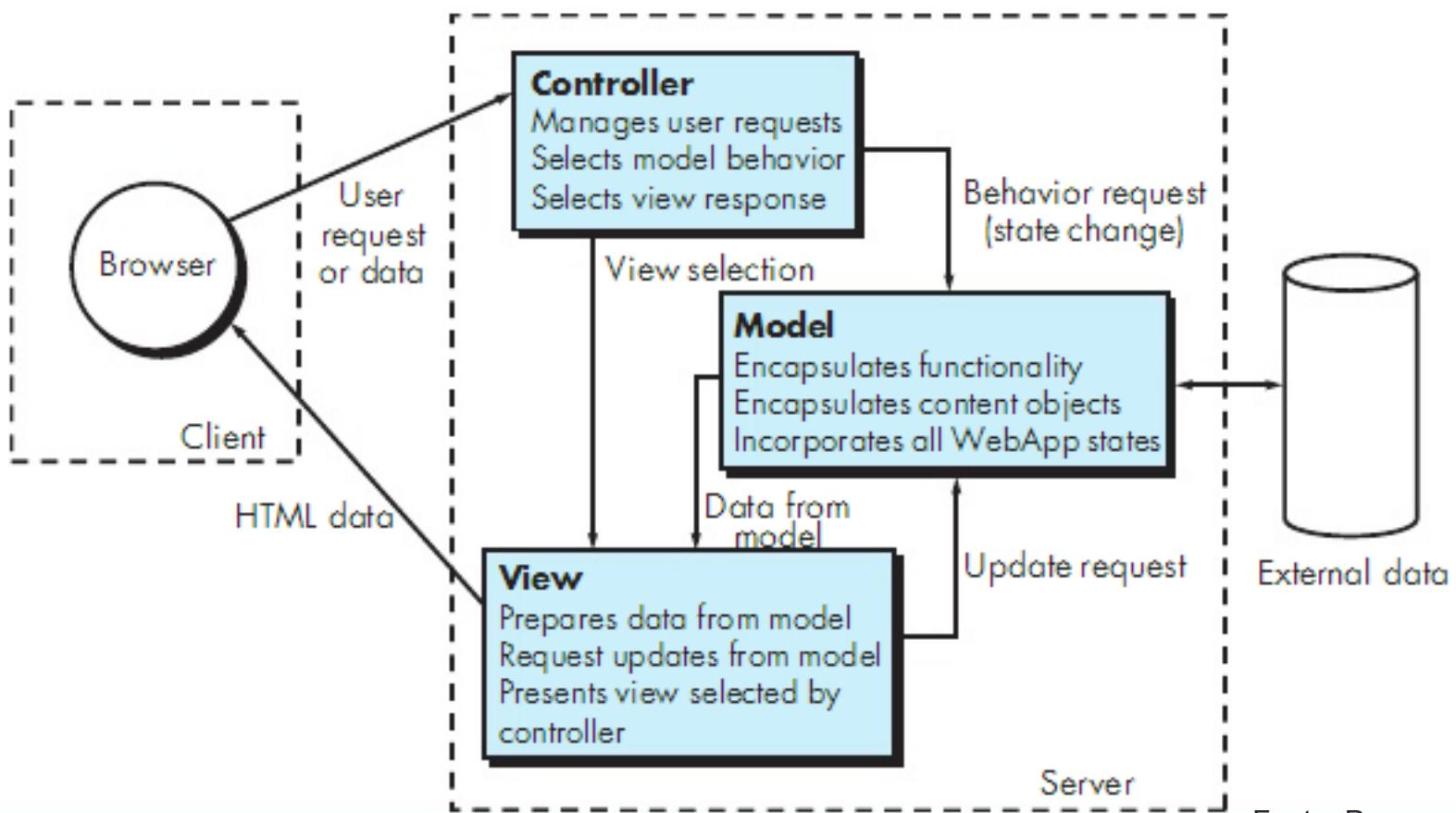
Arquitetura de sistemas web



Fonte: <https://msdn.microsoft.com/en-us/library/ee658099.aspx>

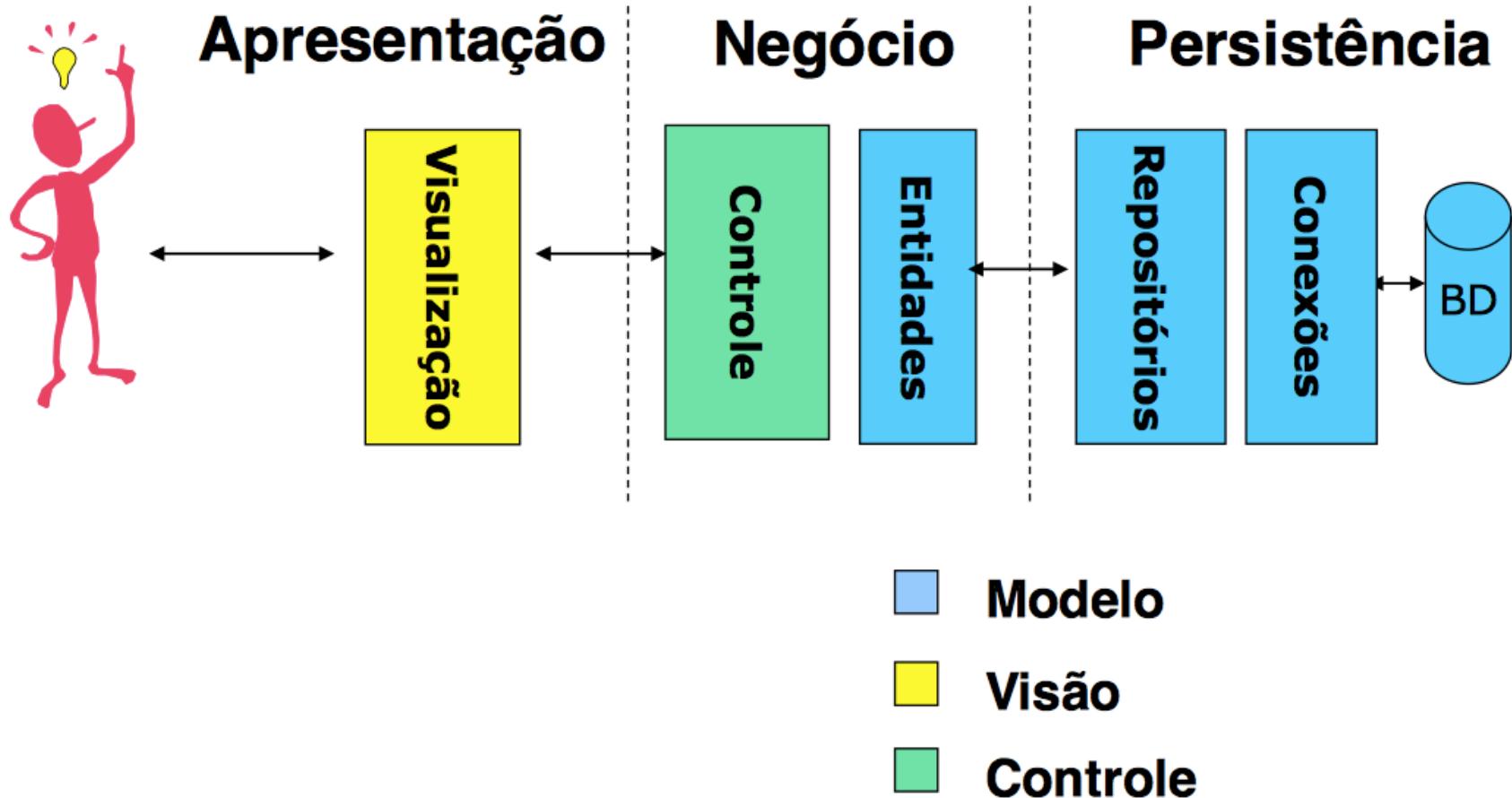
Arquitetura de sistemas web

MVC



Fonte: Pressman, 2014

Padrão MVC



Outros padrões

MVP - Model View **P**resenter

MVA – Model View **A**dapter

MVVP – Model View ViewModel

Necessidades <-> Arquitetura

- Facilidade de identificação de problemas
 - Log
 - Como vocês vão fazer isso no projeto?
- Deploy contínuo
 - Nova versão colocada automaticamente no servidor de testes ou produção
 - Testes automáticos validados
 - Colocar novas versões no ar sem parar o sistema
 - Como vocês vão fazer isso no projeto?
- Monitoramento do uso do sistema
 - Usuários únicos, tempo de carregamento das informações, telas mais acessadas, erros
 - Como vocês vão fazer isso no projeto?

ARQUITETURA DE SISTEMAS MOBILE

Arquitetura de sistemas mobile

Única camada (*thin client*)

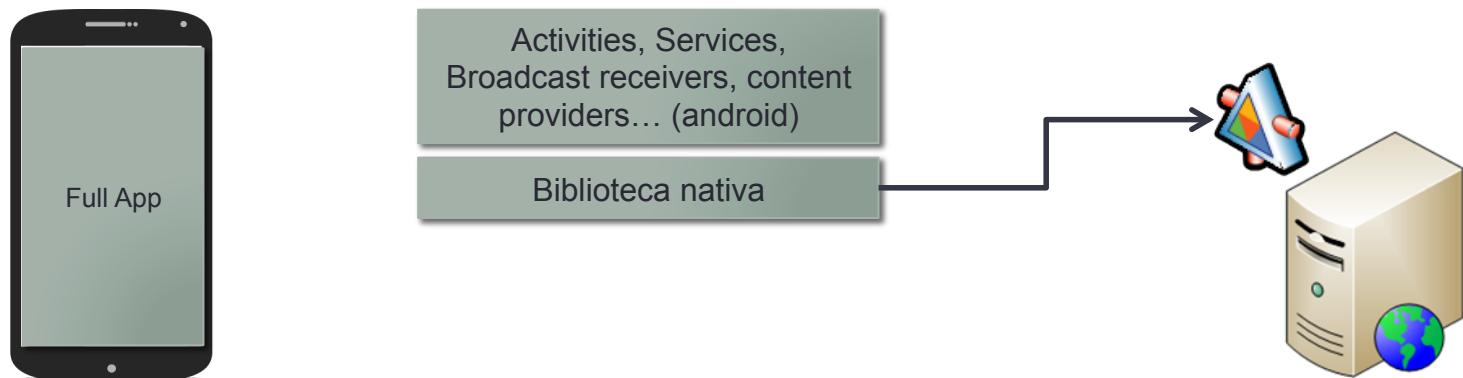
- View – Browser
- Vantagem:
 - Cross-plataforma (develop once, run anywhere)
 - Aproveita a familiaridade de desenvolvedores web
- Desvantagens
 - Suporte limitado a serviços nativos (câmera, calendário, sensores)
 - Funcionalidades off-line limitadas



Arquitetura de sistemas mobile

Multicamadas (*rich client*)

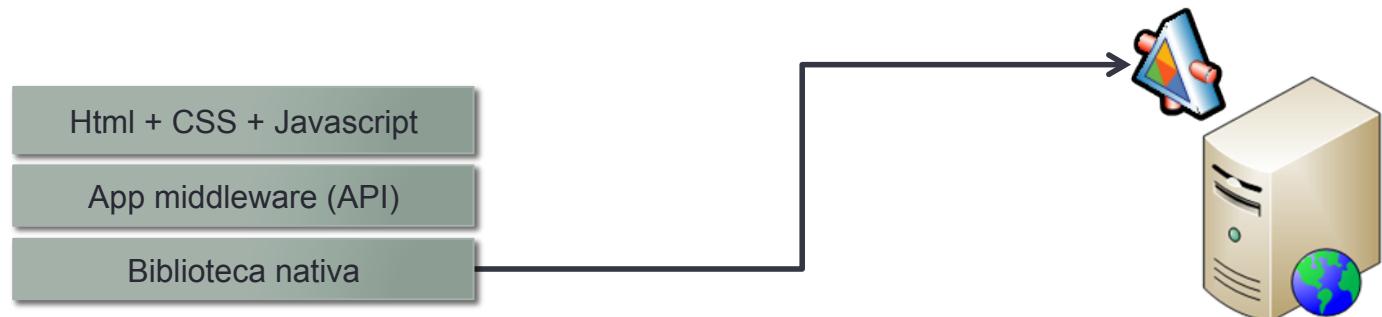
- Vantagens:
 - Melhor performance
 - Disponibilidade off-line
 - Acesso a serviços nativos (câmera, apps, sensores)
- Desvantagens
 - Específico para uma plataforma
 - Curva de aprendizado de uma nova tecnologia (APIs, Arquitetura)



Arquitetura de sistemas mobile

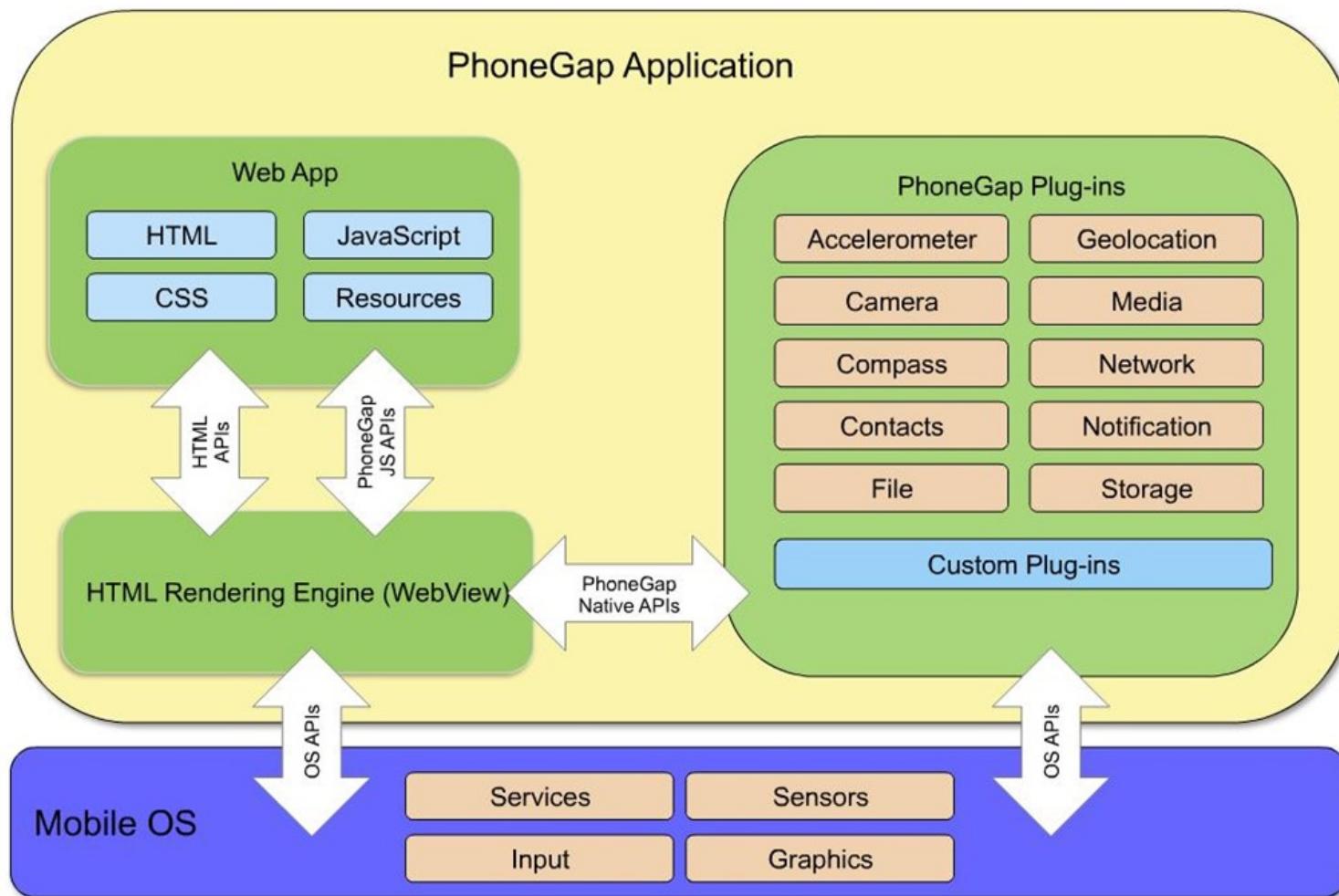
Híbrido (*Phone gap, Titanium, AppDeck, etc.*)

- Vantagens:
 - HTML + Javascript (familiaridade de desenvolvedores web)
 - Pode ter disponibilidade off-line
 - Cross-plataform
- Desvantagens
 - Ainda precisa de código específico para cada plataforma
 - Curva de aprendizado de uma nova tecnologia (APIs, Arquitetura)
 - Acesso limitado a serviços nativos (câmera, apps, sensores)



Arquitetura de sistemas mobile

Ex: Phone Gap



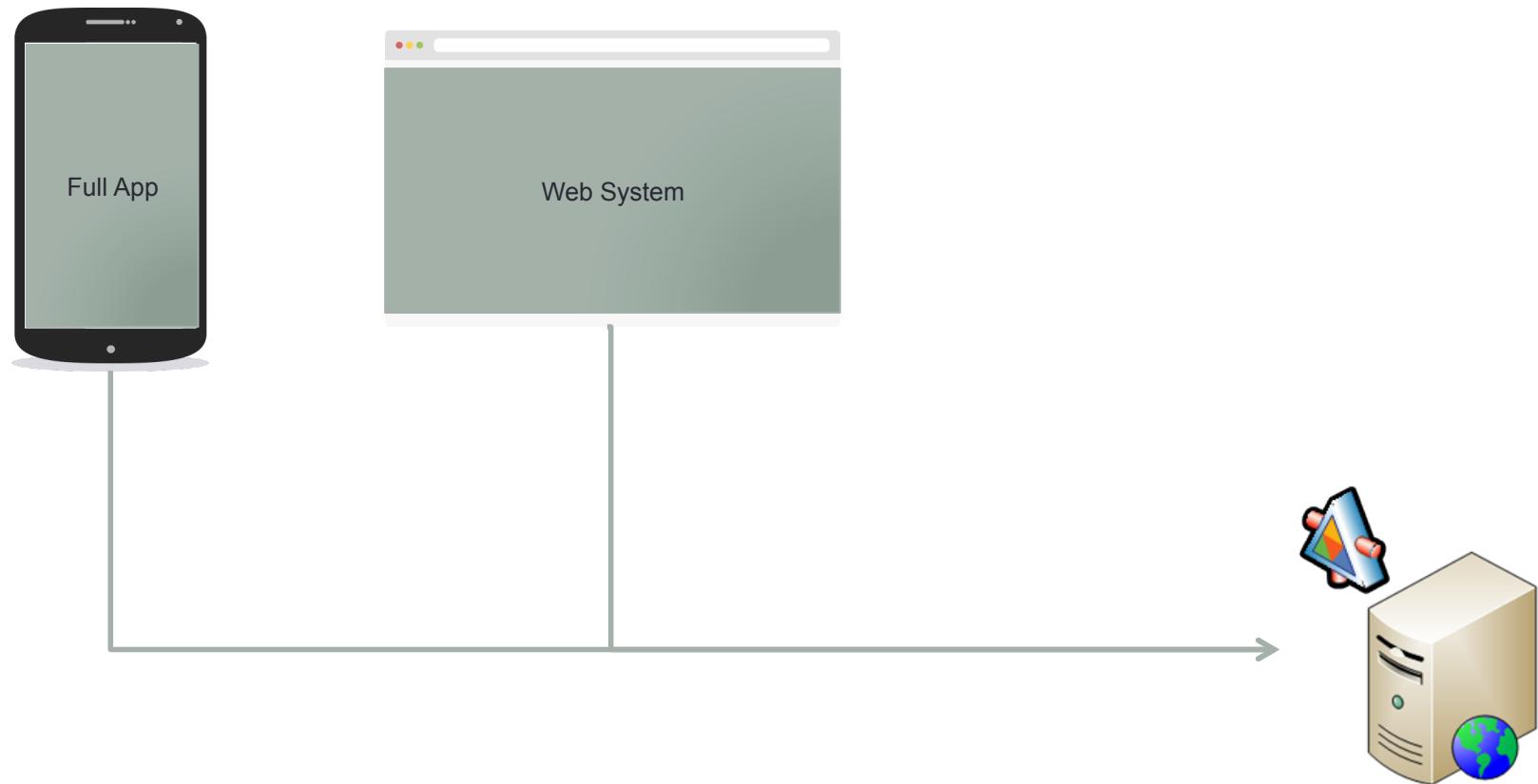
Arquitetura de sistemas mobile

Particularidades

- Diferentes tamanho de tela
- Capacidade de processamento
- Recursos disponíveis (GPS, tamanho da memória, acelerômetro, capacidade de bateria, recursos de interface com o usuário, disponibilidade de conectividade, ...)

Sistemas web + Mobile

- Compartilhamento de informações

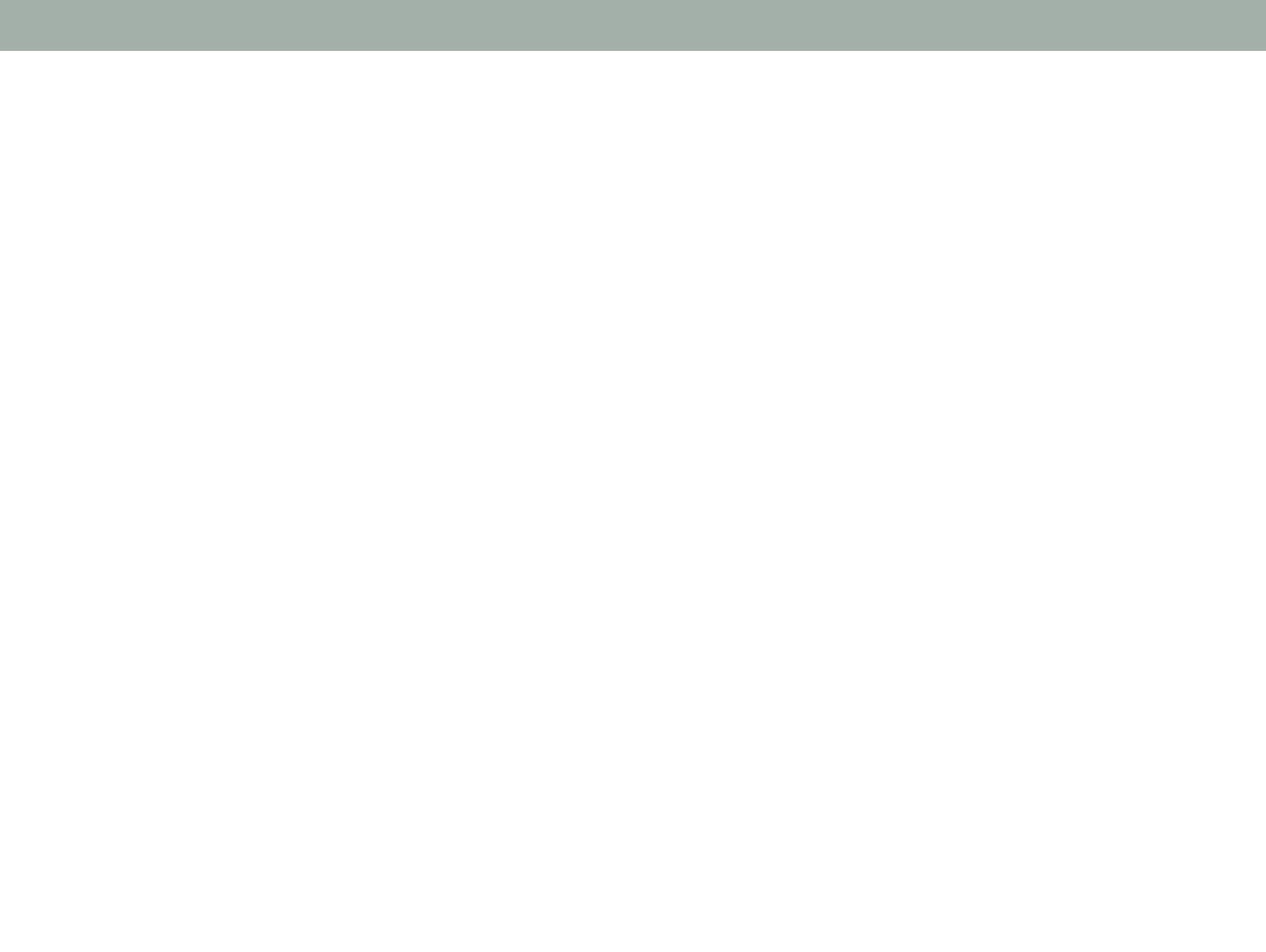


Sistemas web + Mobile

- Onde ficam as regras de negócio? Duplicadas? Reuso?

SOA

Service Oriented Architecture



Decisões!

- Qual será o modelo de documento de arquitetura usado?
- Vai haver biblioteca de utilitários do sistema? Onde fica?
- Como será implementado o log de erros?
- Qual protocolo de comunicação Servidor-App será usado no projeto?
- Como incorporar interface Web futuramente?
- Os dados vão ficar off-line no mobile? Como?
- Como fazer deploy automático e contínuo?
- Como monitorar o uso do sistema?

WHAT'S
YOUR
CHOICE?