# Arquitetura de Software

Instituto Federal do Espirito Santo – Campus Colatina
Disciplina: Projeto de Sistemas
Professor: Giovany Frossard Teixeira

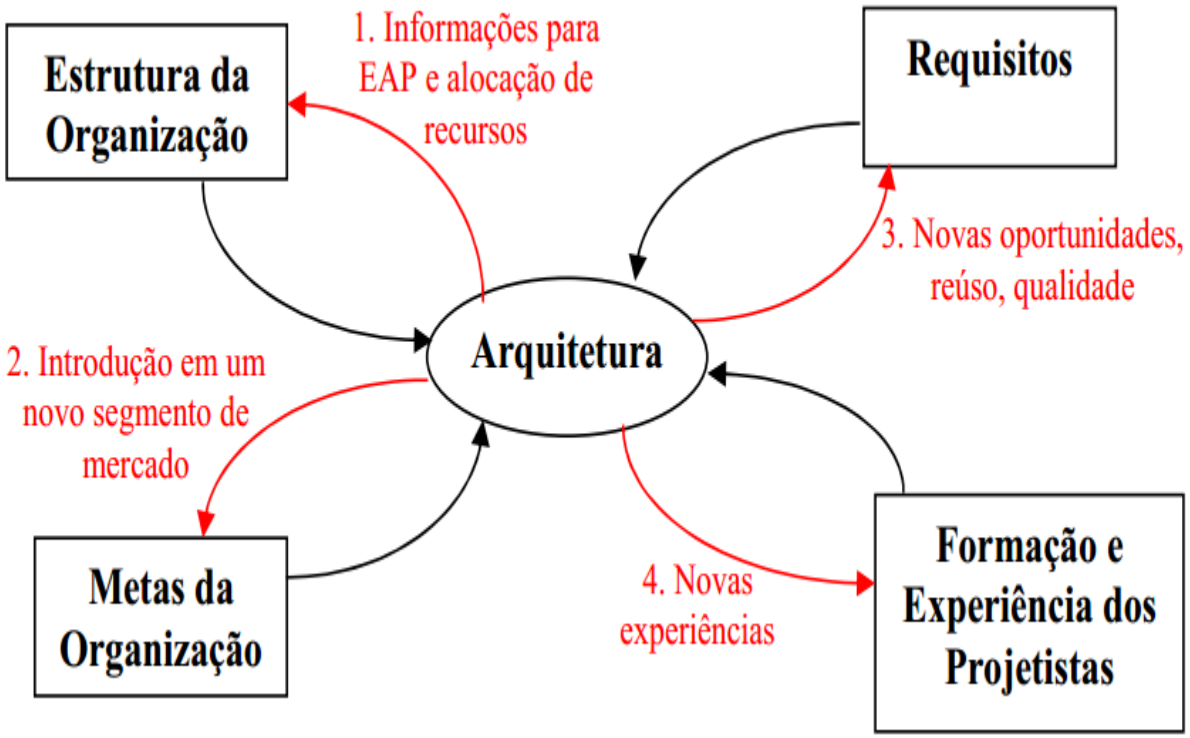
# Definição

A arquitetura de software refere-se às estruturas de **grandes sistemas** de software. A visão arquitetônica de um sistema **é abstrata**, retirando de cena detalhes de implementação, algorítmicos e de representação de dados e procurando se concentrar no comportamento e interação entre elementos considerados caixas pretas.

# Arquitetura do Software e suas Influências

1. A arquitetura afeta a estrutura da organização de desenvolvimento. Ao prescrever uma estrutura para um sistema, a arquitetura prescreve também as unidades de software a serem implementadas (ou obtidas) e integradas para formar o sistema. Essas unidades formam a base para o desenvolvimento da Estrutura Analítica do Projeto (EAP) e para a alocação de equipes e pessoas às unidades da EAP.

2. A arquitetura pode afetar as metas da organização de desenvolvimento, uma vez que um sistema bem sucedido, construído a partir dela, pode habilitar a organização a estabelecer uma base em uma particular área de mercado.




3. A arquitetura pode afetar os requisitos do cliente para novos sistemas, ao dar ao cliente a oportunidade de receber um sistema baseado na mesma arquitetura de maneira mais econômica, rápida e confiável do que se o mesmo sistema tivesse que ser desenvolvido a partir do zero.

4. O processo de construir o sistema baseado na arquitetura afeta a experiência do projetista.

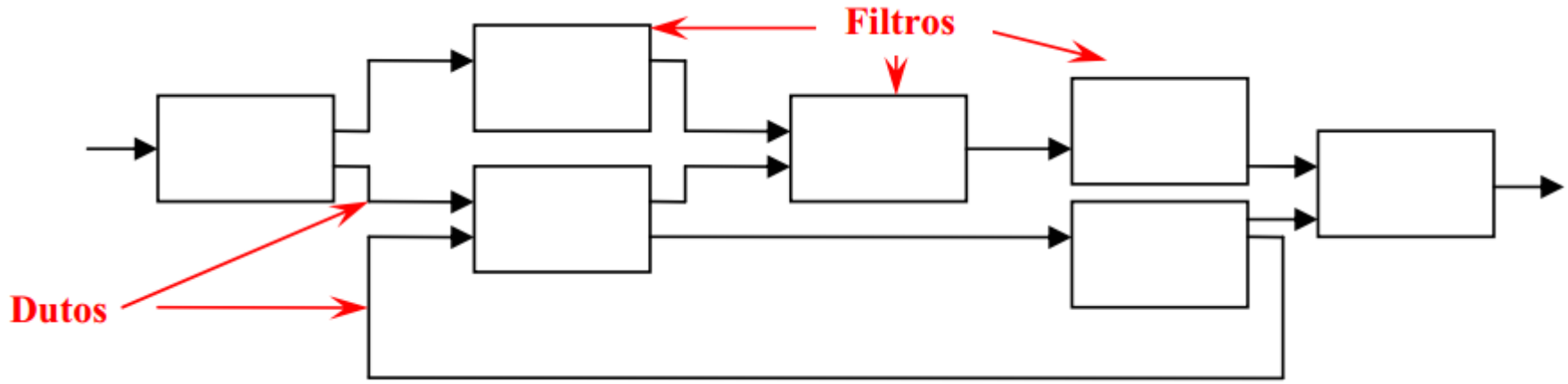
# Classes de Sistemas

- Sistemas similares muito provavelmente terão arquiteturas similares. Existem vários tipos de classes de sistemas. Vamos focar nossos estudos nos sistemas de informação.
- Sistemas de Informação (SIs) são responsáveis por coletar, manipular e preservar grandes quantidades de informações complexas.
- Sistemas podem ser classificados em aplicações desktop, aplicações web e aplicações móveis.
  - **Aplicações Desktop**: executam em estações de trabalho e podem utilizar recursos dessas máquinas. Podem ser executadas em uma máquina ou em várias (aplicações distribuídas).
  - **Aplicações Web**: utilizam o navegador e estão a cada dia mais ricas no que se refere a interfaces com o usuário.
  - **Aplicações móveis**: executam em máquinas com baixo poder de processamento (apesar de gradualmente essa diferença estar diminuindo em relação a estações de trabalho tradicionais).

# Estilos arquitetônicos

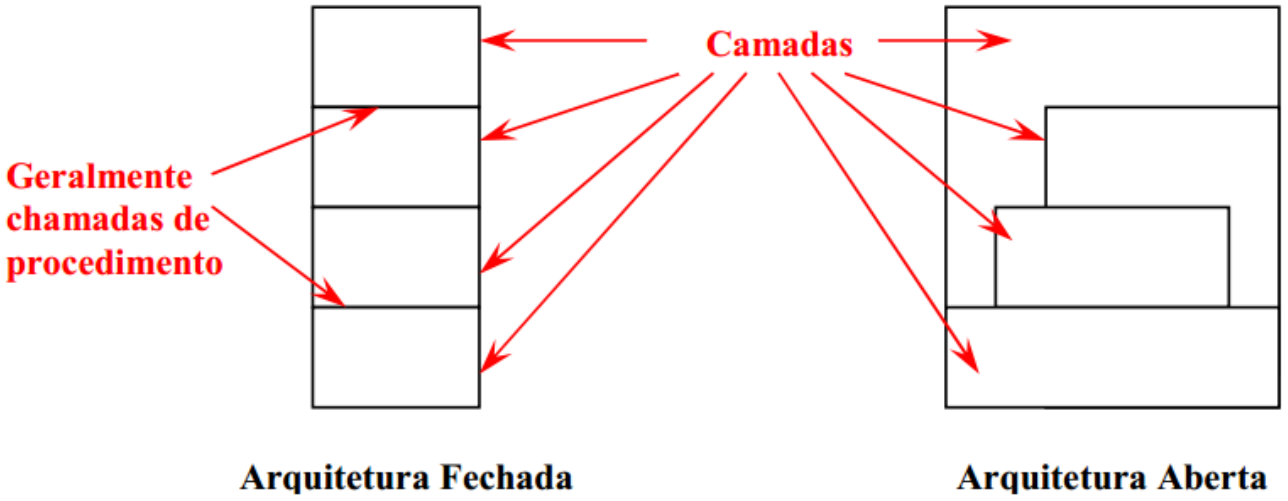
- Dutos e Filtros: considera a existência de uma rede pela qual dados fluem de uma extremidade (origem) até a outra (destino). O fluxo de dados se dá através dos dutos e os dados sofrem transformações nos filtros.
- A canalização de programas no sistema operacional Unix, o modelo clássico de compiladores e sistemas de folha de pagamento são exemplos de uso do estilo dutos e filtros.

**Devido a seu caráter transformacional, este estilo não cai bem para tratar aplicações interativas.**



# Estilos arquitetônicos

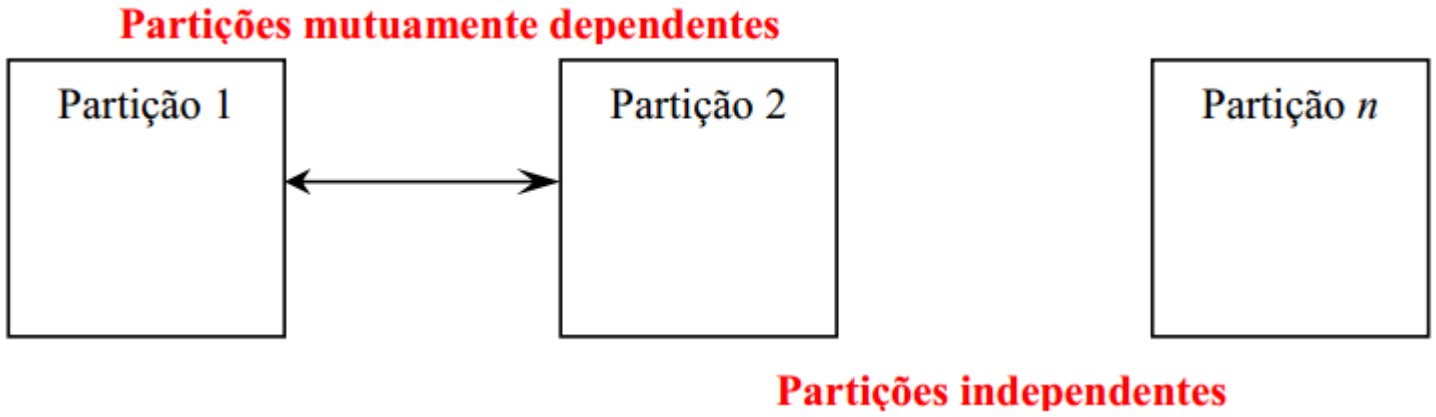
- **Camadas:** um sistema é organizado hierarquicamente, como um conjunto ordenado de camadas, cada uma delas construída em função das camadas abaixo e fornecendo serviços para as camadas acima. O conhecimento é unidirecional, uma camada conhece as camadas abaixo, mas não tem qualquer conhecimento sobre as camadas acima.
- As arquiteturas em camadas podem ser fechadas, quando uma camada é construída apenas em função da camada imediatamente inferior, ou abertas, quando uma camada pode usar recursos de quaisquer camadas inferiores.




**Arquiteturas de redes, tal como a arquitetura de Internet TCP/IP, São bons exemplos de sistemas em camadas.**

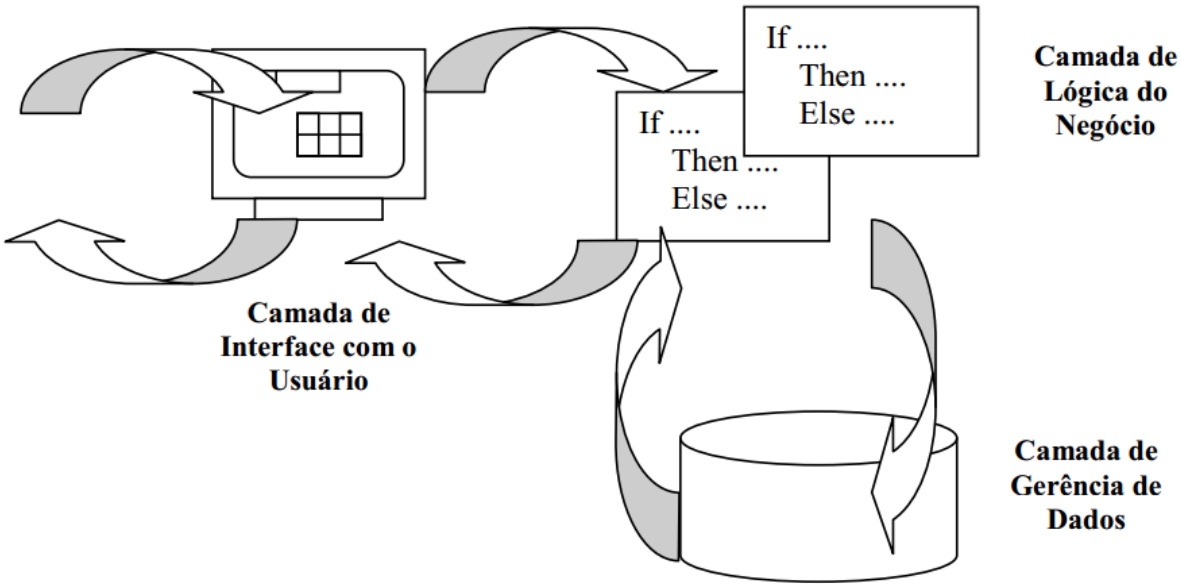
# Estilos arquitetônicos

- **Partições:** dividem um sistema verticalmente em subsistemas fracamente acoplados, cada um fornecendo, normalmente, um tipo de serviço. Este é o caso, por exemplo, de **sistemas operacionais,** que possuem partições para **controle de arquivos, gerência de memória, controle de processo** etc. As partições podem ter algum conhecimento acerca das outras, mas esse conhecimento não é profundo e evita maiores dependências de projeto. Ao contrário das camadas, que variam em seu nível de abstração, as partições simplesmente dividem um sistema em partes, todas tendo um nível de abstração semelhante



# Padrões Arquitetônicos para Projetos de Sistemas de Informação

- Quando trabalhamos com sistemas de informação o estilo arquitetônico mais aderente é o de **Camadas**.
- Podemos visualizar um Projeto de Sistema de Informação conforme o esquema ao lado proposto por Fowler.
- Nesse padrão o Projeto é dividido em 3 camadas:
  - Camada de Interface com o usuário
  - Camada de Lógica de Negócio
  - Camada de Gerência de Dados



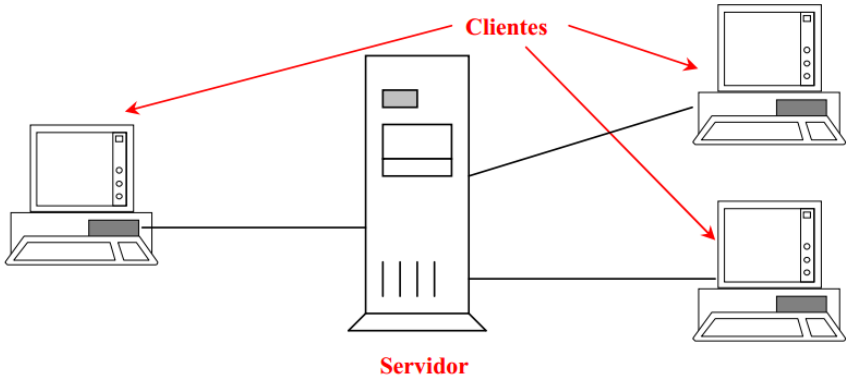

- **Camada de Apresentação ou de Interface com o Usuário**: sua função é tratar a interação entre o usuário e o sistema. As responsabilidades principais dessa camada são exibir informações para os usuários e interpretar comandos do usuário em ações da lógica de negócio e da persistência de dados.
- **Camada de Lógica de Negócio**: contém as funcionalidades que apoiam os processos de negócio. Conceitos do domínio, regras de negócio, processamentos e cálculos são encontrados nesta camada.
- **Camada de Persistência ou de Gerência de Dados**: provê acesso a dados corporativos. É sua responsabilidade gerenciar requisições concorrentes de acesso às bases de dados, assim como a sincronização de elementos de dados distribuídos.

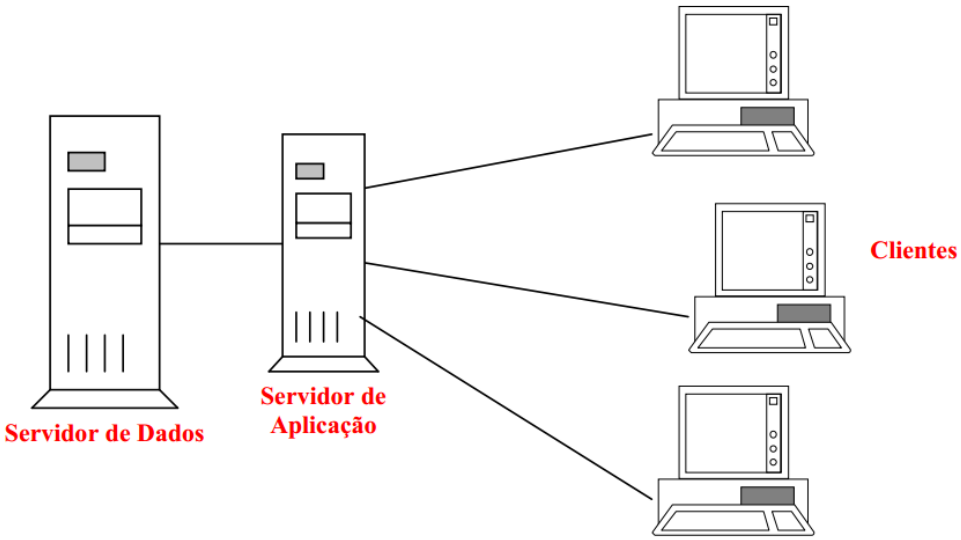
# Projeto de Sistemas de Informação Distribuídos

- Antigamente, ainda no paradigma estruturado, costumava-se dividir os sistemas de informação distribuídos em 2 camadas físicas. A camada cliente e a camada servidor. Nesse contexto a interface com o usuário ficava no lado cliente e os dados no lado servidor. Já a lógica de negócio ficava ora no lado do cliente (que comumente tinha baixo processamento, o que gerava problemas) ora do lado do servidor (através do uso de *stored procedures*, o que também gerava muitos problemas).
- Ao longo do tempo a arquitetura cliente-servidor tem evoluído, mas a questão de onde colocar a lógica de negócio sempre deve ser analisada. Pode-se colocar tudo no cliente (cliente pesado), tudo no servidor (servidor pesado) ou dividi-la. Colocar tudo no servidor facilita a manutenibilidade mas pode prejudicar a interatividade do usuário (interfaces mais ricas decorrentes de um cliente um pouco mais “gordo” - uso de AJAX, por exemplo)
- A separação de hardware é um a característica marcante da arquitetura cliente-servidor.
- Em relação à arquitetura de software, o termo cliente-servidor pode ser usado para descrever diferentes componentes de software se comunicando uns com os outros, ainda que rodando em uma mesma máquina.
- Considerando o mapeamento de camadas lógicas em camadas físicas, um uso bastante comum de arquiteturas cliente-servidor consiste em explorar o poder dos computadores pessoais para gerenciar interfaces gráficas com o usuário, mantendo os serviços e dados do negócio em servidores.

# Projeto de Sistemas de Informação Distribuídos

- No que se refere ao hardware, a arquitetura cliente-servidor utiliza mais comumente os modelos abaixo:



# Considerações sobre Arquiteturas Físicas

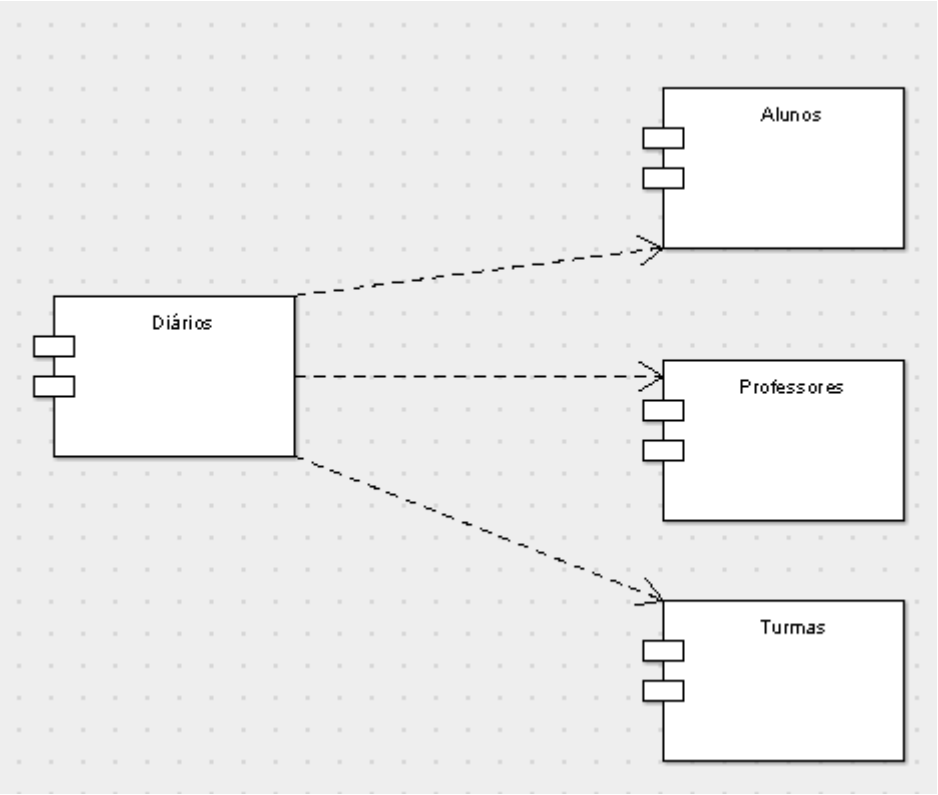
- Para sistemas simples, arquiteturas físicas não são tão importantes.
- Os diagramas de implementação da UML tratam essa questão para sistemas mais complexos. Os diagramas de componentes e de implantação são os dois tipos de diagramas de implementação.

# Diagrama de Componentes

- Mostra componentes de software e suas dependências.

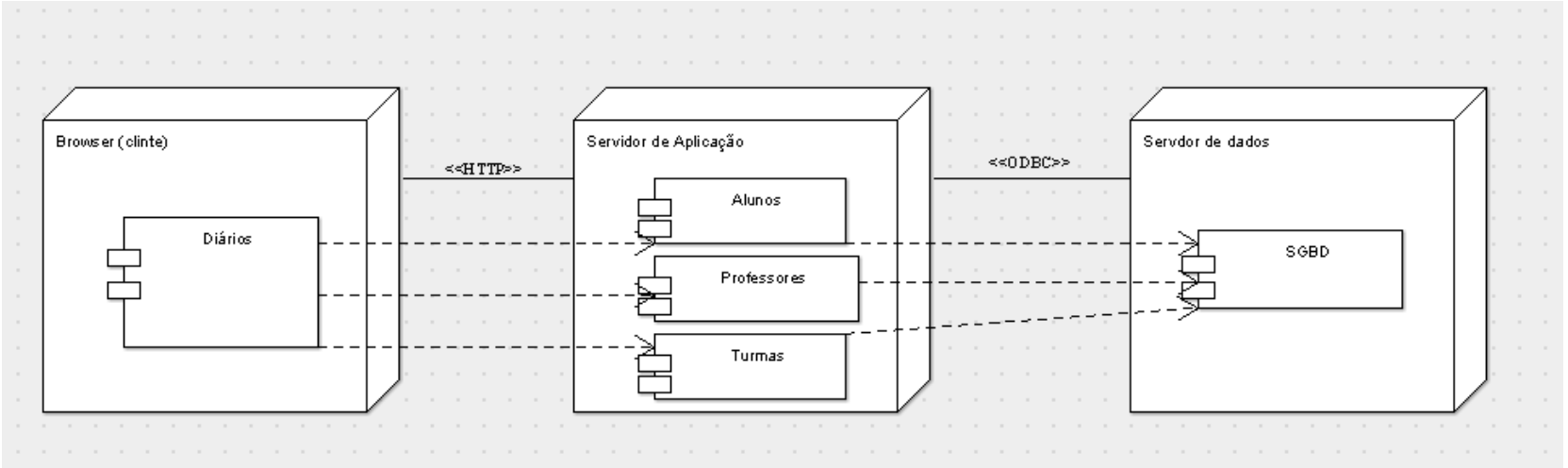
Componente

**Os componentes pode ser executáveis, documentos, tabelas, dlls, etc**

# Diagrama de Implantação

- Apresenta a topologia física do sistema e opcionalmente os componentes utilizados nessa topologia.



# Dúvidas?