

# Padrões de Projeto e Microserviços

## Sistemas Distribuídos e Mobile

Prof. Me. Gustavo Torres Custódio  
gustavo.custodio@anhembi.br



Padrões de Projeto e Micro-  
serviços

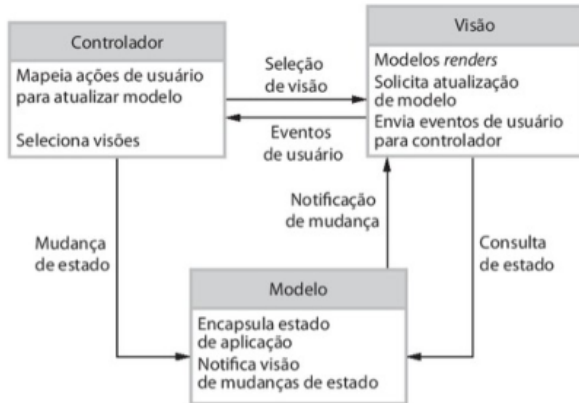
MVC

# MVC

- O padrão de projeto MVC divide o código em 3 partes:
  - Model;
  - View;
  - Controller.

**Figura 6.2**

A organização do MVC



## MVC - Model

- Camada para armazenamento de dados.
  - É a camada que tende a se comunicar com o banco de dados da aplicação.
  - As regras de negócio do mundo real são inseridas nessa camada.

## MVC - View

- É a camada com a Interface de Usuário contendo os componentes da tela.
- Proporciona a visualização dos dados contidos na camada *Model*.

## MVC - Controller

- Faz a conexão entre o **Model** e o **View**.
- Contém toda a lógica da aplicação.
- Atualiza o Model conforme o usuário fornece respostas.



Padrões de Projeto e Microservices

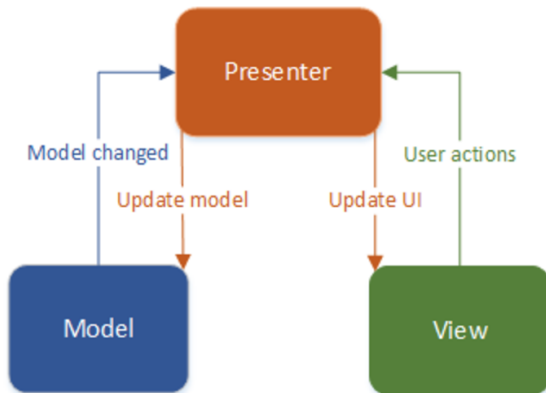
MVP



# MVP

- A arquitetura MVP é derivada da arquitetura MVC.
- Ela conta com três elementos:
  - *Model*;
  - *View*;
  - *Presenter*;

## MVP



# MVP - Model

- Camada para o armazenamento de dados.
  - É responsável por lidar com:
    - a lógica do domínio (regras de negócios do mundo real)
    - a comunicação com o banco de dados e as camadas de rede.

## MVP - View

- A visão do usuário.
- Mostra como os dados são mostrados.
- Detalhes da *User Interface* (UI).
- Direciona comandos do usuário (eventos) para o *Presenter*, dessa forma, ele pode executar esses eventos.

## MVP - Presenter

- O *Presenter* interage tanto com o *Model* quanto com o *View*
- Ele recupera dados da aplicação vindos do modelo e formata para serem utilizados na *View*.
- Gerencia o estado da *View*.

# MVP

- A maior diferença do MVP em relação ao MVC é não permitir a comunicação entre a camada *Model* e a camada *View*.

- Vantages:
  - Facilidade de realizar testes, por conta da camada da lógica de negócio ser independente da *View*.
  - *View* não precisa ser adaptado ao padrão de dados do *Model*.
  - Permite a criação de múltiplos *presenters* diferentes, permitindo mostrar os dados da aplicação de diferentes formas.



Padrões de Projeto e Micro-  
services

# Microserviços



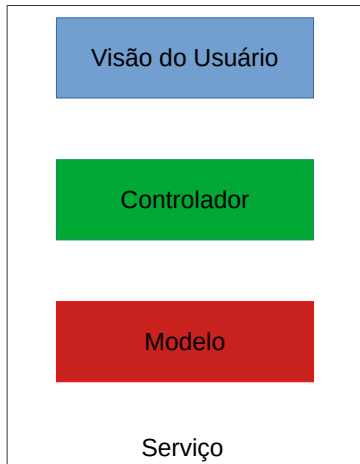
# Microserviços

- A arquitetura de microserviços é uma arquitetura para a criação de aplicações.
- Ela é frequentemente utilizada dentro de grandes organizações por facilitar a criação de novas tecnologias.
- Alternativa à arquitetura monolítica.

# Arquitetura Monolítica

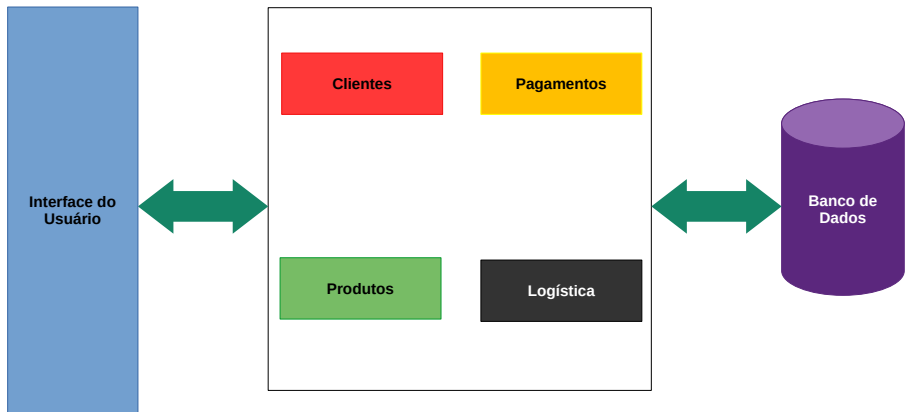
- Aplicações onde a interface do usuário e o ambiente de acesso de dados pertencem ao mesmo programa.
  - Monolítico: unido, inseparável, homogêneo.
- Podem utilizar a arquitetura *MVC (Model View Controller)*.

# Arquitetura Monolítica



## Arquitetura Monolítica

- Suponha uma aplicação monolítica que precise gerenciar a cadeia de produção de uma empresa:



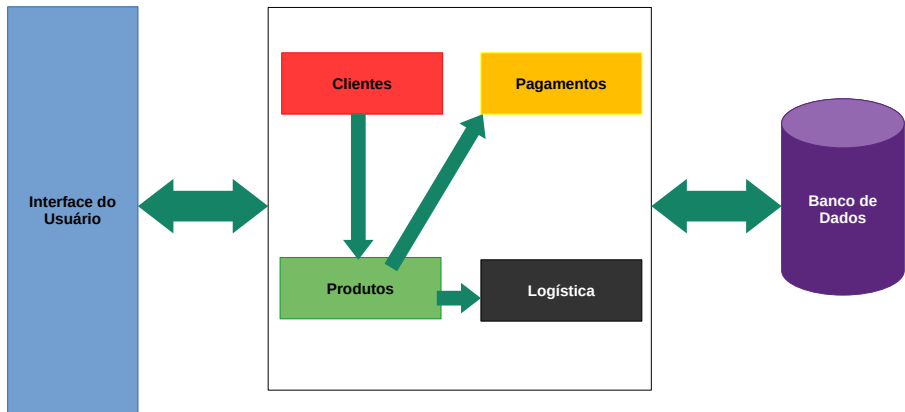
# Arquitetura Monolítica

- Elementos **muito acoplados**.
  - Manutenção e desenvolvimento de novas aplicações se tornam difíceis.
- Como lidar com esses problemas?
- A arquitetura de microsserviços é uma alternativa.

# Microserviços

- Uma aplicação é dividida em pequenos serviços **independentes** que se comunicam por meio de APIs.
- Arquiteturas de Microserviços contribuem para:
  - Escalabilidade;
  - Velocidade desenvolvimento de novas aplicações e novos recursos.
- Também há menor preocupação com possíveis efeitos colaterais em outras funções da aplicação.

# Microsserviços



## Microsserviços × Arquitetura Monolítica

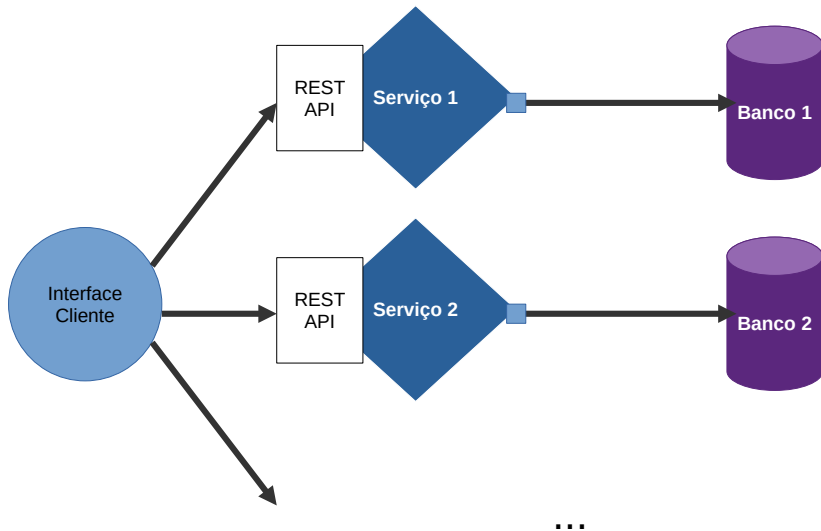
- Arquitetura monolítica:
  - Todos os processos são altamente acoplados e executam como um único serviço.
  - Se um processo do aplicativo apresentar maior demanda, toda a arquitetura terá que ser escalada.
  - Conforme mais recursos são adicionados, se torna cada vez mais difícil adicionar novos recursos na arquitetura monolítica.
  - A falha em um único processo pode comprometer toda a aplicação.



# Microserviços × Arquitetura Monolítica

- Arquitetura de microserviços:
  - Um aplicativo é criado como um conjunto de serviços independentes.
  - Cada serviço realiza uma única função.
  - Como são executados de forma independente, cada serviço pode ser alterado e escalado conforme demandas sem afetar outros serviços.

# Características de Microsserviços



# Características de Microsserviços

- Autônomos:
  - Um serviço não afeta o comportamento de outro.
- Os serviços não precisam compartilhar código com nenhum outro serviço.
- Todas as comunicações ocorrem por meio de APIs bem definidas.

# Características de Microsserviços

- Especializados:
  - Cada serviço é dedicado a um problema específico.
- Se um serviço começa a crescer muito, ou o problema se torna muito complexo...
  - É possível quebrar o serviço em serviços menores.

# Características de Microsserviços

- Agilidade:
  - Os microsserviços promovem a criação de equipes pequenas que cuidam de serviços específicos.
  - Acelera o ciclo de desenvolvimento.
  - Permite que cada equipe seja especializada em seu serviço.
    - Faça apenas uma coisa e faça ela bem.

# Características de Microsserviços

- Escalabilidade flexível:
  - Os microsserviços permitem que cada serviço seja escalado de forma independente para atender à demanda do recurso de aplicativo oferecido por esse serviço.
  - Permite que as equipes dimensionem corretamente a necessidade da infraestrutura.

# Características de Microsserviços

- Liberdade tecnológica:
  - As arquiteturas de microsserviços não seguem uma abordagem generalista.
  - As equipes são livres para escolher a melhor ferramenta para resolver problemas específicos.
  - O resultado é que as equipes que criam microsserviços podem optar pela melhor.

# Características de Microsserviços

- Código reutilizável:
  - A divisão do software em módulos pequenos e bem definidos permite que as equipes usem funções para várias finalidades.
  - Um serviço criado para uma determinada função pode ser usado como componente básico para outro recurso.
  - Isso permite que os aplicativos sejam reutilizados, pois os desenvolvedores podem criar recursos sem precisar desenvolver código.



# Características de Microsserviços

- Resiliência:
  - A independência do serviço aumenta a resistência a falhas do aplicativo.
  - Em uma arquitetura monolítica, a falha de um único componente poderá causar a falha de todo o aplicativo.
  - Com os microsserviços, os aplicativos lidam com a falha total do serviço degradando a funcionalidade, sem interromper todo o aplicativo

## Vantagens de Microserviços

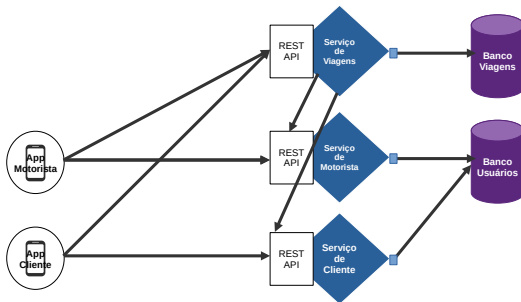
- Adoção de novas tecnologias com maior facilidade.
- Alta disponibilidade.
- Escalabilidade.
- Facilidades no *Deployment*.
- Melhor organização do trabalho.

## Desvantagens de Microsserviços

- Complexidades de utilizar Sistemas Distribuídos.
- Possibilidade maior de ocorrer erros de comunicação entre serviços.
- Dificuldade de manter muitos serviços ao mesmo tempo.

## Exercício

- A figura abaixo mostra uma arquitetura de microsserviços fictícia para um aplicativo de carona:





- Como alterar essa arquitetura para adicionar uma opção de acesso pelo navegador?

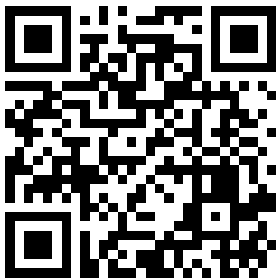
## Exercício

- Reúnam-se em grupos e modelem uma solução de microsserviços para atender soluções abaixo:
  - NETFLIX
  - Spotify
  - Banco do Brasil

## Referências

-  AWS (2022).  
O que são microsserviços.  
Último acesso: 11 de abril de 2022.
-  Richardson, C. (2017).  
What are microservices?  
Último acesso: 11 de abril de 2022.

## Conteúdo



<https://gustavotcustodio.github.io/sdmobile.html>

Obrigado

[gustavo.custodio@anhembi.br](mailto:gustavo.custodio@anhembi.br)