Análise e Projeto de Software

Arquitetura de Software

Matheus Paixão

Arquitetura de Software

Arquitetura de Software

A arquitetura representa como o software deve ser organizado

Indica como os módulos de alto nível se relacionam e se comunicam

Serve como ponte entre os requisitos e suas implementações

Arquitetura de Software

O objetivo da arquitetura é fomentar a discussão entre stakeholders

Arquitetura de um software não é somente um artefato

> “Conjunto de decisões sobre as coisas mais importantes do software”

Decisões importantes, uma vez tomadas, são difíceis de ser alteradas

O projeto de arquitetura de um software é um processo criativo

Padrões Arquiteturais

Padrões Arquiteturais

Arquiteturas utilizadas com sucesso em outros softwares

- Arquitetos reusam um padrão para projetar a arquitetura de um software

Diversos padrões foram propostos ao longo dos anos

- De acordo com a evolução das tecnologias e das demandas de usuários

A discussão sobre padrões arquiteturais é uma discussão histórica

- Cada padrão deve ser entendido sob seu contexto

Também chamados de **estilos arquiteturais**

Arquitetura em Camadas

Arquitetura em Camadas

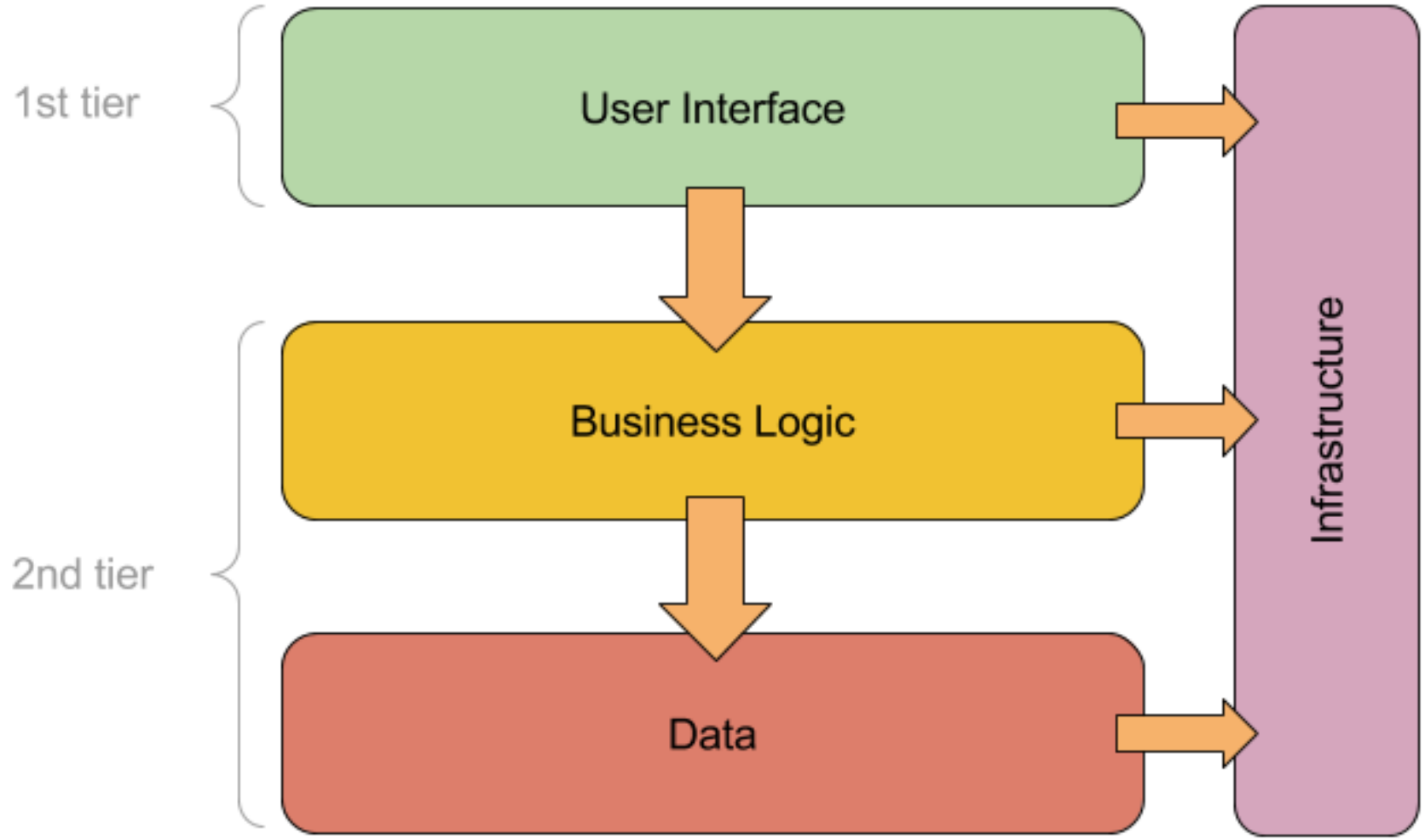
Proposta nos anos 60 e 70 com o desenvolvimento de softwares robustos

Recursos são organizados em grandes módulos, chamados de camadas

As camadas são dispostas de forma hierárquica

- Uma camada usa somente recursos da camada imediatamente inferior

Arquitetura em Camadas



Arquitetura em Camadas

Arquitetura em camadas particiona a complexidade do software

O controle de interfaces ajuda na manutenção e evolução do software

- Torna-se mais fácil trocar uma camada por outra, mantendo a mesma interface
- Além disso, é mais fácil o reúso de camadas em sistemas diferentes

Camadas de drivers são reusadas por todo e qualquer software atual

Arquitetura Cliente-Servidor

Arquitetura Cliente-Servidor

Teve origem com o desenvolvimento de sistemas distribuídos

O software é dividido em dois componentes executando independente

O cliente é a interface com o usuário, exibe dados e captura interações

Servidor recebe comandos, controlando a lógica de negócio e manipulando os dados

Arquitetura Cliente-Servidor

Usuários acessam o software simultaneamente com clientes independentes

Primeira aplicação com “terminais burros”

- Dispositivos com pouco poder de processamento acessando computadores robustos

Arquitetura Cliente-Servidor

Arquitetura Cliente-Servidor

Usuários acessam o software simultaneamente com clientes independentes

Primeira aplicação com “terminais burros”

- Dispositivos com pouco poder de processamento acessando computadores robustos

Cliente-Servidor evoluiu com a internet, servindo de base para o desenvolvimento de aplicações web

Arquitetura Cliente-Servidor

Em softwares web, o browser executa o software cliente

Servidor é executado em máquinas remotas

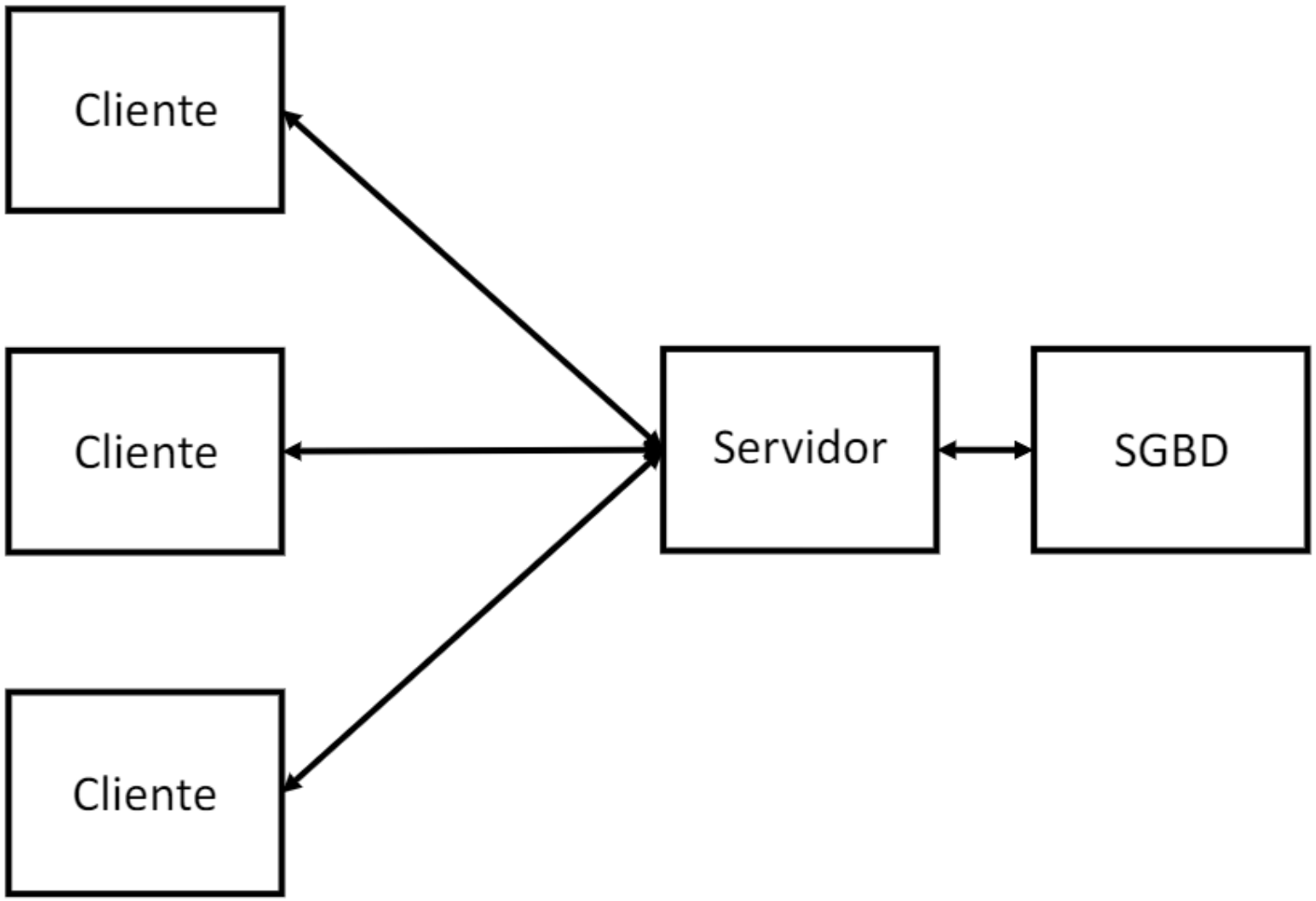
Cliente e servidor se comunicam com protocolos da internet

- HTTP, SMTP, TCP/IP etc

O banco de dados é gerenciado por um SGBD

- Executado na mesma máquina do servidor ou na mesma rede local

Arquitetura Cliente-Servidor



Arquitetura Model-View-Controller (MVC)

Arquitetura Model-View-Controller (MVC)

Proposto nos anos 80 para o desenvolvimento da linguagem Smalltalk

- Primeira linguagem a dar suporte mais robusto à interfaces gráficas

Com mais recursos, o código de interface se tornou mais complexo

Necessidade de uma divisão entre código de interface e de negócio

Arquitetura Model-View-Controller (MVC)

- Com evolução do cliente/servidor, MVC foi adaptado para a web
  - MVC “atual” é uma adaptação do MVC “clássico”
    - Dada a popularidade de sistemas web, pode-se dizer que MVC web é de fato o “padrão”
- No MVC web, a View é a página web sendo executada no browser
  - Front-end
- A camada de Controller é executada no servidor
  - Lida com todas as questões relacionadas à comunicação web
  - Controller recebe as requisições da View repassando-as para a Model

Arquitetura Model-View-Controller (MVC)

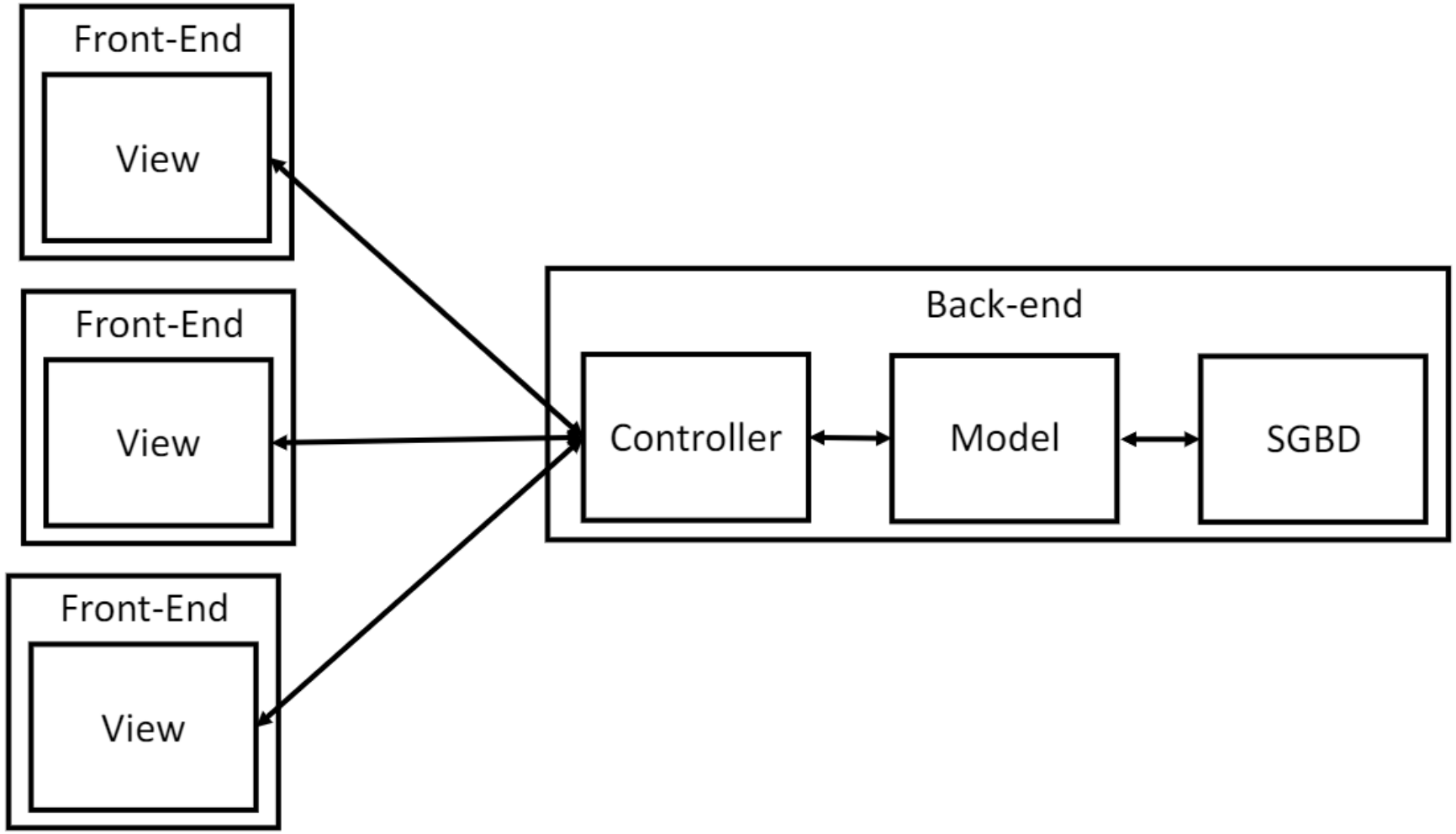
A Model é uma camada de software que faz parte do servidor

A Model manipula os dados do software seguindo regras de negócio

A Model se comunica com o SGBD e manipula os dados do banco

Na web, a combinação de Controller, Model e SGBD é o back-end

Arquitetura Model-View-Controller (MVC)



Microsserviços

Microsserviços

Desenvolvimento de sistemas de software é dividido em equipes
- Diferentes desenvolvedores desenvolvem diferentes componentes do sistema

Apesar do desenvolvimento independente, a execução é monolítica
- Módulos são integrados em um único software, compartilhando recursos

Defeitos em módulos podem se propagar para os demais

Com problemas de performance, aumenta-se todos os recursos

Entre 1990 e 2010, softwares back-end na web adotavam estratégia monolítica

Microsserviços

Martin Fowler

Participante do manifesto ágil

Popularizou práticas de desenvolvimento

- Refatoração de código
- Integração Contínua

Microsserviços

Em 2014, Martin Fowler publicou o artigo intitulado “Microservices”

Padrão arquitetural que decompõe um software em módulos independentes não só em desenvolvimento, mas também em execução

Software é uma composição de serviços

Serviços executados de forma independente e se comunicam pela web

Microsserviços

Cada microsserviço implementa uma funcionalidade completa

- Regras de negócio, alteração do estado do sistema e manipulação de dados
- “Micro” não indica tamanho, mas que o serviço implementa uma funcionalidade

Microsserviços traz vantagens para o desenvolvimento de sistemas

- Times são responsáveis pelo serviço
  - Implementação, implantação e monitoramento
- Além de flexibilidade e produtividade, evita-se problemas de integração

Microsserviços

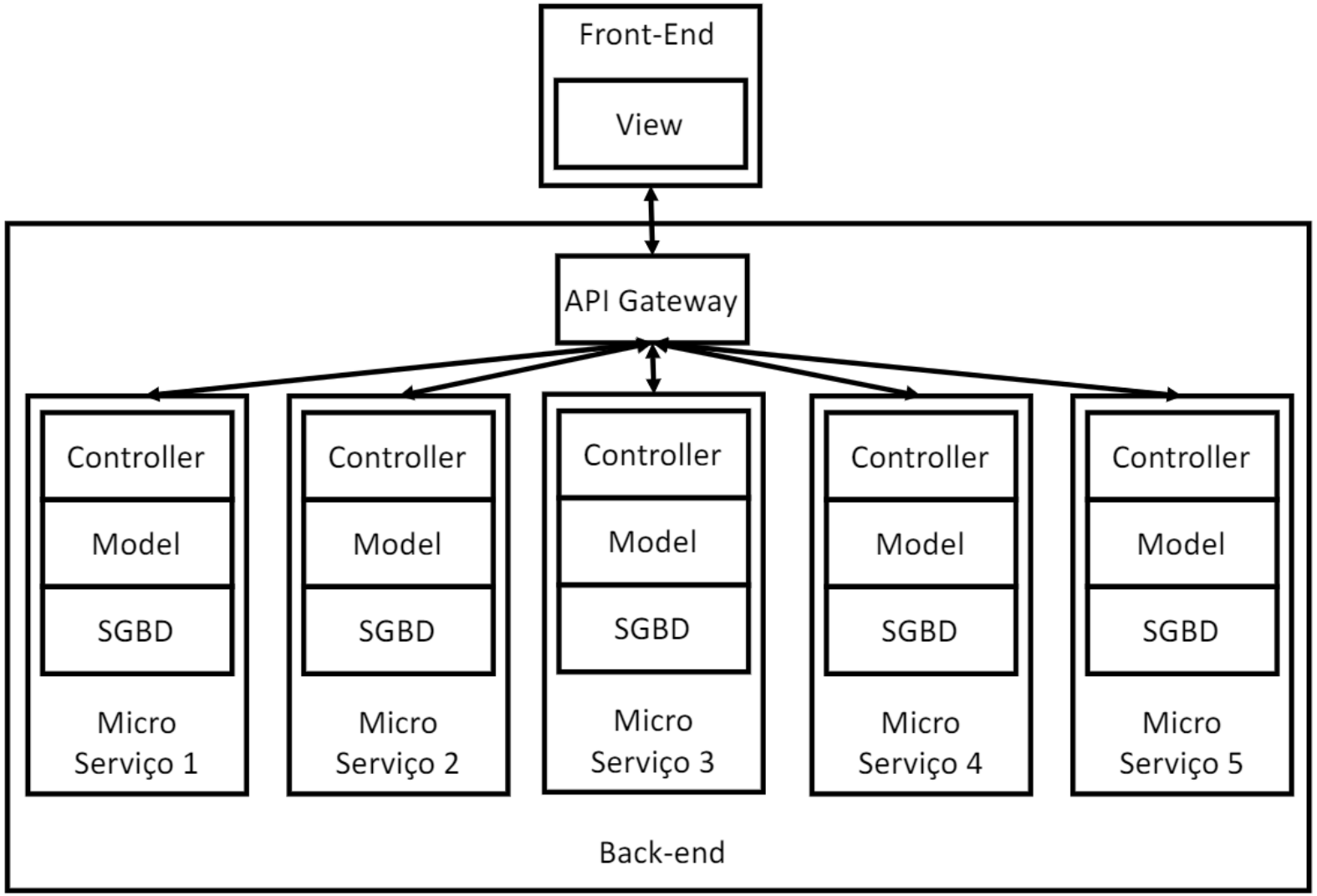
Em um sistema web, os microsserviços formam o back-end

Um microsserviço provê uma funcionalidade, gerenciando seus dados
- Independência de dados é uma das dificuldades de Microsserviços “puros”

Software com vários requisitos tem muitos microsserviços
- Impraticável que front-end se comunique diretamente com os microsserviços

Para contornar esse problema, utiliza-se a estratégia de API gateway
- Recebe as requisições do front-end e redireciona para cada microsserviço

Microsserviços



Lista de Exercícios 4

Exercícios do Capítulo 7 do livro Engenharia de Software Moderna

https://engsoftmoderna.info/cap7.html

Entrega até 03/11 às 23:59

Calendário Projeto Etapa de Arquitetura

Calendário do Projeto – Etapa de Arquitetura

28/10: Aula Arquitetura de Software

30/10: Aula Arquitetura Web

04/11: Primeiras Discussões Arquitetura (em sala)

06/11 : Alinhamentos Entrega Arquitetura

11/11 e 13/11: Apresentações e Entrega da Arquitetura

Referências

Referências

Capítulo **Arquitetura** do livro *Engenharia de Software Moderna,* Marco Túlio Valente

Capítulo **Projeto de Arquitetura** do livro *Engenharia de Software,* Ian Sommerville

Leituras Adicionais

Leituras Adicionais

Shaw, M., & Garlan, D. (1996). Software architecture: perspectives on an emerging discipline. Prentice-Hall, Inc.

Rozanski, N., & Woods, E. (2011). Software systems architecture: working with stakeholders using viewpoints and perspectives. Addison-Wesley.