



UNIVERSIDADE
FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

Back-end, Front-end e API REST

QXD0020 - Desenvolvimento de Software para Web

Prof. Bruno Góis Mateus (brunomateus@ufc.br)

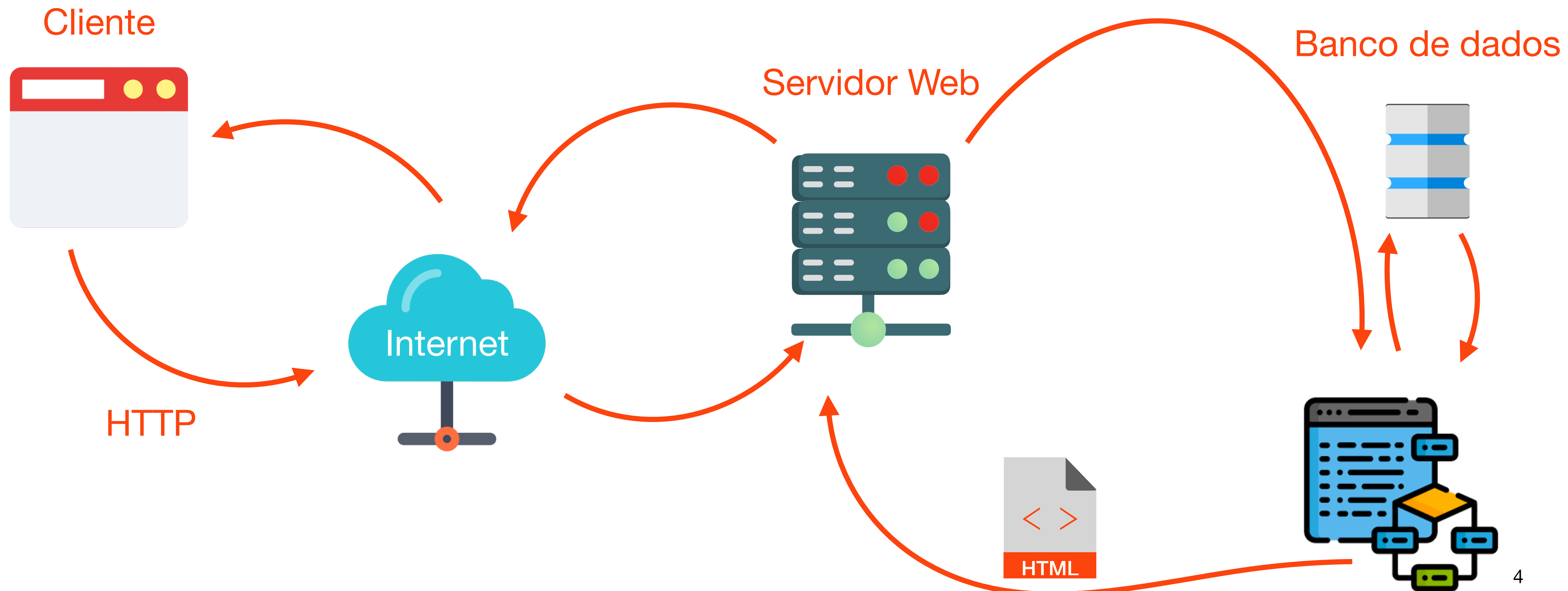
Agenda

- Introdução
- Arquitetura MVC - Model View Controller
- API REST
- Back-end vs Front-end

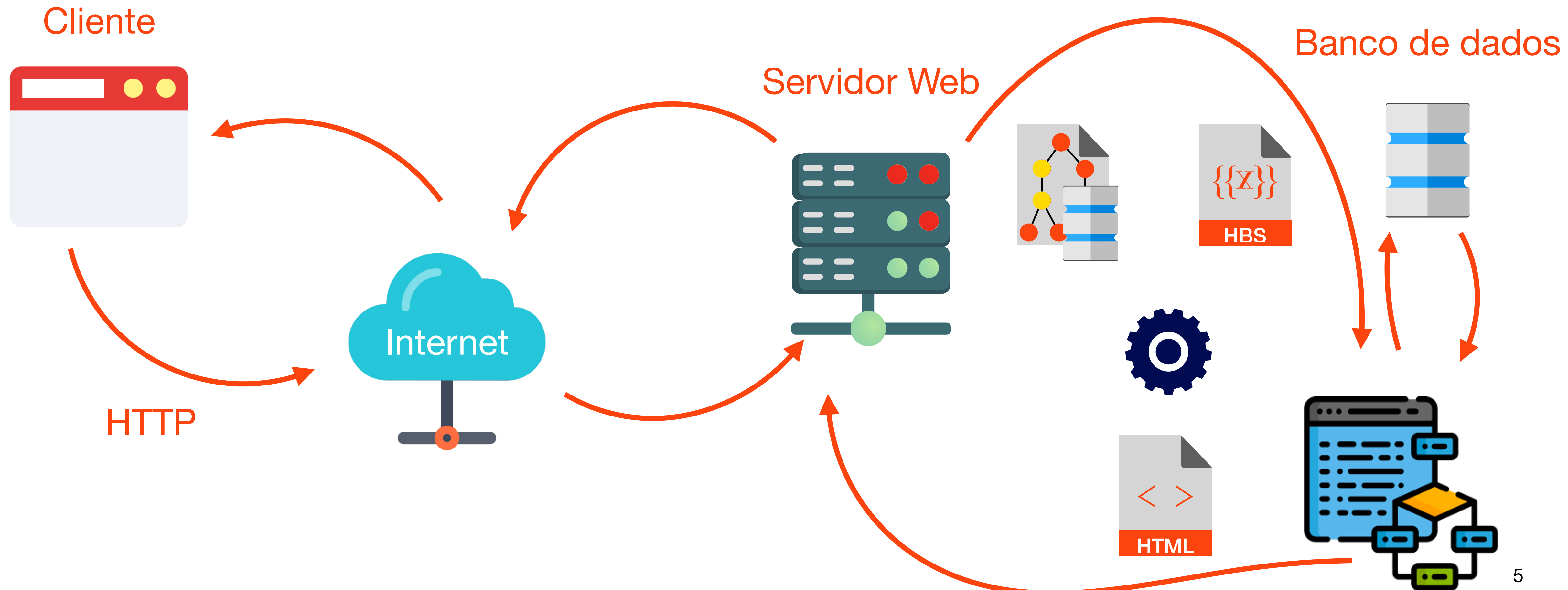
Introdução



Introdução



Introdução



Arquitetura
MVC

ex

Arquitetura MVC

Model - View - Controller

- Padrão arquitetural que se tornou popular em meados de 1970
- Separa a representação da informação da visualização da mesma
- Divide o sistema em três partes interconectadas
 - Model
 - View
 - Controller

Arquitetura MVC

Controller

Controller

- Realizam a ligação entre o usuário e o sistema
- Devem aguarda por requisições HTTP
 - Aceita entradas e converte para comandos para view ou model
 - Delega as regras de negócio para modelos e serviços
- Retorna com uma resposta significativa

Arquitetura MVC

View

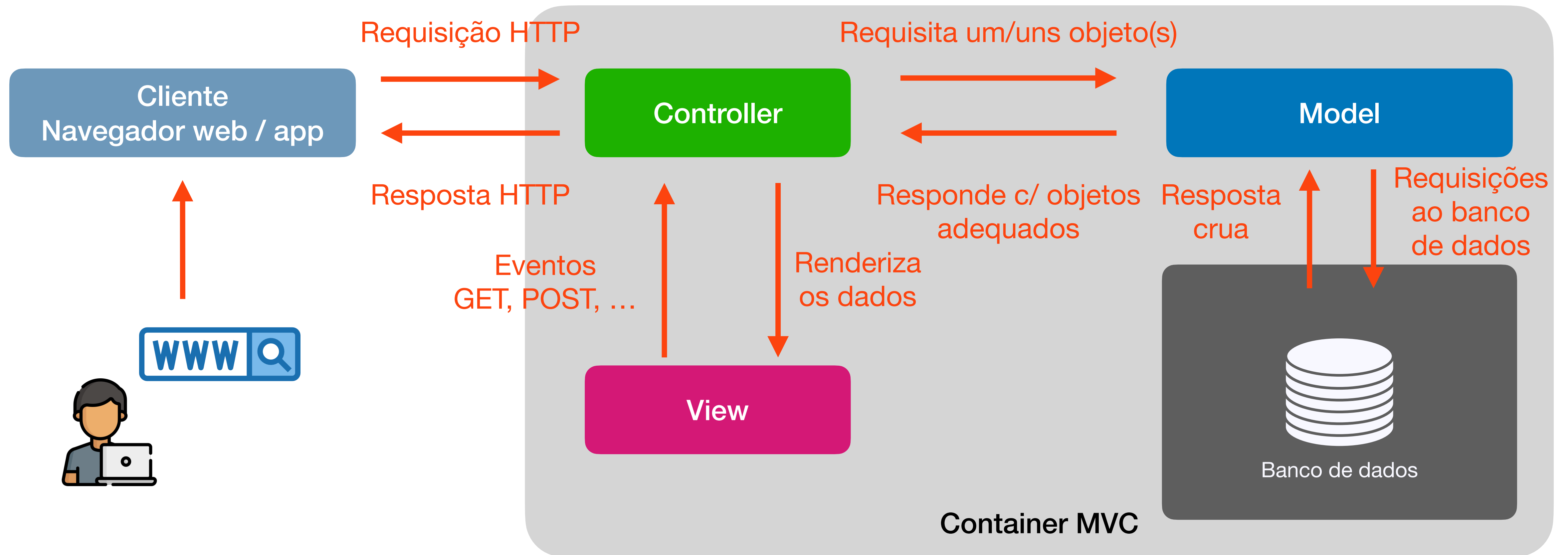
- Representação visual da nossa aplicação (GUI - Graphical User Interface)
- Mostram os dados ao usuário em forma fácil de entender baseado nas suas ações
 - Camada de interação com o usuário
 - Um mesmo conjunto de dados pode ser visualizado de n maneiras diferentes
 - Ex: Gráfico de barras , Diagrama de pizza
- Deve refletir mudanças ocorridas nos modelos

Arquitetura MVC

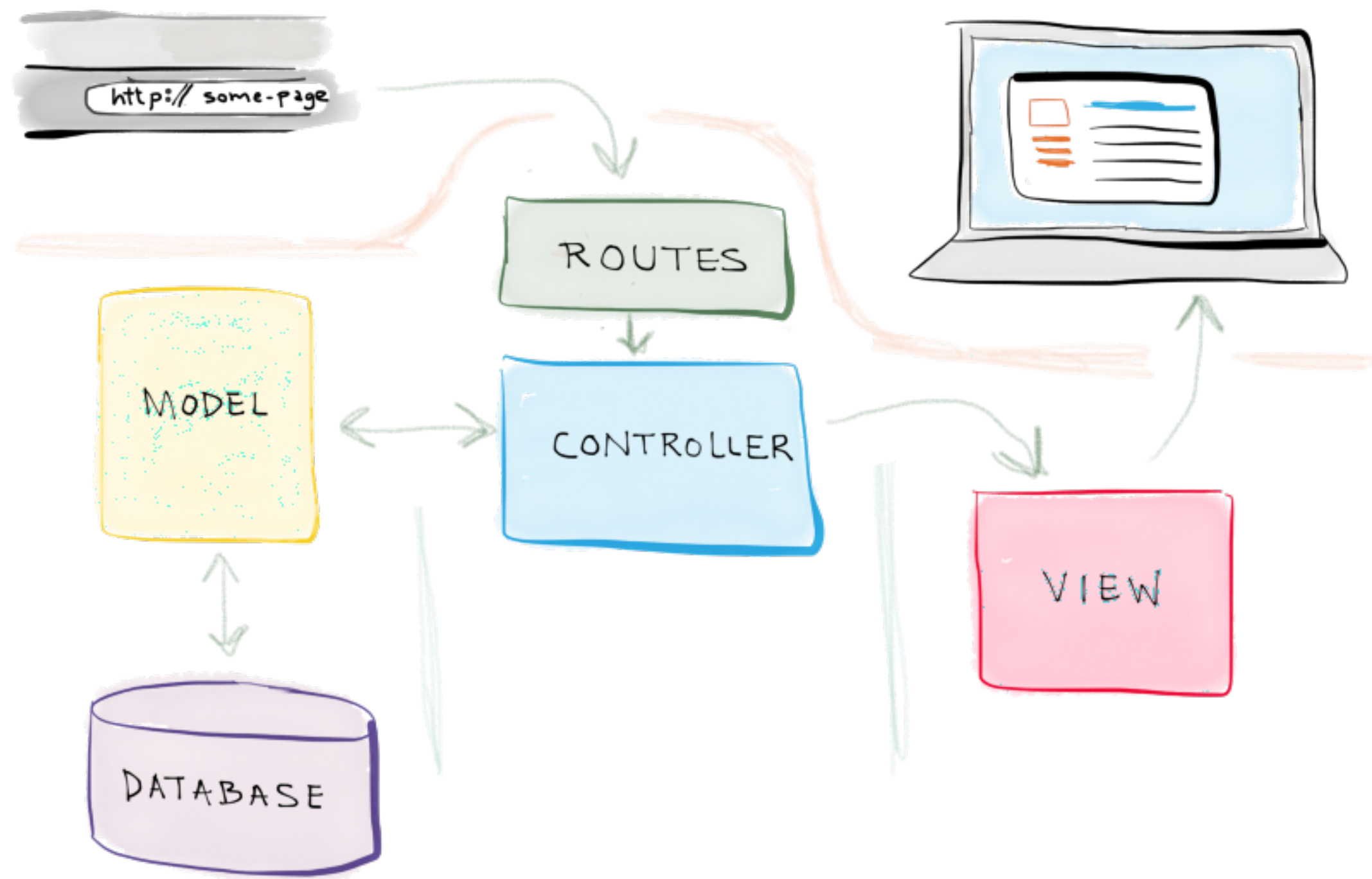
Model

- Modelo representam o conhecimento do domínio da aplicação
- Gerencia os dados, a lógica e as regras da aplicação
- Independente da interface com o usuário
- Encapsulam os dados do banco de dados
 - Tabelas

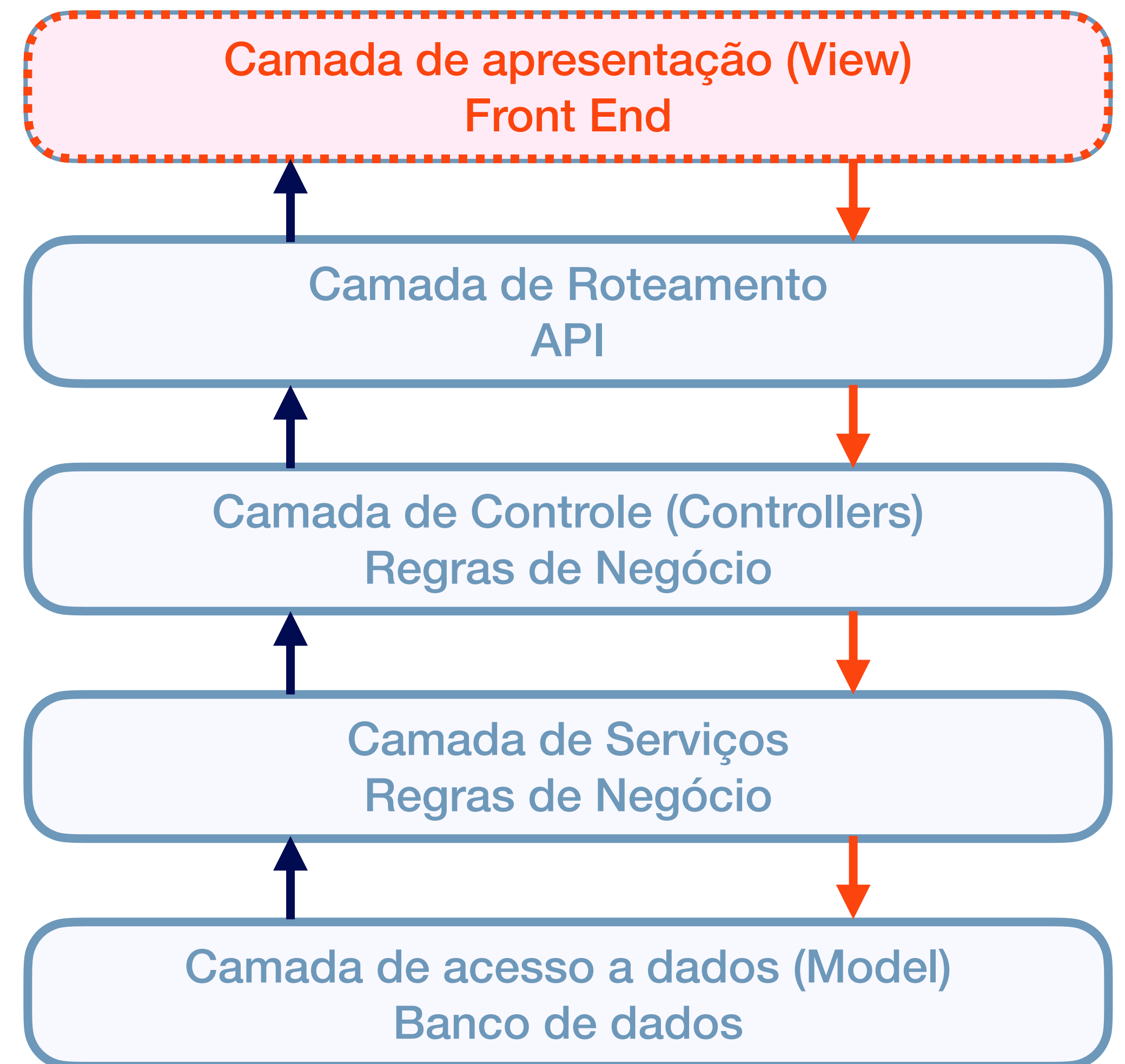
Arquitetura MVC



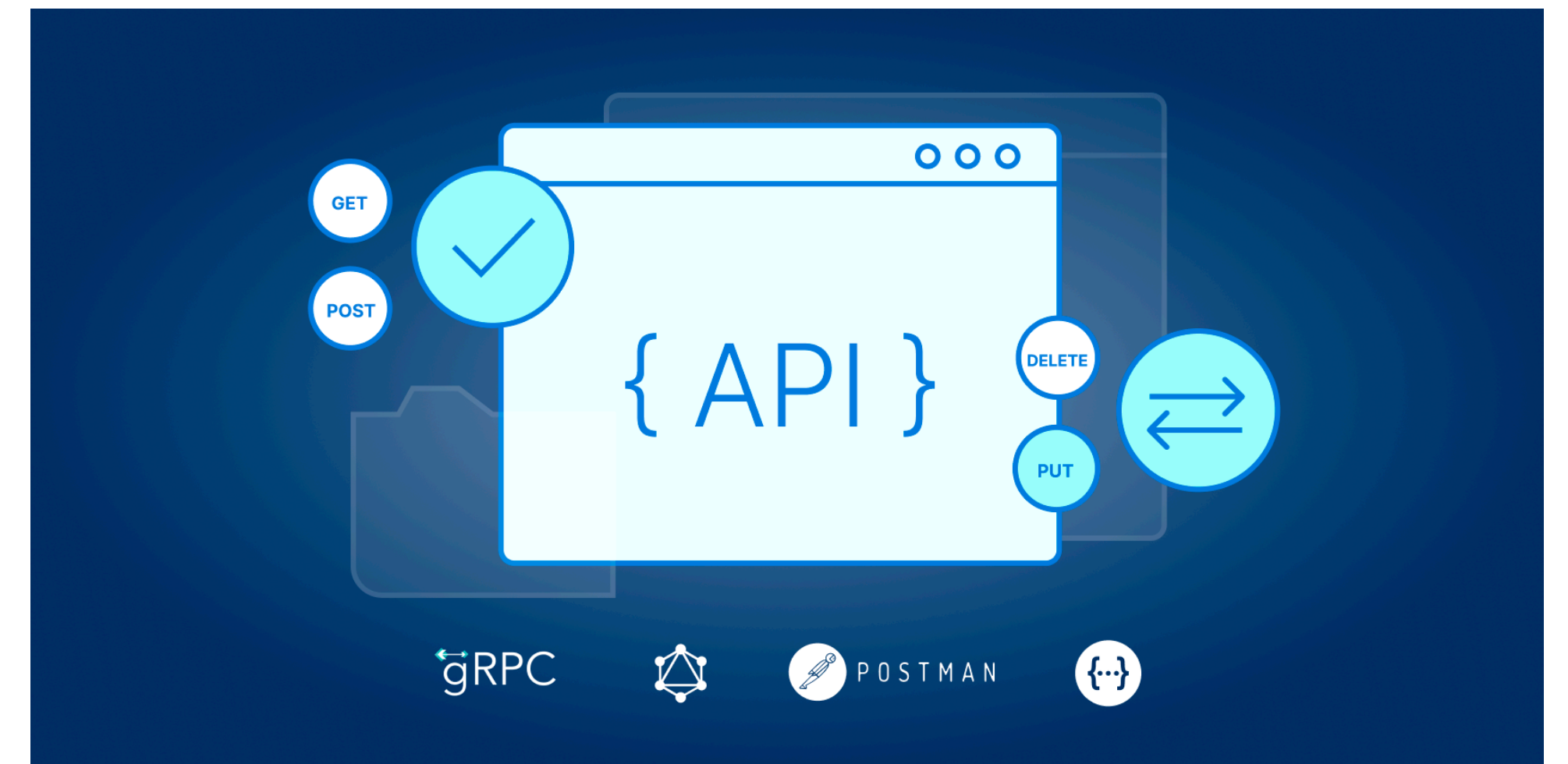
Arquitetura MVC



Créditos: Real Python



API REST



API REST

Introdução

API - Application Programming Interface

- Apresenta funções e regras que permitem a interação e comunicação entre diferentes aplicativos
- Facilitam a integração de aplicativos
 - Permitindo que os desenvolvedores criem produtos digitais poderosos
- Faz a mediação entre aplicativos por meio de solicitações e respostas
 - Vários protocolos e arquiteturas podem ser utilizados
 - XML-RPC, JSON-RPC, SOAP, REST

API REST

Introdução

REST - REpresentational State Transfer

- Estilo arquitetural para sistemas distribuídos de hipermídia
- A manipulação dos recursos disponibilizados pela API é realizada através do protocolo HTTP
 - O formato JSON é o mais para representar os estados
- Conjunto de regras e boas práticas para o desenvolvimento dessas APIs
 - 6 princípios

API REST

Princípios REST

Interface Uniforme (*Uniform Interface*)

- Os **recursos** devem ser unicamente identificados por meio de **uma URL**
- Os **recursos** não devem ser grandes, mas devem conter toda a informação que o **cliente** possa necessitar

API REST

Princípios REST

Cliente-Servidor

- Trata da separação de responsabilidades
- **Cliente** e **Servidor** podem evoluir de forma independente
- O contrato entre eles se mantém intacto

API REST

Princípios REST

Sem estados (*Stateless*)

- Toda requisição realizada deve conter toda a informação necessária para que ela seja entendida
 - O **servidor** não deve possuir conhecimento sobre requisições feitas previamente
 - O **servidor** não de armazenar informações sobre as requisições

API REST

Princípios REST

Cache

- Sempre que possível o conteúdo deve ser *cacheable*
 - Seja no *cliente* ou no *servidor*
 - As respostas do devem possuir uma label indicando essa condição
 - Diretivas no cabeçalho HTTP
- O objetivo é melhorar a *performance no cliente* e aumentar a *escalabilidade do servidor*

API REST

Princípios REST

Camadas

- Arquitetura em camadas
- Devem ser projetadas de forma que nem o **cliente** nem o **servidor** saibam se eles estão se comunicando com a aplicação fim
 - Ex: Múltiplas camadas de **servidores**

API REST

Princípios REST

Código sobre demanda (Opcional)

- Na maioria das vezes o **servidor** responde com **recursos estáticos**, no entanto, em certos casos o **servidor** deve poder incluir **código executável**

API REST

Anatomia de uma API REST

- A comunicação com uma API REST é feita através de requisições HTTP



API REST

Método HTTP

- Em geral, uma API REST contempla as operações de CRUD
- Nesse ponto, o entendimento dos métodos HTTP é crucial para a construção de uma boa API

Finalidade

HTTP Method

Buscar dados em uma API

GET

Criar recurso em uma API

POST

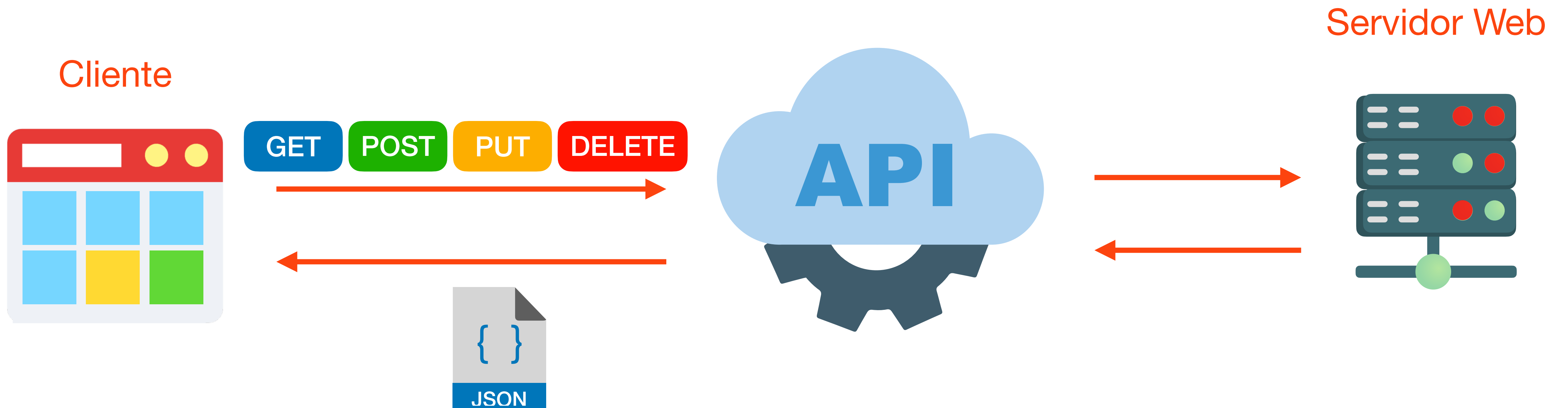
Atualizar um recurso em uma API

PUT PATCH

Excluir um recurso em uma API

DELETE

API REST



API REST

Códigos de status HTTP

Grupo	Código	Quando
1xx - Respostas informativas	Raramente são utilizadas	Raramente são utilizadas
2xx - Envia em caso de sucesso	200 OK	Código mais utilizado. Requisição processada com sucesso
	201 Created	Indica que um novo registro foi criado. Usado em respostas a requisições POST

API REST

Códigos de status HTTP

Grupo	Código	Quando
3xx - Define respostas de redirecionamento	301 Moved Permanently	Informa que o recurso A agora é o recurso B
	304 Not modified	Resposta utilizada em cenários de cache. Informa ao cliente que a resposta não foi modificada. Portanto, o cliente pode usar a mesma versão em cache da resposta
	307 Temporary redirect	Informa sobre o redirecionamento de uma URL para outra, porém com caráter temporário

API REST

Códigos de status HTTP

Grupo	Código	Quando
4xx - Informa erros no lado do cliente	400 Bad Request	Indica que o servidor não consegue entender a requisição, devido a sintaxe ou estrutura inválida
	401 Unauthorized	Informa que existe uma camada de segurança para recurso solicitado, e que você não está utilizando as credenciais corretas nessa requisição
	403 Forbidden	Credenciais reconhecidas, porém o servidor informa que o usuário não tem os direitos de acessos necessários
	404 Not Found	Informa que o servidor não encontrou o recurso solicitado
	429 Too Many Requests	Não é tão comum, mas pode ser utilizar para informar que o cliente excedeu o limite permitido de requisições

API REST

Códigos de status HTTP

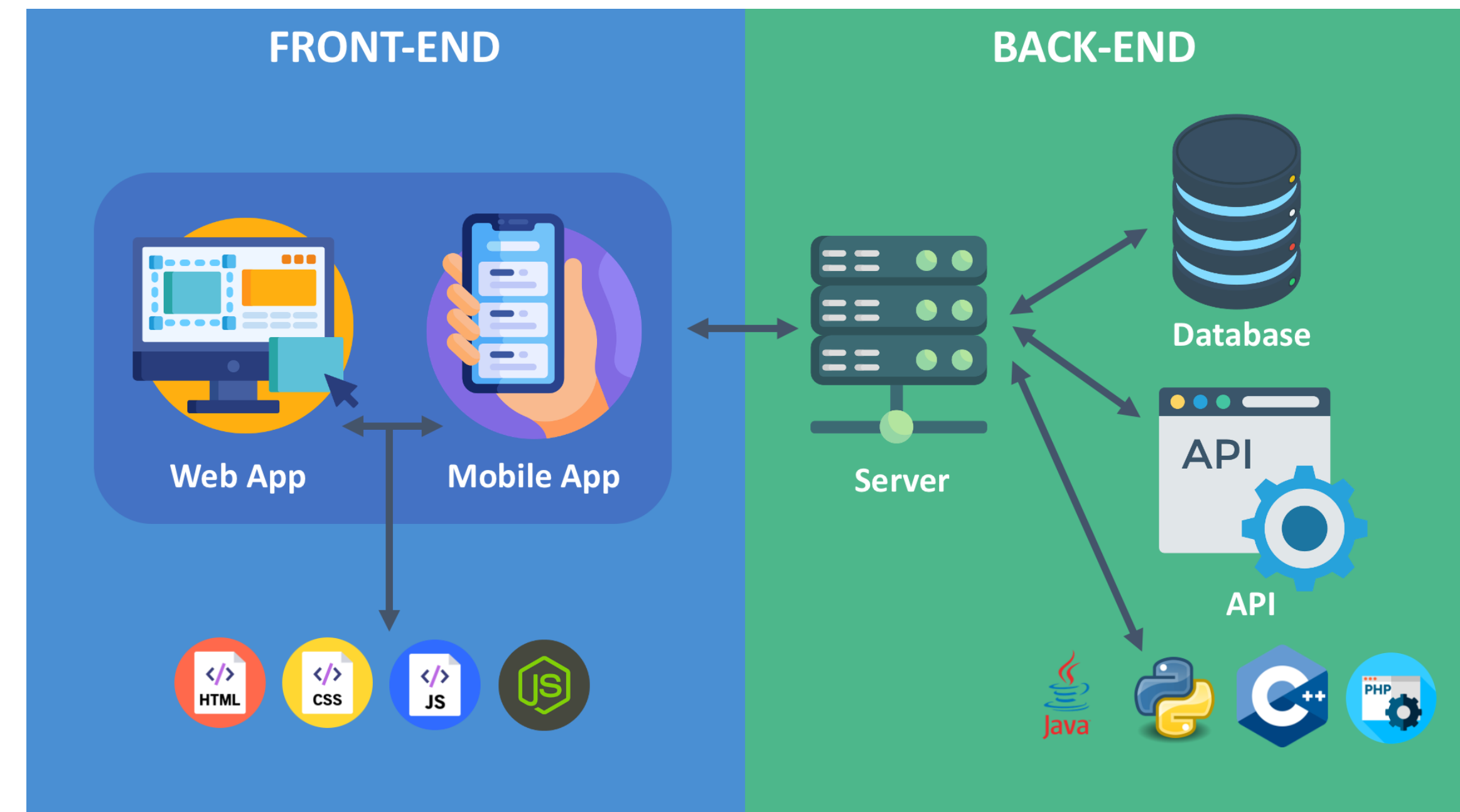
Grupo	Código	Quando
5xx - Enviadas quando ocorre um erro no lado do servidor	500 Internal Server Error	Erro mais genérico do grupo. Informa que o servidor encontrou um cenário inesperado de erro com o qual não soube lidar
	503 Service Unavailable	Normalmente é utilizado para informar que o servidor está fora do ar, em manutenção ou sobrecarregado

API REST

Exemplos de rotas

Tarefa/Funcionalidade	HTTP Method	URL
Listar mangás	GET	/mangas
Adicionar um mangá	POST	/mangas
Ver detalhes de um mangá	GET	/mangas/:id
Atualizar um mangá	PUT	/mangas/:id
Remover um mangá	DELETE	/mangas/:id

Back-end vs Front-end



Back-end vs Front-end

Desenvolvedores de Front-end trabalham naquilo que o usuário consegue ver enquanto desenvolvedores de back-end trabalham na infraestrutura de suporte

Back-end vs Front-end

Front-end

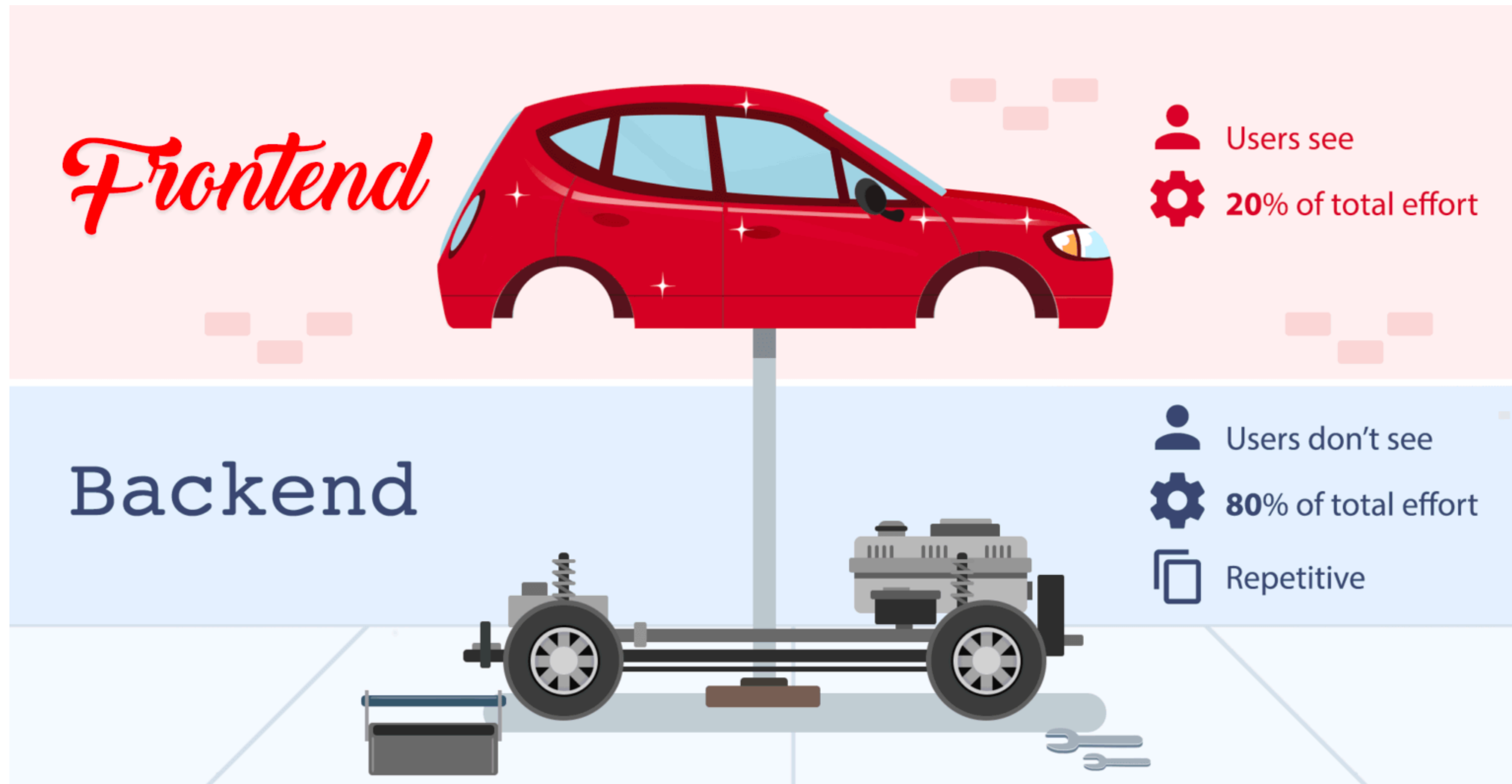
- Trabalham com a interface de usuário
- Lado do “cliente”
- Trabalham sempre pensando no usuário
- Desenvolvem elementos e funcionalidades que serão vistas pelo usuário
- Tecnologias mais utilizadas: **HTML, CSS, JS.**
 - Atualmente: React, Vue, Angular

Back-end vs Front-end

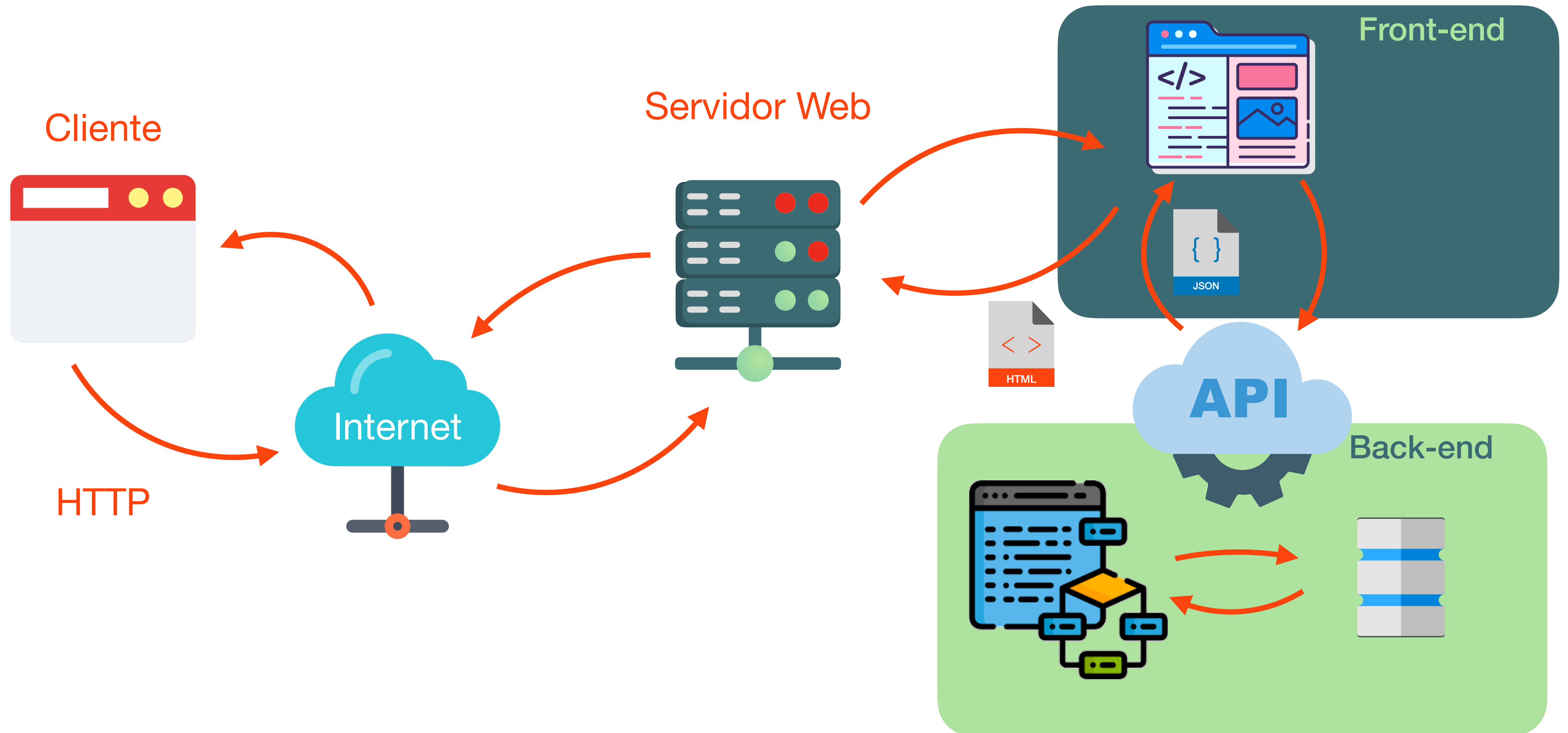
Back-end

- Trabalham na parte que os usuários não conseguem ver
 - É acessado indiretamente via front-end
- Lado do servidor
- Responsável por armazenar os dados e garantir a segurança da aplicação
- Tecnologias que pode ser utilizadas:
 - JavaScript, Java, Python, Ruby e etc ...

Back-end vs Front-end



Back-end vs Front-end



Referências

- What Is the Difference Between Front-End and Back-End Development?
- Front End vs. Back End: What's the Difference?
- Frontend vs. backend: what's the difference?
- O que é a API REST e como ela difere de outros tipos?
- A anatomia de uma API RESTful
- What is REST
- API REST: o que é e como montar uma API sem complicação?

Por hoje é só