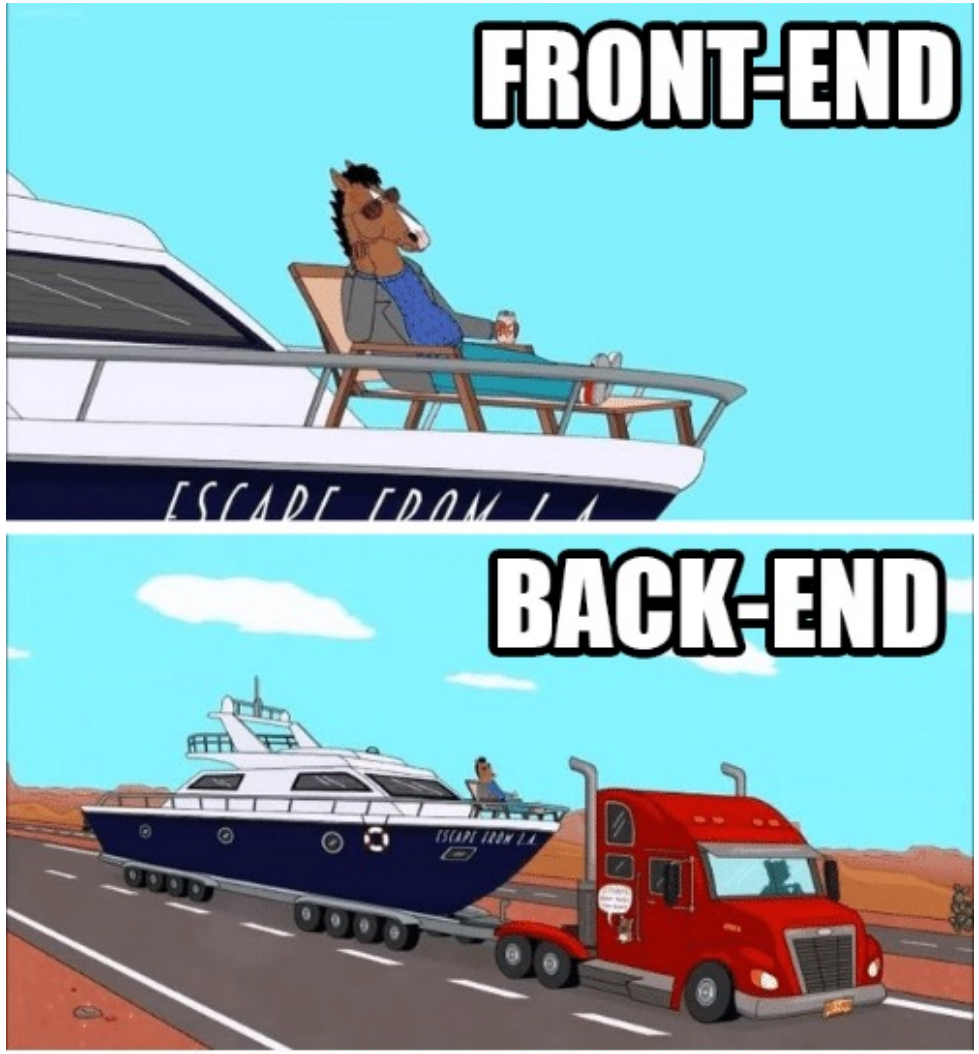
UniEVANGÉLICA
UNIVERSIDADE EVANGÉLICA DE GOIÁS

CURSO DE ENGENHARIA DE SOFTWARE

API

Anápolis – 2023.1

Profº Eduardo F Souza | Associação Educativa Evangélica



Front-end VS Back-end


WEB SERVER
NGINX
BACKEND LANGUAGE
php
Java
node
python
Scala
DATABASE
mongoDB
redis
MySQL
PostgreSQL
jQuery
zepto.js
HTML CSS JS
INTERNET
CLIENT SIDE
SERVER SIDE


BACKEND
FRONTEND
API's

Frontend / Backend

Frontend é a parte visível do aplicativo ou site que os usuários veem e interagem diretamente. É responsável pela interface do usuário, a apresentação visual dos dados e a interatividade do aplicativo. As tecnologias comuns usadas no desenvolvimento de frontend incluem HTML, CSS e JavaScript.

O backend, por outro lado, é a parte não visível do aplicativo ou site que lida com o processamento de dados e a lógica de negócios. Ele é responsável por armazenar e gerenciar os dados do aplicativo, processar solicitações do cliente e fornecer respostas apropriadas. As tecnologias comuns usadas no desenvolvimento de backend incluem bancos de dados, servidores e linguagens de programação como Python, Java e PHP.

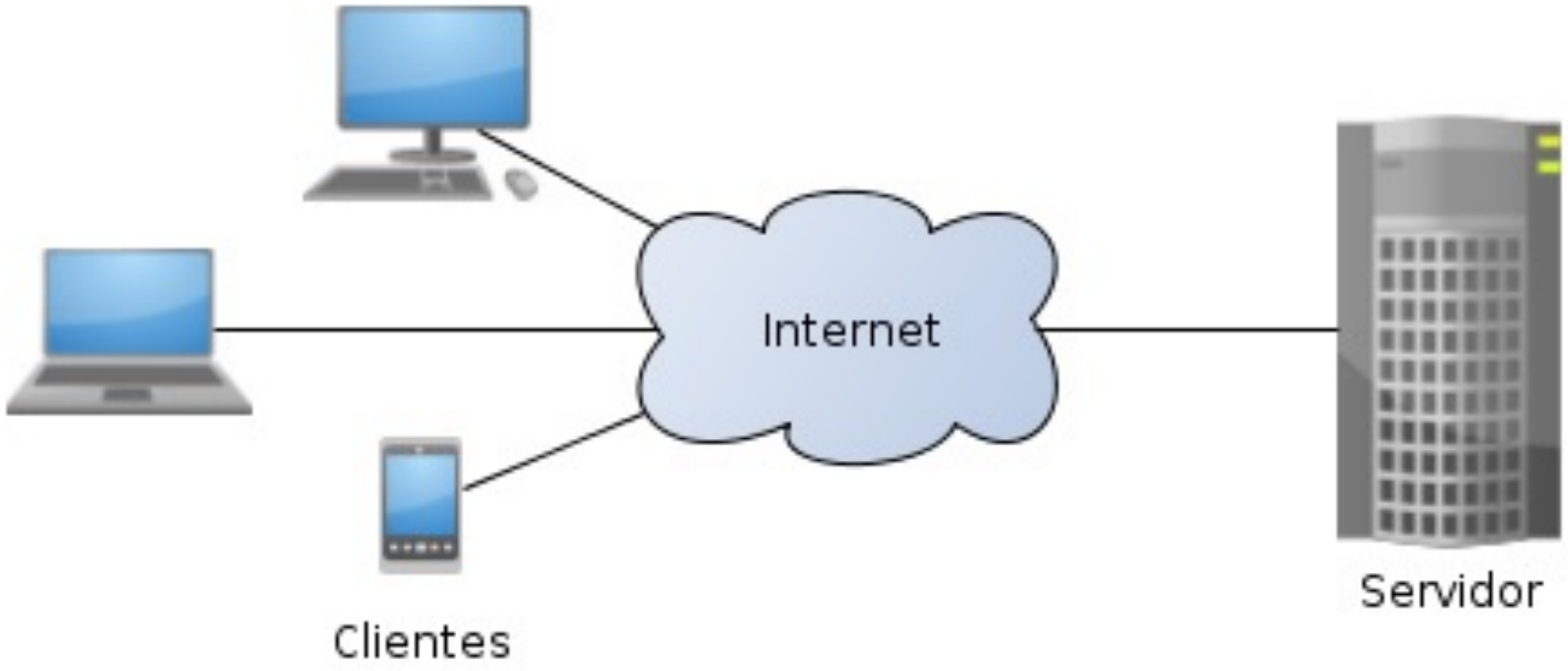
Arquitetura Cliente - Servidor

A arquitetura cliente-servidor é um modelo de comunicação em que os sistemas computacionais são divididos em dois componentes principais: o cliente e o servidor.

O cliente é um programa ou dispositivo que faz solicitações para um servidor.

O servidor é responsável por receber e processar essas solicitações e retornar uma resposta para o cliente.

Arquitetura Cliente - Servidor



HTTP / HTTPS

O HTTP (Hypertext Transfer Protocol) é o protocolo padrão usado para transferir dados entre um cliente e um servidor web. Ele é utilizado para enviar solicitações de páginas web e outros recursos da internet, bem como receber as respostas do servidor.

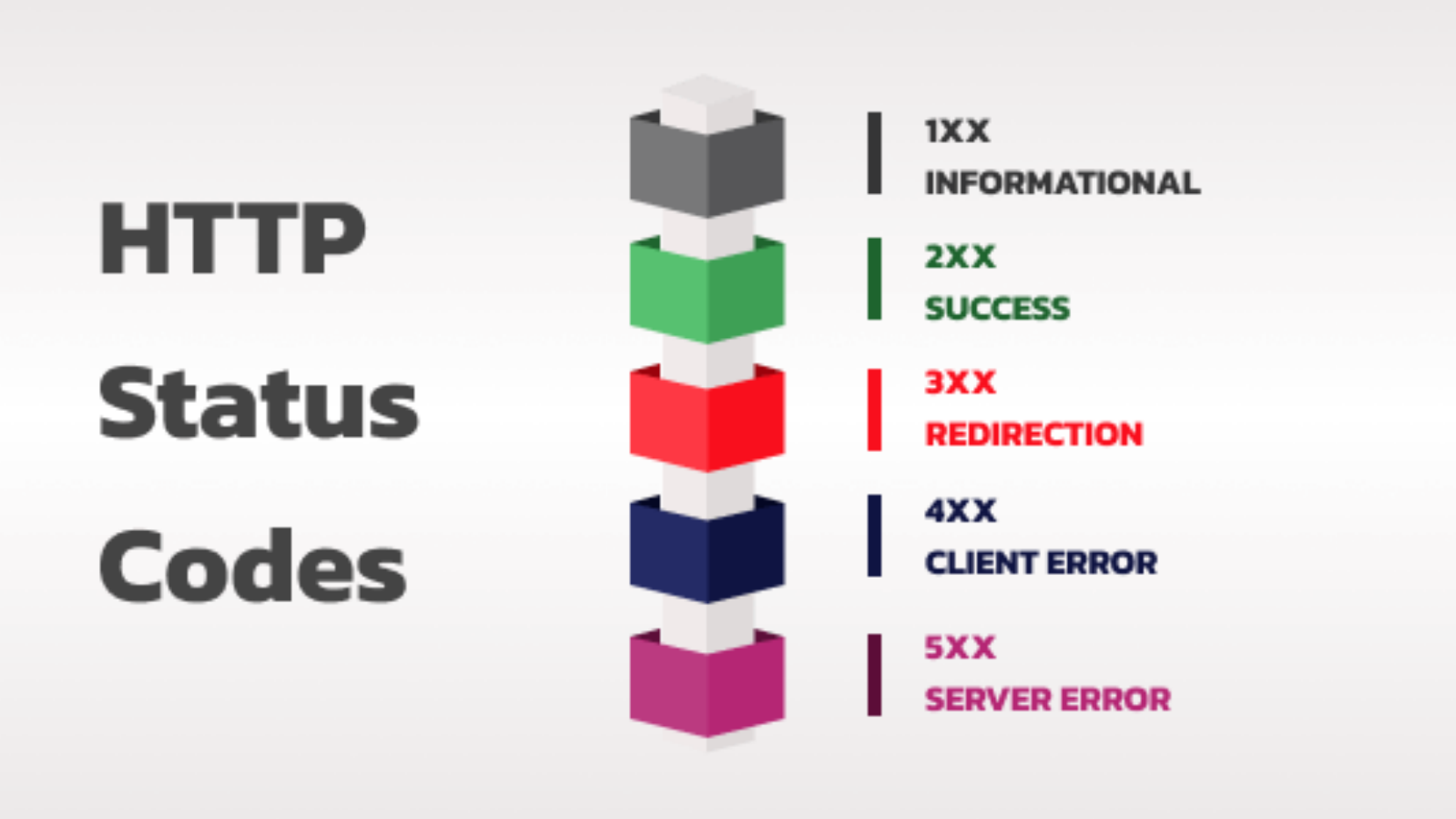
O HTTP é um protocolo de texto simples, o que significa que as informações são transmitidas sem criptografia, tornando as conexões HTTP vulneráveis a ataques de interceptação de dados e outras ameaças à segurança.

HTTP / HTTPS

O HTTPS (Hypertext Transfer Protocol Secure) é uma extensão do HTTP que adiciona uma camada de segurança por meio de criptografia.

O HTTPS usa um certificado digital para autenticar o servidor web e criptografar os dados transmitidos entre o cliente e o servidor. Isso ajuda a proteger as informações do usuário, como senhas, informações de cartão de crédito e outras informações pessoais, de serem interceptadas por hackers ou outras entidades mal-intencionadas.

HTTP Code


HTTP
Status
Codes
1XX
INFORMATIONAL
2XX
SUCCESS
3XX
REDIRECTION
4XX
CLIENT ERROR
5XX
SERVER ERROR

API

API (Application Programming Interface) é uma interface de programação de aplicações que permite a comunicação entre diferentes softwares e sistemas. Em outras palavras, uma API é um conjunto de rotinas, protocolos e ferramentas que permitem que um aplicativo se comunique com outro e use seus recursos.

API Rest

API REST (Representational State Transfer) é um padrão de arquitetura para o design de APIs (Application Programming Interface) baseado no protocolo HTTP (Hypertext Transfer Protocol). As APIs RESTful permitem que os sistemas interajam uns com os outros de maneira padronizada e escalável.

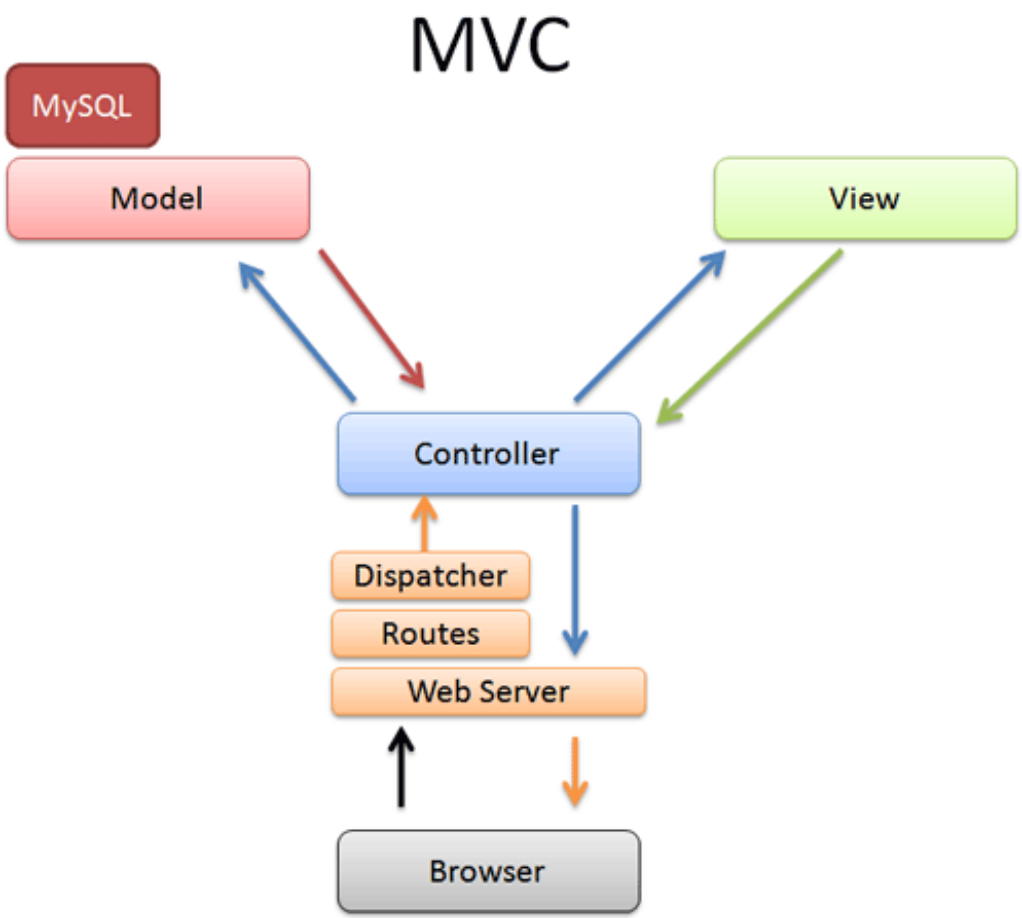
As APIs RESTful são baseadas em recursos, que representam informações ou funcionalidades em um aplicativo. Os recursos são acessados por meio de URLs (Uniform Resource Locators) e podem ser criados, lidos, atualizados e excluídos (CRUD) usando os verbos HTTP padrão, como GET, POST, PUT e DELETE.

CRUD

JSON

```json
{
   "cliente": {
       "id": 2020,
       "nome": "Maria Aparecida"
   },
   "pagamentos": [
       {
           "id": 123,
           "descricacao": "Compra do livro Cangaceiro JavaScript",
           "valor": 50.5
       },
       {
           "id": 124,
           "descricacao": "Mensalidade escolar",
           "valor": 1500
       }
   ]
}
```

Arquitetura MVC



Frameowork

- Um framework, é um conjunto de componentes, padrões, bibliotecas e ferramentas que auxiliam no desenvolvimento de aplicações ou sistemas. Ele fornece uma estrutura básica que facilita a criação e manutenção de projetos, permitindo que os desenvolvedores se concentrem na lógica específica do aplicativo em vez de resolver problemas comuns de baixo nível. O uso de frameworks ajuda a promover a reutilização de código, a padronização e a eficiência no desenvolvimento de software (SOMMERVILLE, 2011).

Framework - Vantagens

- Desenvolvimento mais rápido
- Reutilização de código
- Manutenção facilitada
- Melhores práticas e padrões
- Comunidade e suporte
- Segurança

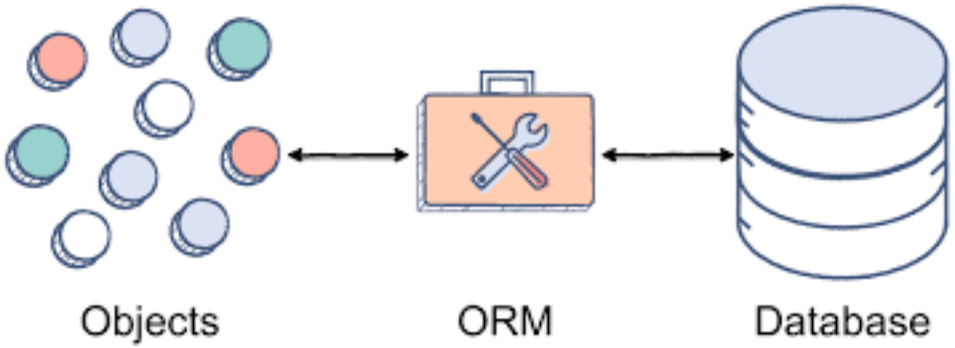
Framework - Desvantagens

- Curva de aprendizado
- Rigidez
- Desempenho
- Dependência
- Tamanho e sobrecarga

Tecnologias Backend

- Java / Spring
- Python / Flask, Django
- Ruby / Rails
- PHP / Laravel
- JS / Express

ORM

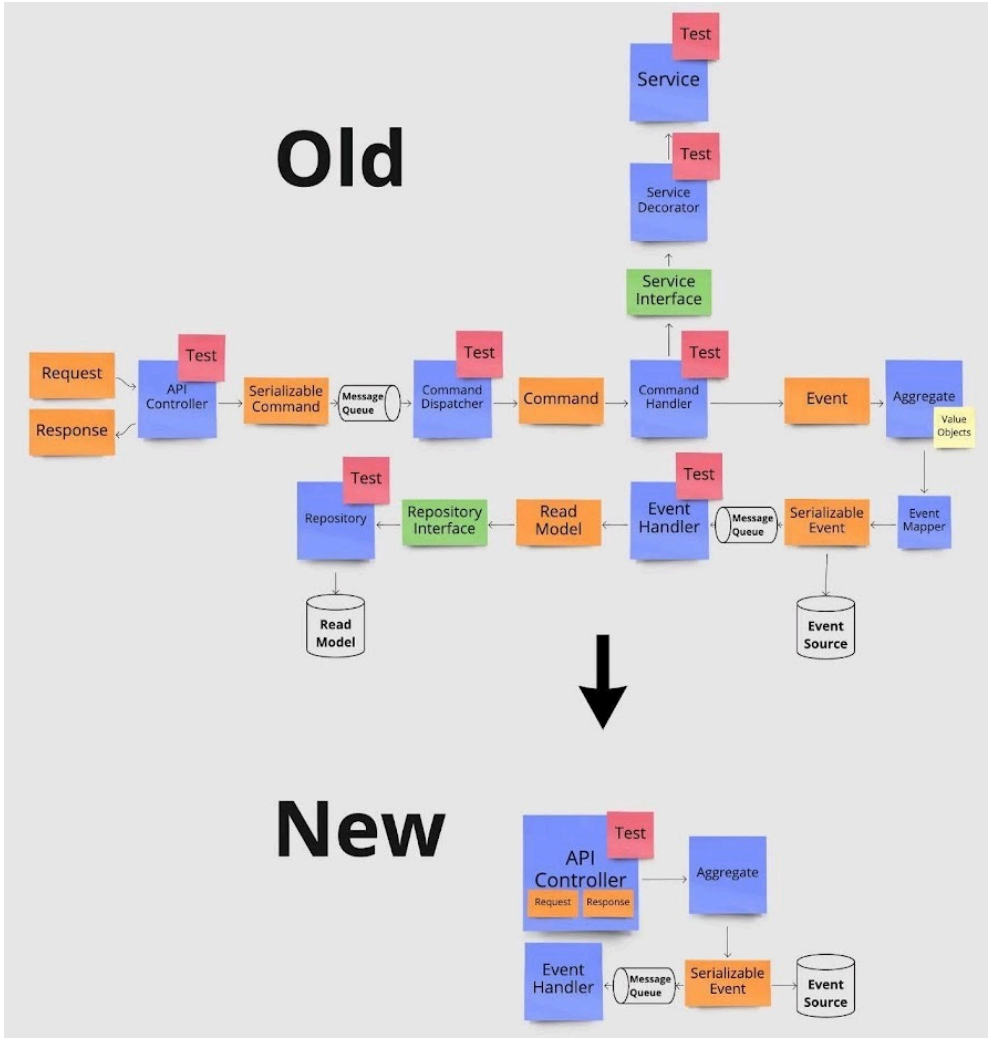


ORM

ORM (Object-Relational Mapping, ou Mapeamento Objeto-Relacional) é uma técnica de programação que facilita a conversão de dados entre sistemas de tipos incompatíveis, como bancos de dados relacionais e linguagens de programação orientadas a objetos. Em outras palavras, um ORM permite que os desenvolvedores interajam com bancos de dados usando objetos e classes em vez de instruções SQL diretas.

Gerenciador de Dependências

Um gerenciador de dependências é uma ferramenta que simplifica o processo de gerenciamento de bibliotecas, pacotes e módulos que um projeto de software utiliza. Ele automatiza o processo de instalação, atualização, configuração e remoção das dependências, garantindo que as versões corretas e compatíveis sejam utilizadas. Além disso, um gerenciador de dependências pode ajudar a resolver conflitos entre pacotes e a garantir a segurança das dependências usadas no projeto.


Old
Test
Service
Test
Service Decorator
Service Interface
Request
Response
Test
API Controller
Serializable Command
Message Queue
Test
Command Dispatcher
Command
Test
Command Handler
Event
Aggregate
Value Objects
Test
Repository
Repository Interface
Read Model
Test
Event Handler
Message Queue
Serializable Event
Event Mapper
Read Model
Event Source
New
Test
API Controller
Request
Response
Aggregate
Event Handler
Message Queue
Serializable Event
Event Source

Atividade Pratica

- Projetar a API REST para de acordo com tema individual da disciplina.
- Ex:

```
GET /api/cars - recuperar a lista de carros
POST /api/cars - criar um novo carro
GET /api/cars/{car_id} - recuperar informações de um carro específico por ID
PUT /api/cars/{car_id} - atualizar informações de um carro específico por ID
DELETE /api/cars/{car_id} - excluir um carro específico por ID
```