
hotmart

Construindo uma aplicação de ponta a ponta com Spring Boot, VueJS e Docker

Quem sou?

Fábio Viana

Hotmart Specialist & DevOps Team Leader

Bacharel em Sistemas de Informação pela PUC Minas - São Gabriel.

Atuo há mais de 10 anos com arquitetura de software, desenvolvimento e operação.

E-mail: fabio.viana@hotmart.com
GitHub: @fabioviana

hotmart

Hotmart

A Hotmart é a maior empresa especializada na venda e distribuição de produtos digitais da América Latina, **líder de mercado desde sua fundação, em 2011.**

Em constante processo de internacionalização, a empresa possui escritórios em **Belo Horizonte, Madri, Amsterdã, Bogotá e Cidade do México.**

Hotmart

+5 Milhões
DE COMPRADORES

+200
PAÍSES
DIFERENTES

+2 Milhões
DE USUÁRIOS

+150 Mil
PRODUTOS

Bora pro que interessa...

CODE CODE CODE

Antigamente...

Características

Monolitos

- Inflexíveis - construídos com apenas uma tecnologia
- Não confiáveis - falha em uma única funcionalidade pode causar falha em todo o sistema
- Não escaláveis - não escalam facilmente nem rapidamente
- Impedem o desenvolvimento contínuo - muitas funcionalidades não podem ser construídas/publicadas de forma paralela e independente
- Desenvolvimento lento - muito esforço entre uma versão e outra
- Impedem o desenvolvimento de aplicações complexas - quanto mais complexo o sistema, mais acopladas são as regras

Características

Micro Serviços

- Este termo surgiu em 2011 para designar um novo estilo arquitetural
- A proposta é desenvolver sistemas flexíveis, escaláveis e de simples manutenção
- A filosofia é “fazer serviços pequenos, mas bem feitos”
- Para começar a explicar este estilo, é importante compará-lo com o estilo monolítico

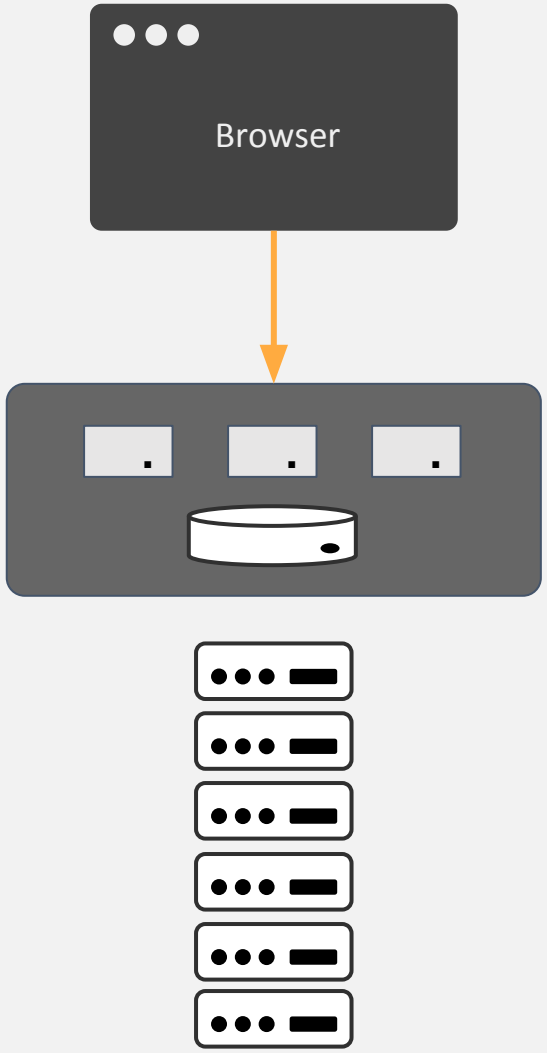
Características

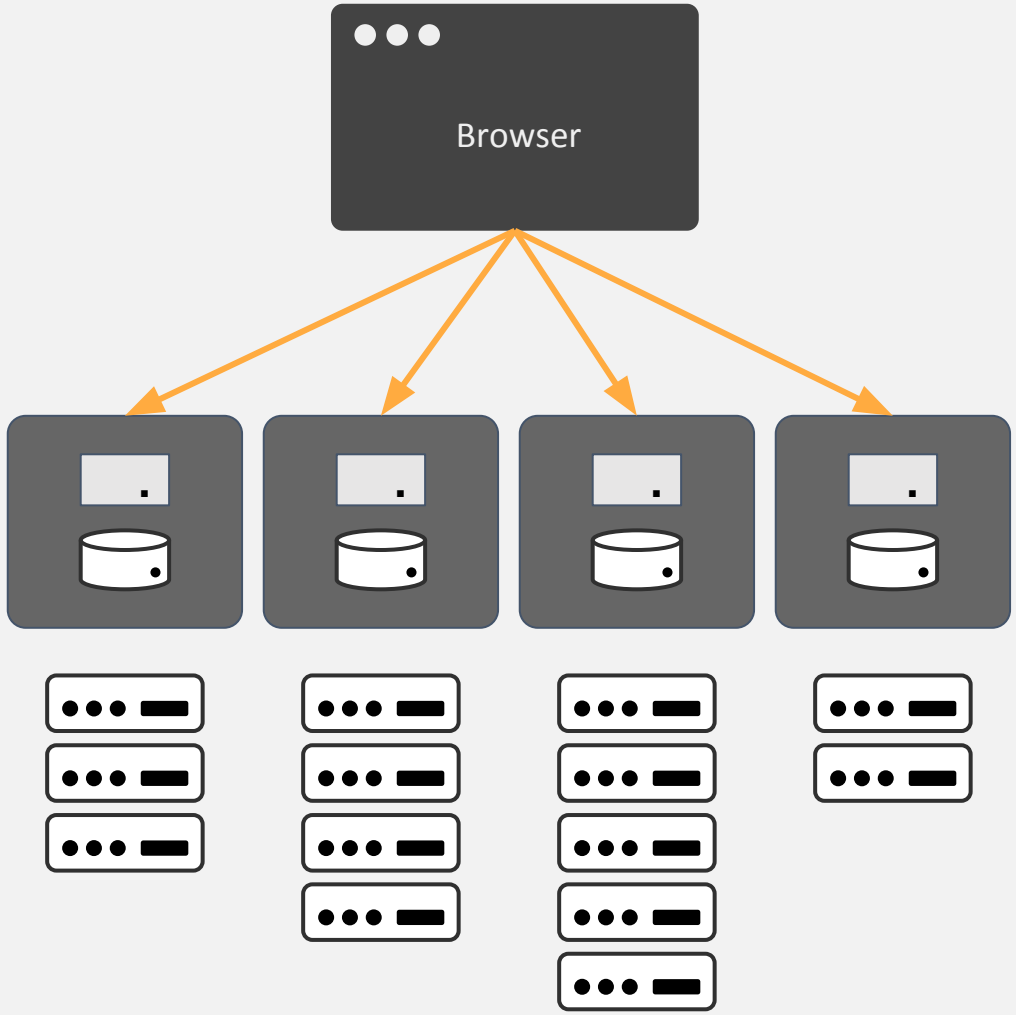
Micro Serviços

- Desacoplamento - sistemas de fácil alteração, publicação e escala
- Componentização - componentes independentes podem ser facilmente atualizados, substituídos e reutilizados
- Áreas de negócios - simples e focados em área de negócios
- Autonomia - desenvolvedores podem trabalhar de forma independente, aumentando a velocidade
- Entrega contínua - permite entrega de software mais frequentes
- Flexibilidade - escolha da melhor tecnologia para cada problema
- Agilidade - funcionalidades são facilmente desenvolvidas ou descartadas

Topologia

Monolitos X Micro Serviços

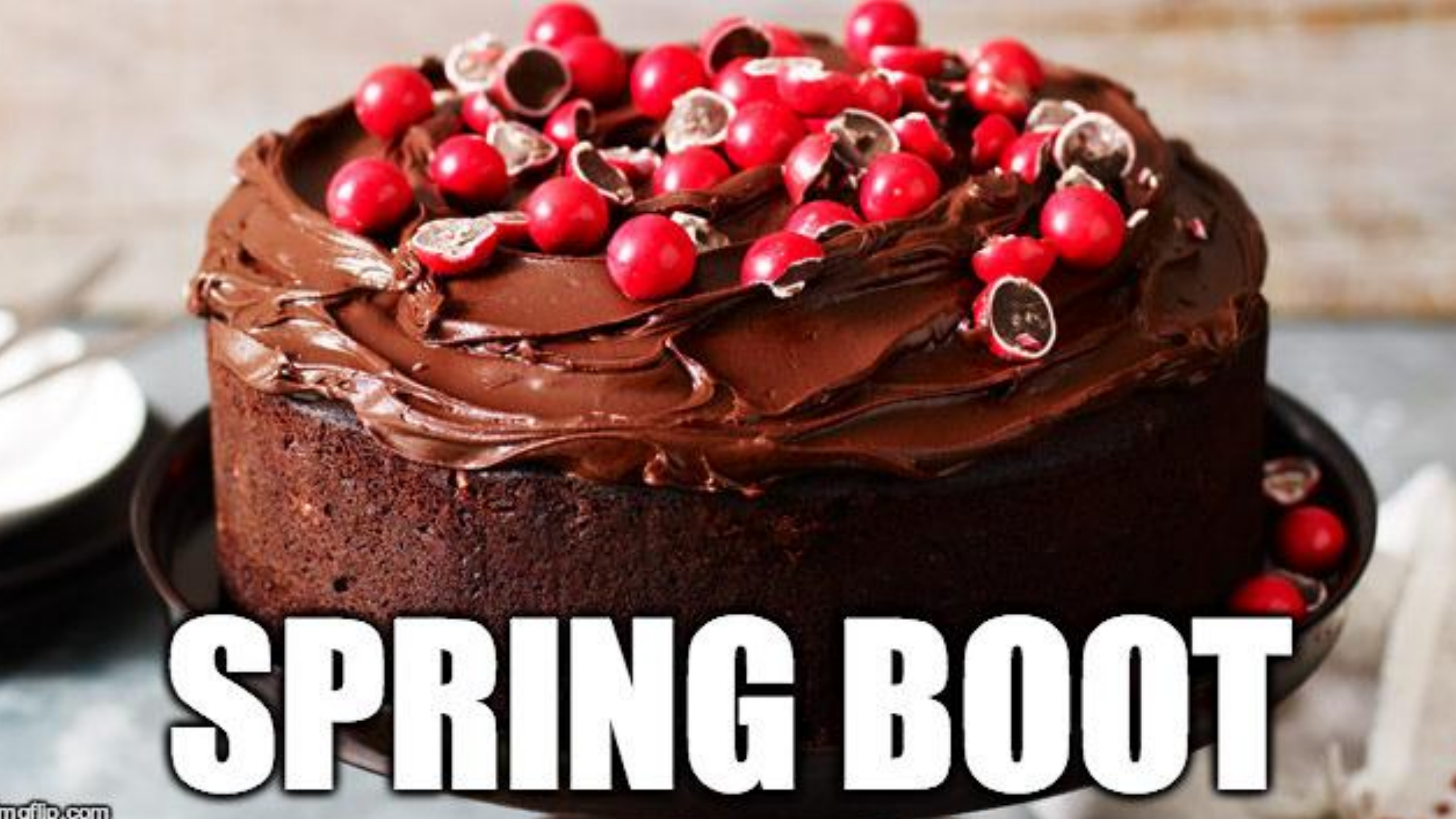




Spring Boot

(Backend)

SPRING FRAMEWORK


SPRING BOOT

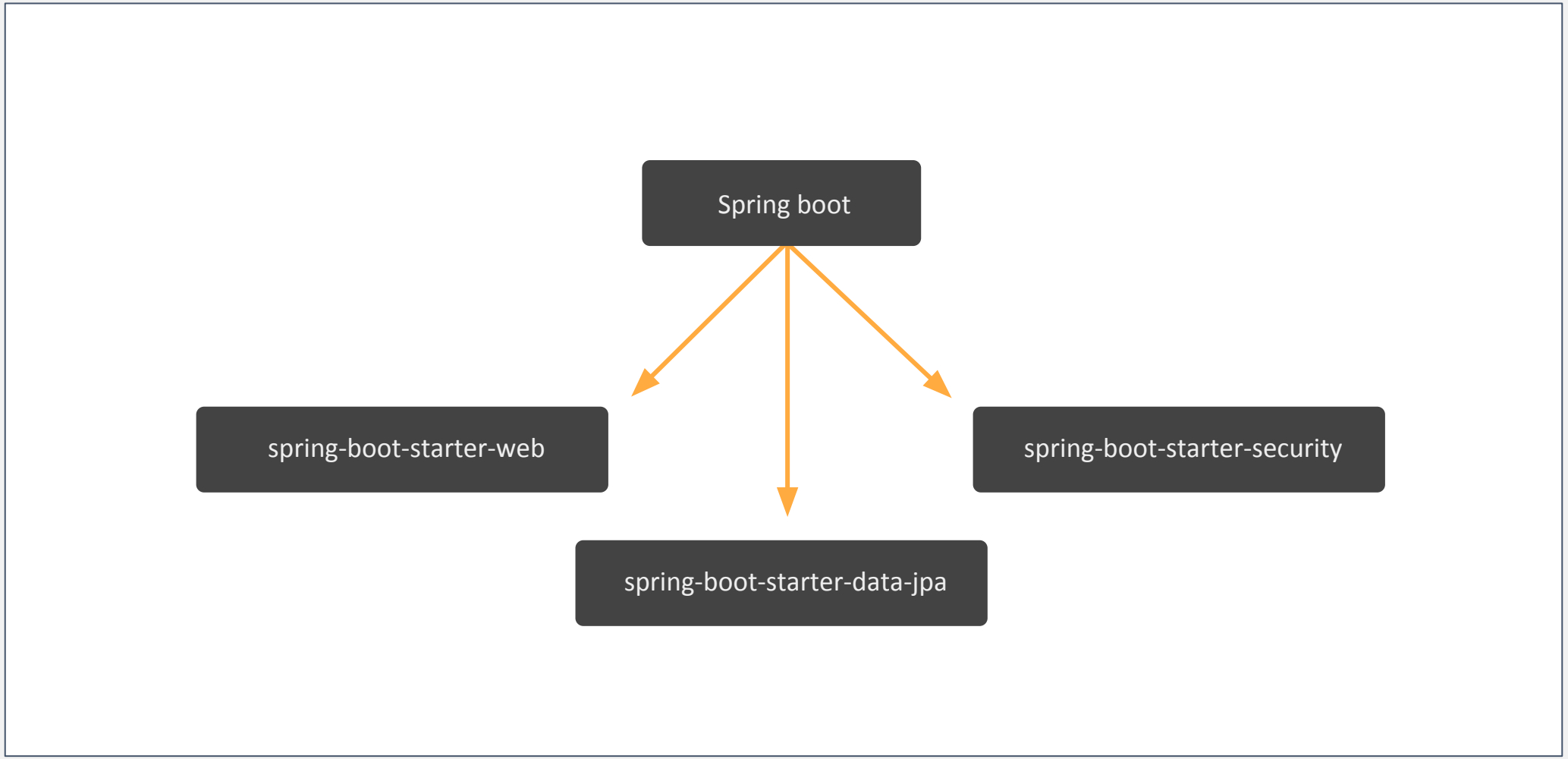
O que é isso?!?!

- Framework que permite construir e rodar uma aplicação facilmente
- Segue os melhores design patterns
- Possui, por padrão, configurações otimizadas para produção
- Possui ferramentas previamente configuradas e prontas para uso
- Uma forma fácil de iniciar o projeto é usando o https://start.spring.io/
- Cada ferramenta é um módulo independente e plugável, conhecidos como starters
- Implementa o padrão RESTful api

Pré requisitos

- RESTful: dentre outras coisas, ele define como deve ser o uso do protocolo HTTP para cada tipo de operação transacional, ex:
  - GET: operação de leitura de registros
  - POST: operação de inclusão de registro
  - PUT: operação de alteração de registro
  - DELETE: operação de remoção de registro
- Injeção de dependências: técnica muito usada para controlar as instâncias de classes de sistema, possibilitando injetar código e comportamentos comuns, como auditoria, logs, etc.
- Maven ou Gradle: gerenciador de pacotes para java

Módulos


Spring boot
spring-boot-starter-web
spring-boot-starter-security
spring-boot-starter-data-jpa

Spring Boot

Aplicação

```java
package com.hotmart.handson;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class HandsonApplication {

    public static void main(String[] args) {
        SpringApplication.run(HandsonApplication.class, args);
    }

}
```

Spring Boot

Entity

```java
package com.hotmart.handson.entity;

import org.springframework.data.annotation.Id;
import org.springframework.data.mongodb.core.mapping.Document;

import javax.validation.constraints.NotBlank;
import java.util.Date;

@Document
public class Idea {

    @Id
    private String id;

    @NotBlank
    private String title;

    private String description;
    private Date createdAt;

    public String getId() {
        return id;
    }
    public void setId(String id) {
        this.id = id;
    }
    public String getTitle() {
        return title;
    }
    public void setTitle(String title) {
        this.title = title;
    }
    public String getDescription() {
        return description;
    }
    public void setDescription(String description) {
        this.description = description;
    }
    public Date getCreatedAt() {
        return createdAt;
    }
    public void setCreatedAt(Date createdAt) {
        this.createdAt = createdAt;
    }
}
```

Spring Boot

Controller

```java
package com.hotmart.handson.controller;

import com.hotmart.handson.entity.Idea;
import com.hotmart.handson.repository.IdeaRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

import javax.validation.Valid;
import java.util.List;

@RestController
@RequestMapping("/api/idea")
@CrossOrigin(origins = "*", methods = {RequestMethod.DELETE, RequestMethod.GET, RequestMethod.OPTIONS,
RequestMethod.POST, RequestMethod.PUT}, allowedHeaders = "*")
public class IdeaController {

    @Autowired
    private IdeaRepository ideaRepository;

    @PostMapping
    public void salvar(@RequestBody @Valid Idea idea) {
        ideaRepository.save(idea);
    }

    @GetMapping
    public List<Idea> listar() {
        return ideaRepository.findAll();
    }
}
```

Spring Boot

Repository

```java
package com.hotmart.handson.repository;

import com.hotmart.handson.entity.Idea;
import org.springframework.data.mongodb.repository.MongoRepository;
import org.springframework.stereotype.Repository;

import java.util.List;

@Repository
public interface IdeaRepository extends MongoRepository<Idea, String> {

    /**
     * Buscará por registros onde o titulo contenha o parâmetro informado
     * @param title
     * @return
     */
    List<Idea> findByTitleLike(String title);

}
```

Prática

- Instalar JDK Java 8 ou superior (definir environment **JAVA_HOME** apontando para a instalação da JDK)
- Instalar editor IntelliJ IDEA ou outro de preferência
- Instalar comando CURL ou outro equivalente para testes na api
- Criar uma aplicação para cadastro de ideias
  - Modelo:

```
{
 "title": "string",
 "description": "string",
 "createdAt": "date"
}
```

  - Operações: cadastro, listagem, alteração e exclusão

Prática

- Utilizar https://start.spring.io/
- Utilizar starters/modules:
  - Spring Data MongoDB
  - Spring Web Starter
- Rodar mongo local: `sudo docker run --rm -d -p 27017:27017 --name mongo mongo`
- Testar api via curl:

```
curl -X POST \
  http://localhost:8080/api/idea \
  -H 'Content-Type: application/json' \
  -H 'cache-control: no-cache' \
  -d '{
     "title": "teste 1"
}'
```

VueJS

(Frontend)

O que é?

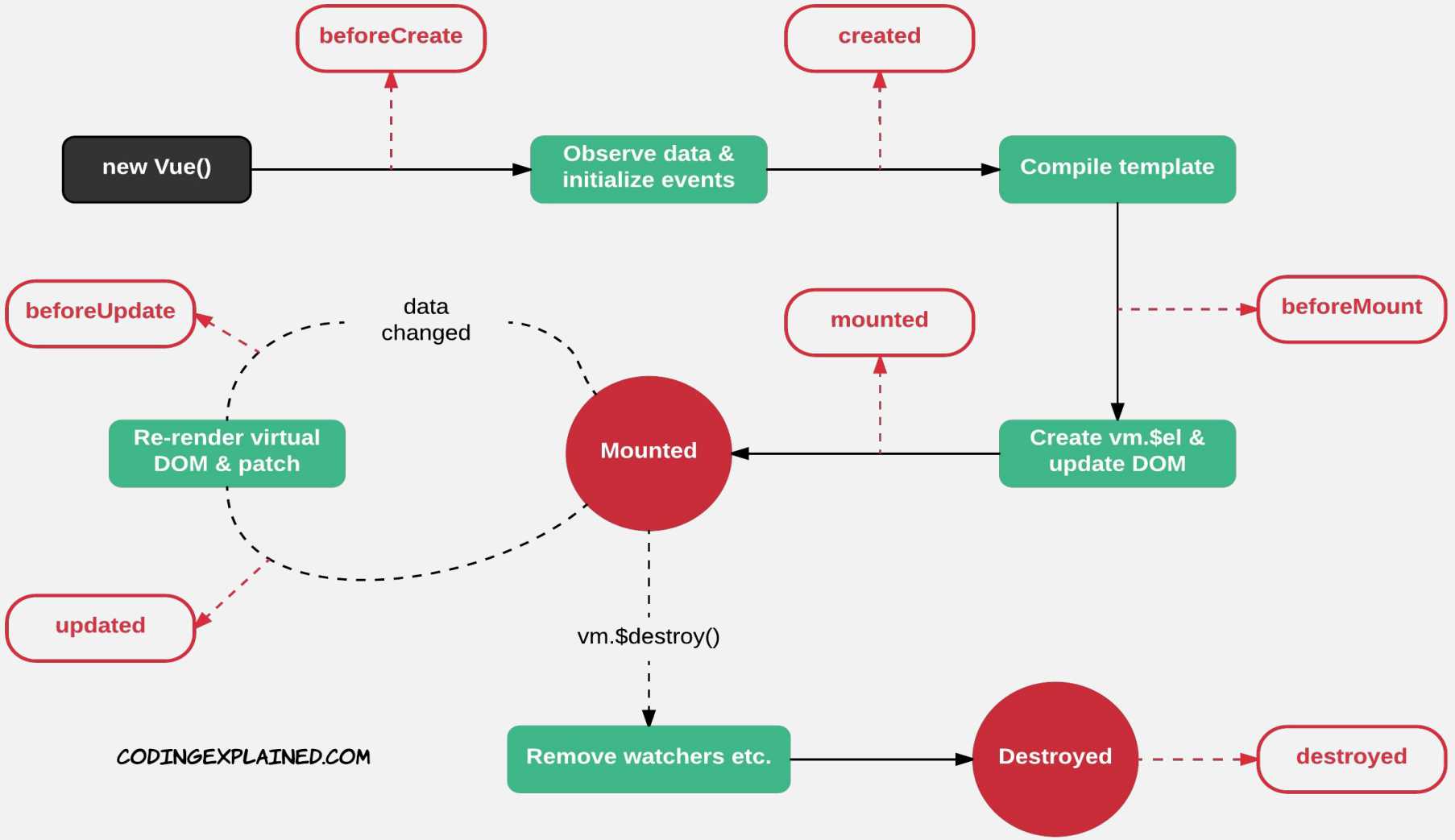
- É um framework criado por um engenheiro do Google em 2014
- Foi inspirado no AngularJS e no padrão MVVM (Model-View-ViewModel)
- Features avançadas como roteamento, gestão de estados e ferramentas de build são oferecidas por pacotes oficiais opcionais
- Utiliza uma sintaxe de template baseada em HTML
- Os templates são compilados em funções que geram um DOM virtual
- Possui um sistema reativo em que cada componente observa suas dependências durante o render

Pré requisitos

- NodeJS: muito usado para criar ferramentas de desenvolvimento para aplicações frontend
- npm: gerenciador de dependências para javascript
- Axios: lib javascript para facilitar consumo de apis RESTful

VueJS

Lifecycle



Main

```
<div id="app-4">
  <ol>
    <li v-for="todo in todos">
      {{ todo.text }}
    </li>
  </ol>
</div>
```

```
var app4 = new Vue({
  el: '#app-4',
  data: {
    todos: [
      { text: 'Learn JavaScript' },
      { text: 'Learn Vue' },
      { text: 'Build something awesome' }
    ]
  }
})
```

VueJS

Component

```
<script>
    export default {
        name: 'SampleComponent',
        props: {},
        data() {
            return {}
        },
        methods: {
        },
    }
</script>
<style>
</style>

<template>
</template>
```

- name: nome do componente
- props: propriedades ou parâmetros que o componente poderá receber
- data: variáveis com valores iniciais
- methods: funções do componente
- validations: definição das validações
- mounted: executado quando o componente está pronto

VueJS

Data

```
data() {
  categories: [],
  form: {
    title: '',
    description: '',
    categoryId: '',
  }
}
```

- Também são utilizadas para definir quais atributos de uma tela serão monitorados para disparar eventos

VueJS

Validations

```
validations: {
    form: {
        title: {
          required,
          minLength: minLength(2)
        },
        description: {
          required,
          minLength: minLength(2)
        },
        categoryId: {
          required
        }
    }
}
```

- Regras de validação dos campos apresentados no formulário
- Definições de origem da biblioteca Vuelidate

VueJS

Methods

```
methods: {
  stopPropagation(event) {
    event.stopPropagation();
  },
  close() {
    parent.postMessage('MODAL_CLOSE', '*');
  },
  submit () {
    this.$v.$touch();
    if (!this.$v.$invalid) {
        ...
    }
  }
},
```

- Funções para controle diversos, como por exemplo o envio de dados de um formulário para o back-end

VueJS

Mounted

```
mounted () {
  parent.postMessage('MODAL_READY', '*');
  ...
}
```

- Útil para realizar chamadas em AJAX ou operações que precisam que o componente esteja pronto
- Uma estratégia é carregar o componente com uma mensagem temporária, por exemplo: “Carregando” e utilizar essa função para carregar o conteúdo dinâmico

VueJS

Router

```
import Vue from 'vue'
import Router from 'vue-router'

Vue.use(Router)

export default new Router({
  routes: [
    {
      path: '/',
      name: 'Home',
      component: () => import('@/pages/Home'),
    },
    {
      path: '/Cadastro',
      name: 'Cadastro',
      component: () => import('@/pages/Cadastro')
    }
  ]
})
```

- Componente usado para realizar navegação entre telas
- Cada página da aplicação precisa ser registrada e seu código importado em cada rota correspondente
- Os paths correspondem ao endereço da página, prefixadas com o endereço e porta de acesso

VueJS

Router

```
<template>
  <div id="app">
    <img alt="Vue logo" src="./assets/logo.png">
    <router-view></router-view>
  </div>
</template>
```

- No componente principal, onde o VueJS iniciará sua instância, em seu template será necessário definir a tag <router-view></router-view>
- Nesta tag será exibido o conteúdo de cada página navegada, funcionando como um template
- Neste exemplo, em todas as páginas o logo será exibido no topo

Statements

- Conditional Rendering:
  v-if
  v-else
  v-show
- List Rendering:
  v-for
- Events Handling:
  v-on:click
  v-on:submit

Statements

```
<h1 v-if="ok">Yes</h1>
<h1 v-else>No</h1>
```

```
<div id="example">
  <button v-on:click="greet">Greet</button>
</div>

var vm = new Vue({
  el: '#example',
  data: {
    name: 'Vue.js'
  },
  // define methods under the `methods` object
  methods: {
    greet: function (event) {
      // `this` inside methods point to the Vue instance
      alert('Hello ' + this.name + '!')
      // `event` is the native DOM event
      alert(event.target.tagName)
    }
  }
})
```

```
<ul id="example-1">
  <li v-for="item in items">
    {{ item.message }}
  </li>
</ul>

var example1 = new Vue({
  el: '#example-1',
  data: {
    items: [
      { message: 'Foo' },
      { message: 'Bar' }
    ]
  }
})
```

Prática

- Instalar NodeJS 10 ou superior
- Instalar Visual Code ou outro editor que preferir
- Criar frontend para o cadastro de ideias com as seguintes features:
  - Tela de listagem, com ação para remover registro e abrir tela de edição
  - Tela de cadastro/edição
- Utilizar https://cli.vuejs.org/ via comando:

```
npm install -g @vue/cli
vue create frontend
```

- Rodar aplicação frontend:

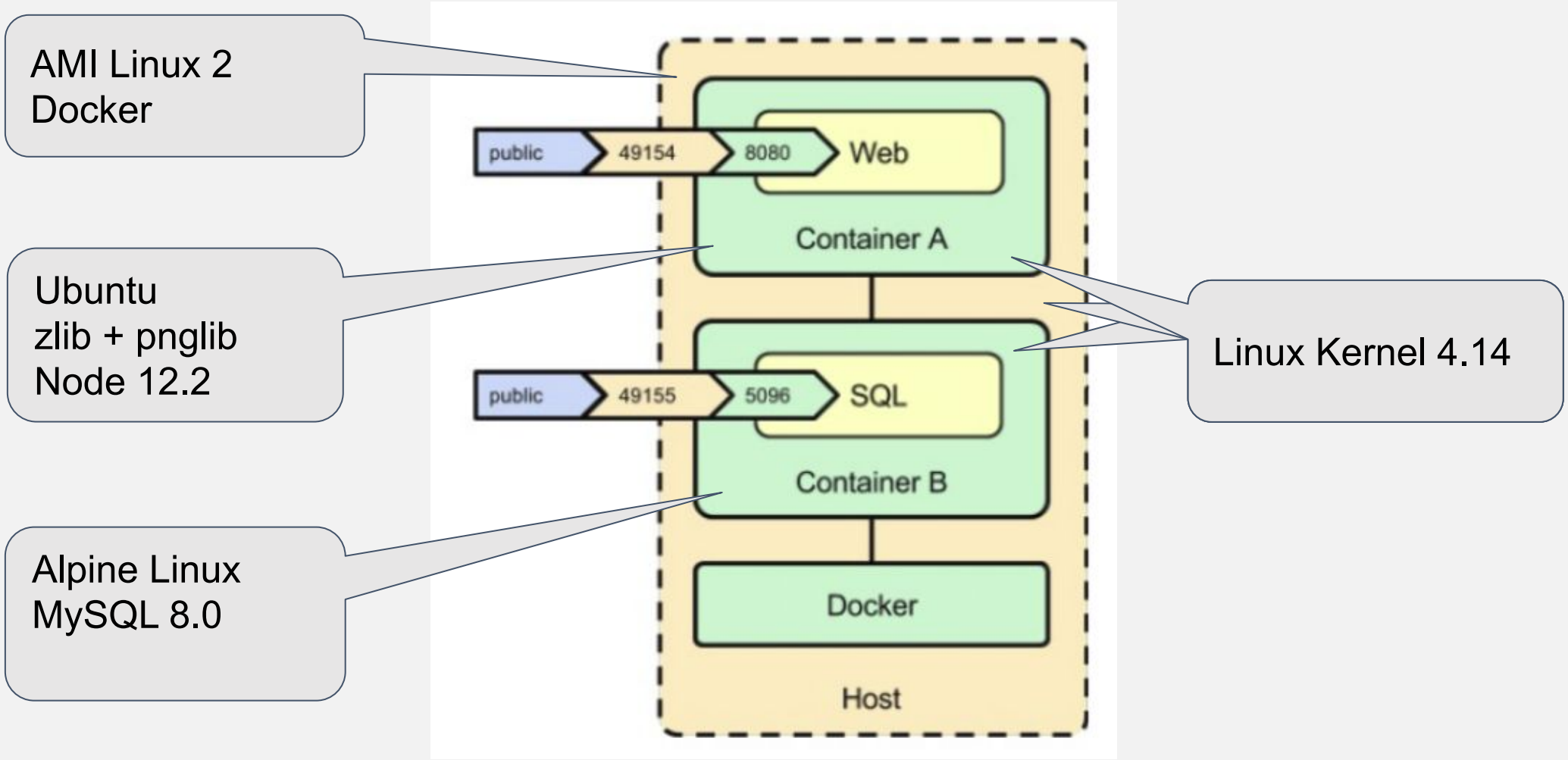
```
npm run serve -- --port 3000
```

Docker

O que é isso?!?!

- Alternativa à virtualização
- Cria um ambiente isolado para processos nativos
- Compartilhamento do Kernel Linux ou virtualização de um único Kernel Linux para containers no Mac/Windows
- Isolamento/compartilhamento entre containers:
  - PIDs, net, mnt, UTS, cgroups
- Sistema de arquivos em formato de união
- Cada declaração no Dockerfile irá gerar uma camada cacheada em cada build, acelerando os builds

O que é isso?!?!

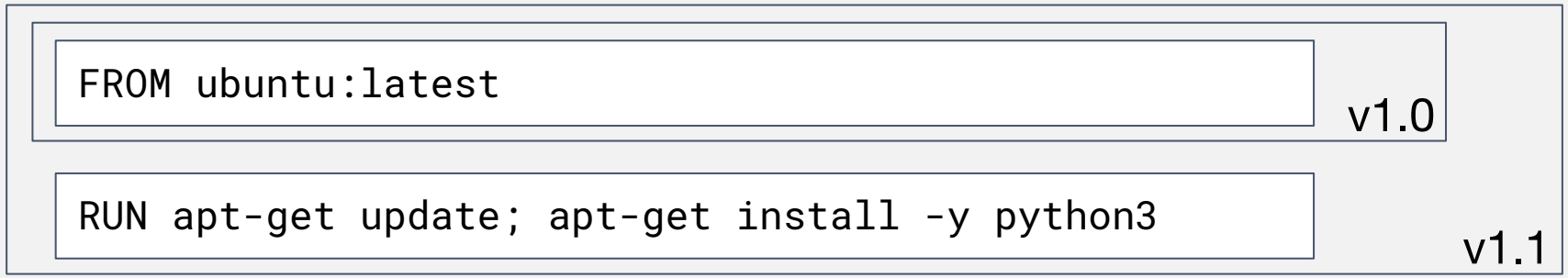

AMI Linux 2
Docker
Ubuntu
zlib + pnglib
Node 12.2
Alpine Linux
MySQL 8.0
public
49154
8080
Web
Container A
public
49155
5096
SQL
Container B
Docker
Host
Linux Kernel 4.14

Criando sua Imagem Docker

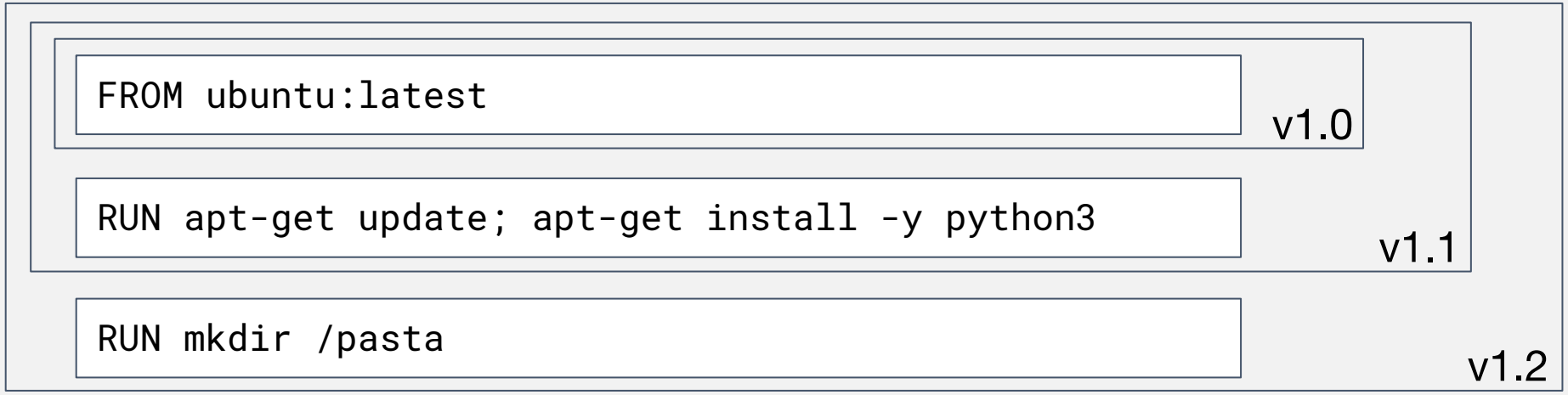
- Imagem é como uma “classe” e o container é o “objeto”
- Toda imagem precisa “herdar” uma imagem base
- Cópia de arquivos para dentro da imagem final
- Execução de comandos (instalações, configurações, etc)
- Comando base para iniciar o container (como um “construtor”)
- O build de uma imagem gera um tipo de “programa”
- Depois é só executar esse “programa”

Imagens Docker

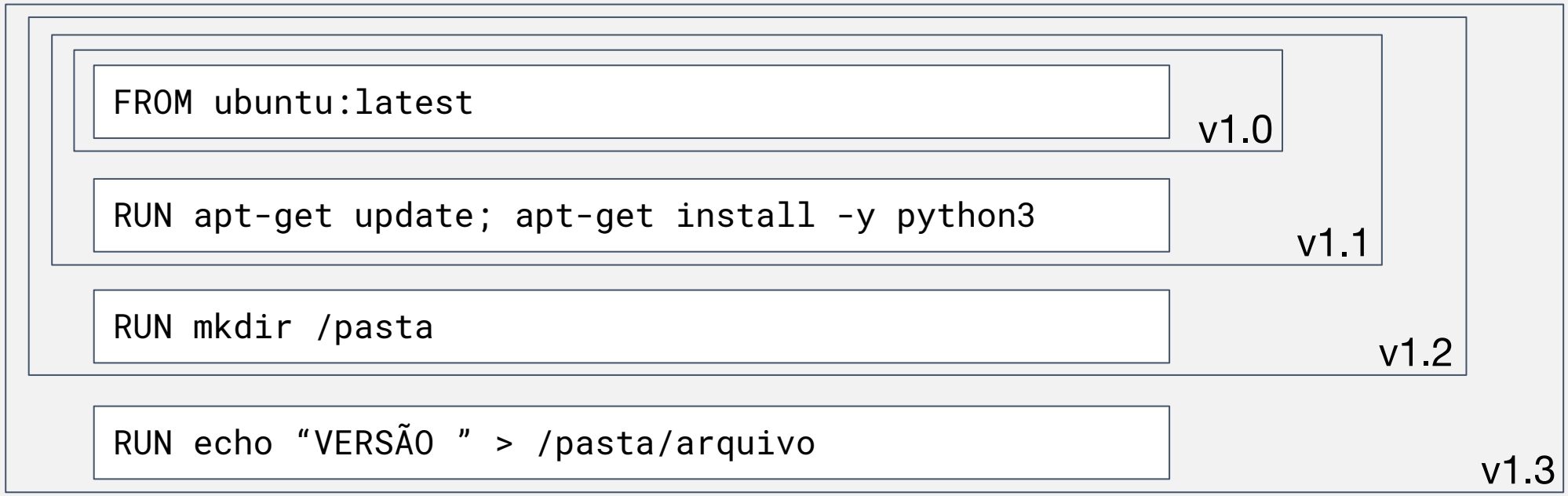
Imagens Docker



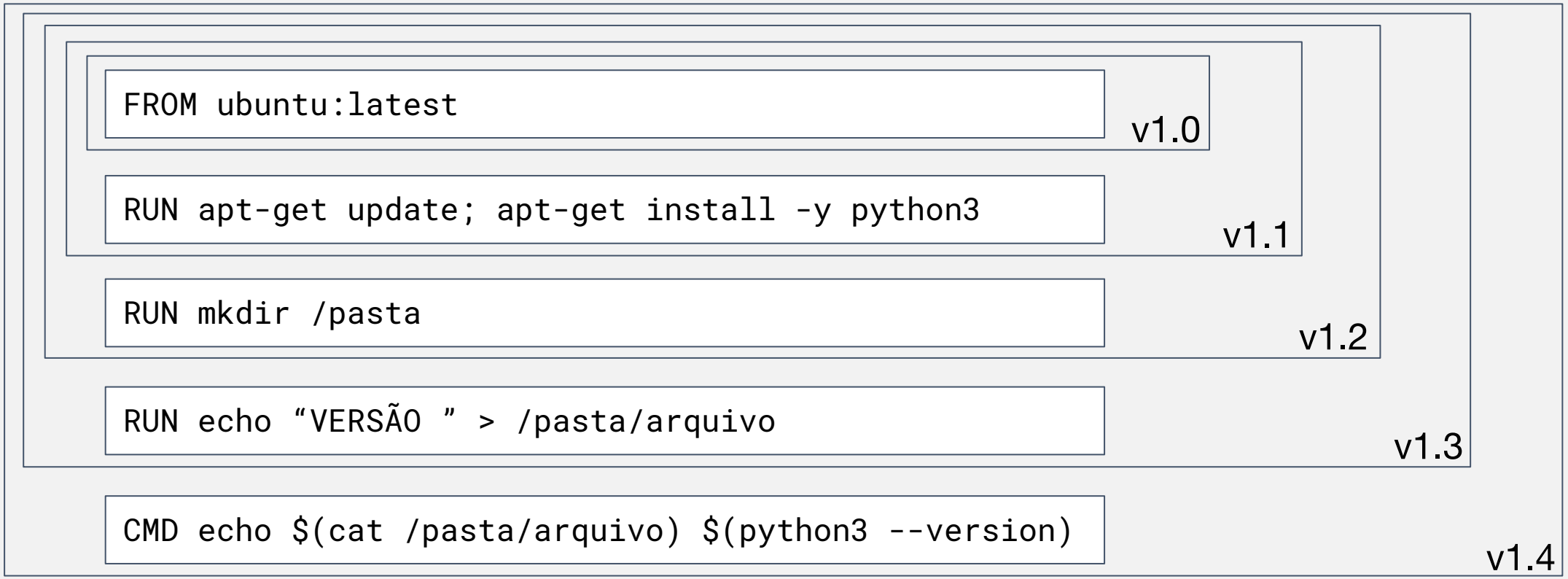
Imagens Docker



Imagens Docker



Imagens Docker



Prática

- Instalar Docker Community 19.03.0 ou superior
- Criar um Dockerfile para cada aplicação (backend e frontend)
- Seguir instruções a seguir para gerar os binários, definir a receita do docker, buildar e rodar

Prática - Backend

1. build do binrário

```
./mvnw clean package
```

2. receita para criar imagem do projeto

```
FROM adoptopenjdk/openjdk12

COPY target/*.jar ./backend.jar

CMD java -Dspring.data.mongodb.host=mongo -jar backend.jar
```

3. buildando imagem docker

```
sudo docker build -t backend .
```

4. rodando imagem docker

```
sudo docker run -it --rm -p 8080:8080 --link mongo:mongo backend
```

Prática - Frontend

1. build do binrário

```
npm run build
```

2. receita para criar imagem do projeto

```
FROM nginx


COPY dist/ /usr/share/nginx/html
```

3. buildando imagem docker

```
sudo docker build -t frontend .
```

4. rodando imagem docker

```
sudo docker run -it --rm -p 3000:80 frontend
```

hotmart.dev/vagas

Dúvidas?
Obrigado!

hotmart