**Finalizando o projeto Angular (FrontEnd)**

Abrindo e executando o projeto:

Samsung@DESKTOP-P9F6D9F MINGW64 /c/dev\_angular\_sqs/projeto\_angular (master)

**$ npm install**

Tela de celular

Descrição gerada automaticamente

Samsung@DESKTOP-P9F6D9F MINGW64 /c/dev\_angular\_sqs/projeto\_angular (master)

**$ ng s**

Texto

Descrição gerada automaticamente

<http://localhost:4200/>

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

Interface gráfica do usuário

Descrição gerada automaticamente

Samsung@DESKTOP-P9F6D9F MINGW64 /c/dev\_angular\_sqs/projeto\_angular (master)

**$ code .**

Tela de computador com jogo

Descrição gerada automaticamente

/app.component.ts

import { Component, OnInit } from '@angular/core';

import { FormGroup, FormControl } from '@angular/forms';

import { HttpClient } from '@angular/common/http';

import { environment } from 'src/environments/environment';

import { NgxSpinnerService } from "ngx-spinner";

@Component({

  selector: 'app-root',

  templateUrl: './app.component.html',

  styleUrls: ['./app.component.css']

})

export class AppComponent implements *OnInit* {

  mensagem\_autenticar: *string* = '';

  mensagem\_criarconta: *string* = '';

  mensagem\_recuperarsenha: *string* = '';

  isLoggedIn: *boolean* = false;

  nomeUsuario: *string* = '';

  emailUsuario: *string* = '';

  ngOnInit(): *void* {

    if (localStorage.getItem('dados-usuario') != null) {

      var dados = JSON.parse(localStorage.getItem('dados-usuario')

as *string*);

*this*.nomeUsuario = dados.usuario.nome;

*this*.emailUsuario = dados.usuario.email;

*this*.isLoggedIn = true;

    }

  }

  constructor(

    private *httpClient*: HttpClient,

    private *spinner*: NgxSpinnerService

  ) {

  }

  //criando um objeto para capturar os

  //campos do formulário de criação de usuário

  formCriarConta = **new** FormGroup({

    nome: **new** FormControl('', []), //campo 'nome'

    email: **new** FormControl('', []), //campo 'email'

    senha: **new** FormControl('', []) //campo 'senha'

  });

  //criando um objeto para capturar os

  //campos do formulário de autenticação

  formAutenticar = **new** FormGroup({

    email: **new** FormControl('', []), //campo 'email'

    senha: **new** FormControl('', []) //campo 'senha'

  });

  //criando um objeto para capturar os

  //campos do formulário de recuperação de senha

  formRecuperarSenha = **new** FormGroup({

    email: **new** FormControl('', []), //campo 'email'

  });

  //função para capturar o SUBMIT do formulário

  criarConta(): *void* {

*this*.spinner.show();

*this*.mensagem\_autenticar = '';

*this*.mensagem\_criarconta = '';

*this*.mensagem\_recuperarsenha = '';

    //capturando os valores preenchidos no formulário

    var dados = *this*.formCriarConta.value;

    //executando a chamada para a API

*this*.httpClient.post(environment.apiUsuarios

+ 'api/criar-conta', dados)

      .subscribe({ //capturando a resposta da API

        next: (*data*: *any*) => { //sucesso!

*this*.mensagem\_criarconta = *data*.mensagem;

*this*.formCriarConta.reset(); //limpar o formulário

        },

        error: (*e*) => { //erro!

          if (*e*.error.mensagem) {

*this*.mensagem\_criarconta = *e*.error.mensagem;

          }

          else if (*e*.error.errors) {

*this*.mensagem\_criarconta = *e*.error.errors;

          }

        }

      }).add(

        () => {

*this*.spinner.hide();

        }

      );

  }

  //função para capturar o SUBMIT do formulário

  autenticar(): *void* {

*this*.spinner.show();

*this*.mensagem\_autenticar = '';

*this*.mensagem\_criarconta = '';

*this*.mensagem\_recuperarsenha = '';

    //capturando os valores preenchidos no formulário

    var dados = *this*.formAutenticar.value;

    //executando a chamada para a API

*this*.httpClient.post(environment.apiUsuarios

+ 'api/autenticar', dados)

      .subscribe({ //capturando a resposta da API

        next: (*data*: *any*) => { //sucesso!

          localStorage.setItem('dados-usuario', JSON.stringify(*data*));

          window.location.href = "/admin-clientes";

        },

        error: (*e*) => { //erro!

*this*.mensagem\_autenticar = *e*.error.mensagem;

        }

      }).add(

        () => {

*this*.spinner.hide();

        }

      );

  }

  //função para capturar o SUBMIT do formulário

  recuperarSenha(): *void* {

*this*.spinner.show();

*this*.mensagem\_autenticar = '';

*this*.mensagem\_criarconta = '';

*this*.mensagem\_recuperarsenha = '';

    //capturando os valores preenchidos no formulário

    var dados = *this*.formRecuperarSenha.value;

    //executando a chamada para a API

*this*.httpClient.post(environment.apiUsuarios

+ 'api/recuperar-senha', dados)

      .subscribe({ //capturando a resposta da API

        next: (*data*: *any*) => { //sucesso!

*this*.mensagem\_recuperarsenha = data.mensagem;

        },

        error: (*e*) => { //erro!

*this*.mensagem\_recuperarsenha = e.error.mensagem;

        }

      }).add(

        () => {

*this*.spinner.hide();

        }

      );

  }

}

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

**Visualizando a local storage do navegador:**

Área utilizada para gravar as informações do usuário autenticado.

Interface gráfica do usuário, Texto

Descrição gerada automaticamente

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

/clientes.component.ts

Componente onde foi implementada a integração com a API de clientes.

import { Component, OnInit } from '@angular/core';

import { FormGroup, FormControl, Validators } from '@angular/forms';

import { HttpClient, HttpHeaders } from '@angular/common/http';

import { NgxSpinnerService } from 'ngx-spinner';

import { environment } from 'src/environments/environment';

@Component({

  selector: 'app-clientes',

  templateUrl: './clientes.component.html',

  styleUrls: ['./clientes.component.css']

})

export class ClientesComponent implements *OnInit* {

  isLoggedIn: *boolean* = false;

  nomeUsuario: *string* = '';

  emailUsuario: *string* = '';

  clientes: *any*[] = [];

  httpHeaders: HttpHeaders = **new** HttpHeaders();

  mensagem\_cadastro: *string* = '';

  mensagem\_exclusao: *string* = '';

  mensagem\_edicao: *string* = '';

  constructor(

    private *httpClient*: HttpClient,

    private *spinner*: NgxSpinnerService

  ) {

  }

  formCadastro = **new** FormGroup({

    nome: **new** FormControl('', []),

    email: **new** FormControl('', []),

    telefone: **new** FormControl('', []),

    cpf: **new** FormControl('', [])

  });

  formEdicao = **new** FormGroup({

    idCliente: **new** FormControl('', []),

    nome: **new** FormControl('', []),

    email: **new** FormControl('', []),

    telefone: **new** FormControl('', []),

    cpf: **new** FormControl('', [])

  });

  ngOnInit(): *void* {

    if (localStorage.getItem('dados-usuario') != null) {

      var dados = JSON.parse(localStorage.getItem('dados-usuario')

as *string*);

*this*.nomeUsuario = dados.usuario.nome;

*this*.emailUsuario = dados.usuario.email;

*this*.isLoggedIn = true;

*this*.httpHeaders = **new** HttpHeaders({

        'Authorization': `Bearer ${dados.token}`

      })

*this*.consultarClientes();

    }

    else {

      window.location.href = '';

    }

  }

  consultarClientes(): *void* {

*this*.spinner.show();

*this*.httpClient.get(environment.apiClientes + 'api/clientes', {

      headers: *this*.httpHeaders

    })

      .subscribe(

        (*data*) => {

*this*.clientes = data as *any*[];

        }

      ).add(

        () => {

*this*.spinner.hide();

        }

      );

  }

  cadastrarCliente(): *void* {

*this*.spinner.show();

*this*.httpClient.post(environment.apiClientes + 'api/clientes',

*this*.formCadastro.value, { headers: *this*.httpHeaders })

      .subscribe({

        next: (*data*: *any*) => {

*this*.mensagem\_cadastro = data.mensagem;

*this*.formCadastro.reset();

*this*.consultarClientes();

        },

        error: (*e*) => {

*this*.mensagem\_cadastro = e.error.mensagem;

        }

      }).add(

        () => {

*this*.spinner.hide();

        }

      );

  }

  excluirCliente(*idCliente*: *number*): *void* {

    if (window.confirm('Deseja realmente excluir

o cliente selecionado?')) {

*this*.spinner.show();

*this*.httpClient.delete(environment.apiClientes

+ 'api/clientes/' + idCliente,

        { headers: *this*.httpHeaders })

        .subscribe({

          next: (*data*: *any*) => {

*this*.mensagem\_exclusao = data.mensagem;

*this*.consultarClientes();

          },

          error: (*e*) => {

*this*.mensagem\_exclusao = e.error.mensagem;

          }

        }).add(

          () => {

*this*.spinner.hide();

          }

        );

    }

  }

  obterCliente(*idCliente*: *number*): *void* {

*this*.spinner.show();

*this*.httpClient.get(environment.apiClientes

+ 'api/clientes/' + idCliente,

      { headers: *this*.httpHeaders })

      .subscribe({

        next: (*data*: *any*) => {

*this*.formEdicao.patchValue(data);

        },

        error: (*e*) => {

          console.log(e.error);

        }

      }).add(

        () => {

*this*.spinner.hide();

        }

      );

  }

  atualizarCliente(): *void* {

*this*.spinner.show();

*this*.httpClient.put(environment.apiClientes + 'api/clientes/',

*this*.formEdicao.value, { headers: *this*.httpHeaders })

      .subscribe({

        next: (*data*: *any*) => {

*this*.mensagem\_edicao = data.mensagem;

*this*.consultarClientes();

        },

        error: (*e*) => {

          console.log(e.error);

        }

      }).add(

        () => {

*this*.spinner.hide();

        }

      );

  }

  logout(): *void* {

    if (window.confirm('Deseja realmente sair do sistema?')) {

      localStorage.removeItem('dados-usuario');

      window.location.href = '';

    }

  }

}

Interface gráfica do usuário, Texto, Aplicativo, Tabela, Email

Descrição gerada automaticamente

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

**Deploy dos projetos Angular:**

Realizando o processo de BUILD (compilar o projeto,   
gerando os arquivos para publicação em servidor).

Samsung@DESKTOP-P9F6D9F MINGW64 /c/dev\_angular\_sqs/projeto\_angular (master)

**$ ng build**

Texto

Descrição gerada automaticamente

Os arquivos gerados pelo DEPLOY ficam dentro da pasta **/dist**

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

**Conteúdo gerado:**

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

.htaccess

Arquivo de configuração para corrigir problemas de URLs ou links “quebrados” na navegação dos sites.

Interface gráfica do usuário, Texto

Descrição gerada automaticamente

**Pasta para publicação no AWS:**

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

**AWS S3**

Serviço para publicação de sites estáticos, ou seja, aplicações que só possuem arquivos HTML, CSS e JS.



Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamente

Selecione a opção:

**Criar Bucket**

Interface gráfica do usuário, Texto

Descrição gerada automaticamente

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

Interface gráfica do usuário, Texto, Aplicativo, Word

Descrição gerada automaticamente

Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamente

Interface gráfica do usuário, Texto

Descrição gerada automaticamente

**Fazendo o upload dos arquivos:**

Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamente

Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamente

**Selecione e envie os arquivos de BUILD:**

Interface gráfica do usuário

Descrição gerada automaticamente

Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamente

Interface gráfica do usuário, Texto, Aplicativo, Word

Descrição gerada automaticamente

**Arquivos enviados com sucesso:**

Interface gráfica do usuário, Aplicativo, Tabela, Excel

Descrição gerada automaticamente

**Configurando o Bucket como um website:**

Propriedades

Interface gráfica do usuário, Texto, Aplicativo, Tabela

Descrição gerada automaticamente

Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamente

Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamente

Interface gráfica do usuário, Texto, Aplicativo, Word

Descrição gerada automaticamente

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

**Configurando o WebSite:**

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

**{**

**"Version": "2012-10-17",**

**"Statement": [**

**{**

**"Sid": "Statement1",**

**"Principal": "\*",**

**"Effect": "Allow",**

**"Action": "s3:GetObject",**

**"Resource": "arn:aws:s3:::projetoangular/\*"**

**}**

**]**

**}**

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

**Status do Site:**

Publicamente acessível

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

**Obtendo o endereço do site:**

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

**Projeto Angular:**

<http://projetoangular.s3-website.us-east-2.amazonaws.com/>

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

**Instalando o DOCKER DESKTOP:**

<https://docs.docker.com/desktop/install/windows-install/>

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

Logotipo

Descrição gerada automaticamente

Docker é uma plataforma aberta, criada com o objetivo de facilitar o desenvolvimento, a implantação e a execução de aplicações em ambientes isolados. Foi desenhada especialmente para disponibilizar uma aplicação da forma mais rápida possível.

Ícone

Descrição gerada automaticamente

Usando o Docker, você pode facilmente gerenciar a infraestrutura da aplicação, isso agilizará o processo de criação, manutenção e modificação do seu serviço.

Todo processo é realizado sem necessidade de qualquer acesso privilegiado à infraestrutura corporativa.

Assim, a equipe responsável pela aplicação pode participar da especificação do ambiente junto com a equipe responsável pelos servidores.

O Docker viabilizou uma "linguagem" comum entre desenvolvedores e administradores de servidores. Essa nova "linguagem" é utilizada para construir arquivos com as definições da infraestrutura necessária e como a aplicação será disposta nesse ambiente, em qual porta fornecerá seu serviço, quais dados de volumes externos serão requisitados e outras possíveis necessidades.

O Docker utiliza o modelo de container para “empacotar” a aplicação que, após ser transformada em imagem Docker, pode ser reproduzida em plataforma de qualquer porte; ou seja, caso a aplicação funcione sem falhas em seu notebook, funcionará também no servidor ou no mainframe. Construa uma vez, execute onde quiser.

Os containers são isolados a nível de disco, memória, processamento e rede. Essa separação permite grande flexibilidade, onde ambientes distintos podem coexistir no mesmo host, sem causar qualquer problema. Vale salientar que o overhead nesse processo é o mínimo necessário, pois cada container normalmente carrega apenas um processo, que é aquele responsável pela entrega do serviço desejado.

Em todo caso, esse container também carrega todos os arquivos necessários (configuração, biblioteca e afins) para execução completamente isolada.

O modelo de isolamento utilizado no Docker é a virtualização a nível do sistema operacional, um método de virtualização onde o kernel do sistema operacional permite que múltiplos processos sejam executados isoladamente no mesmo host.

Esses processos isolados em execução são denominados no Docker de container.

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

Para criar o isolamento necessário do processo, o Docker usa a funcionalidade do kernel, denominada de [namespaces](http://man7.org/linux/man-pages/man7/namespaces.7.html), que cria ambientes isolados entre containers: os processos de uma aplicação em execução não terão acesso aos recursos de outra.

A menos que seja expressamente liberado na configuração de cada ambiente.

Para evitar a exaustão dos recursos da máquina por apenas um ambiente isolado, o Docker usa a funcionalidade [cgroups](https://en.wikipedia.org/wiki/Cgroups) do kernel, responsável por criar limites de uso do hardware a disposição.

Com isso é possível coexistir no mesmo host diferentes containers sem que um afete diretamente o outro por uso exagerado dos recursos compartilhados.

**Criando um projeto no eclipse:**

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

Interface gráfica do usuário, Texto, Aplicativo, Word

Descrição gerada automaticamente

/pom.xml

* Adicionando as bibliotecas do Swagger

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0

https://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>2.7.9</version>

<relativePath/> <!-- lookup parent from repository -->

</parent>

<groupId>br.com.cotiinformatica</groupId>

<artifactId>api\_exemplodocker</artifactId>

<version>1.0</version>

<name>api\_exemplodocker</name>

<description>API exemplo Docker</description>

<properties>

<java.version>11</java.version>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-devtools</artifactId>

<scope>runtime</scope>

<optional>true</optional>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

</dependency>

**<dependency>**

**<groupId>io.springfox</groupId>**

**<artifactId>springfox-swagger-ui</artifactId>**

**<version>3.0.0</version>**

**</dependency>**

**<dependency>**

**<groupId>io.springfox</groupId>**

**<artifactId>springfox-boot-starter</artifactId>**

**<version>3.0.0</version>**

**</dependency>**

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

</project>

* Adicionar uma configuração para que o projeto possa ser executado em um container DOCKER.
  + Vamos usar um PLUGIN desenvolvido pela equipe do spotify para configurar um projeto Java fazendo com que possamos executar o projeto dentro de um container DOCKER.

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>2.7.9</version>

<relativePath/> <!-- lookup parent from repository -->

</parent>

<groupId>br.com.cotiinformatica</groupId>

<artifactId>api\_exemplodocker</artifactId>

<version>1.0</version>

<name>api\_exemplodocker</name>

<description>API exemplo Docker</description>

<properties>

<java.version>11</java.version>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-devtools</artifactId>

<scope>runtime</scope>

<optional>true</optional>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

</dependency>

<dependency>

<groupId>io.springfox</groupId>

<artifactId>springfox-swagger-ui</artifactId>

<version>3.0.0</version>

</dependency>

<dependency>

<groupId>io.springfox</groupId>

<artifactId>springfox-boot-starter</artifactId>

<version>3.0.0</version>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

**<plugin>**

**<groupId>com.spotify</groupId>**

**<artifactId>dockerfile-maven-plugin</artifactId>**

**<version>1.4.13</version>**

**<executions>**

**<execution>**

**<id>default</id>**

**<goals>**

**<goal>build</goal>**

**<goal>push</goal>**

**</goals>**

**</execution>**

**</executions>**

**<configuration>**

**<useMavenSettingsForAuth>**

**true**

**</useMavenSettingsForAuth>**

**<repository>**

**${project.artifactId}**

**</repository>**

**<tag>${project.version}</tag>**

**<buildArgs>**

**<JAR\_FILE>**

**${project.build.finalName}.jar**

**</JAR\_FILE>**

**</buildArgs>**

**</configuration>**

**</plugin>**

</plugins>

</build>

</project>

Criando uma classe para configurar a biblioteca do Swagger:

/config/SwaggerConfig.java

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

**package** br.com.cotiinformatica.config;

**import** java.util.Collections;

**import** org.springframework.context.annotation.Bean;

**import** org.springframework.context.annotation.Configuration;

**import** org.springframework.web.servlet.config.annotation.EnableWebMvc;

**import** springfox.documentation.builders.PathSelectors;

**import** springfox.documentation.builders.RequestHandlerSelectors;

**import** springfox.documentation.service.ApiInfo;

**import** springfox.documentation.service.Contact;

**import** springfox.documentation.spi.DocumentationType;

**import** springfox.documentation.spring.web.plugins.Docket;

**import** springfox.documentation.swagger2.annotations.EnableSwagger2;

@Configuration

@EnableWebMvc

@EnableSwagger2

**public** **class** SwaggerConfig {

@Bean

**public** Docket api() {

**return** **new** Docket(DocumentationType.***SWAGGER\_2***)

.select()

.apis(RequestHandlerSelectors.*basePackage*

("br.com.cotiinformatica"))

.paths(PathSelectors.*ant*("/\*\*"))

.build()

.apiInfo(apiInfo());

}

**private** ApiInfo apiInfo() {

**return** **new** ApiInfo(

"API exemplo - Docker",

"Sistema Spring Boot API",

"Versão 1.0",

"http://www.cotiinformatica.com.br",

**new** Contact("COTI Informática",

"http://www.cotiinformatica.com.br", "contato@cotiinformatica.com.br"),

"Licença da API",

"http://www.cotiinformatica.com.br",

Collections.*emptyList*()

);

}

}

**Criando um controlador para gerar um ENDPOINT na API:**

/controllers/HelloWorldController.java

**package** br.com.cotiinformatica.controllers;

**import** org.springframework.web.bind.annotation.GetMapping;

**import** org.springframework.web.bind.annotation.RestController;

**import** io.swagger.annotations.Api;

**import** io.swagger.annotations.ApiOperation;

@Api(tags = "Hello World!")

@RestController

**public** **class** HelloWorldController {

@ApiOperation("ENDPOINT para teste de mensagem.")

@GetMapping("/api/helloworld")

**public** String getMensagem() {

**return** "Olá, eu sou uma API!";

}

}

/application.properties

Configurando a porta para execução do projeto.

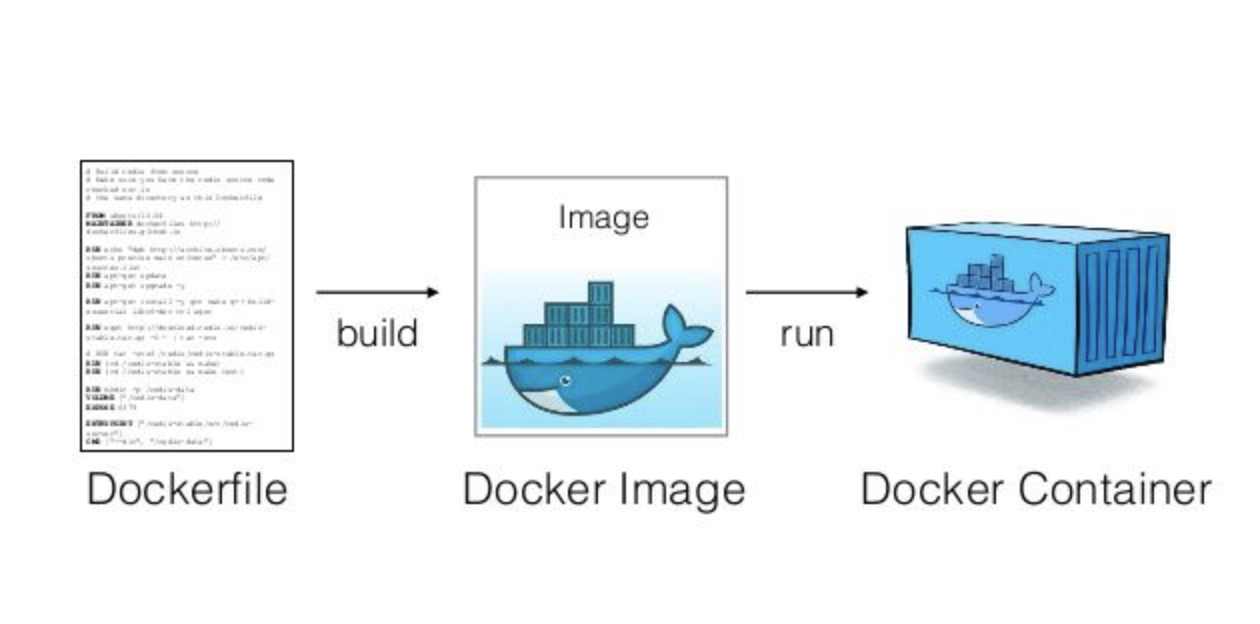
**server.port=8084**

Interface gráfica do usuário, Aplicativo, Word

Descrição gerada automaticamente

Precisamos criar no projeto um arquivo de configuração para definir todo o ambiente que deverá criado no DOCKER para rodar o projeto.

Este arquivo irá configurar a **IMAGEM** do **CONTAINER** onde o projeto será executado. Essa imagem define todas as bibliotecas ou configurações para que o projeto seja executado.



Interface gráfica do usuário, Aplicativo, Word

Descrição gerada automaticamente

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

**FROM** openjdk:11-jre

**RUN** mkdir app

**ARG** JAR\_FILE

**ADD** /target/${JAR\_FILE} /app/api\_exemplodocker.jar

**EXPOSE** 8084

**WORKDIR** /app

**ENTRYPOINT** java -jar api\_exemplodocker.jar

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

**Fazendo o processo de compilação do projeto:**

C:\java\_dev\api\_exemplodocker>**mvnw clean package**

Texto

Descrição gerada automaticamente

**As vantagens do uso do Docker**

* O Docker permite um uso mais eficiente dos recursos do sistema;
* As instâncias de aplicativos em container usam muito menos memória do que as máquinas virtuais, elas são inicializadas e interrompidas mais rapidamente e podem ser armazenadas muito mais densamente em um hardware host. Tudo isso equivale a menos gastos com TI;
* A redução de custos irá variar dependendo de quais aplicativos estão em jogo e de quão intensivos os recursos podem ser, mas os containers funcionam invariavelmente como mais eficientes que as VMs.