**Read.me**

Projeto Devops

A aplicação é um CRUD em java com Spring Boot e Banco de Dados H2.

Foram definidas as seguintes APIs:

|  |  |  |
| --- | --- | --- |
| **Métodos** | **Urls** | **Ações** |
| POST | /api/documentos | Cria um novo documento |
| GET | /api/documentos | Mostra todos os documentos |
| GET | /api/documentos/:id | Retorna um documento pelo :id |
| PUT | /api/documentos/:id | Atualiza um documento pelo :id |
| DELETE | /api/documentos/:id | Deleta um documento pelo :id |
| DELETE | /api/documentos | Deleta todos os documentos |
| GET | /api/documentos/publicados | Mostra todos os documentos publicados |
| GET | /api/documentos?titulo=[título doc] | Mostra o documento que contém o título informado |

**Print da estrutura do projeto**

Tela de computador

Descrição gerada automaticamente

**Registros incluídos pelo Postman**

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

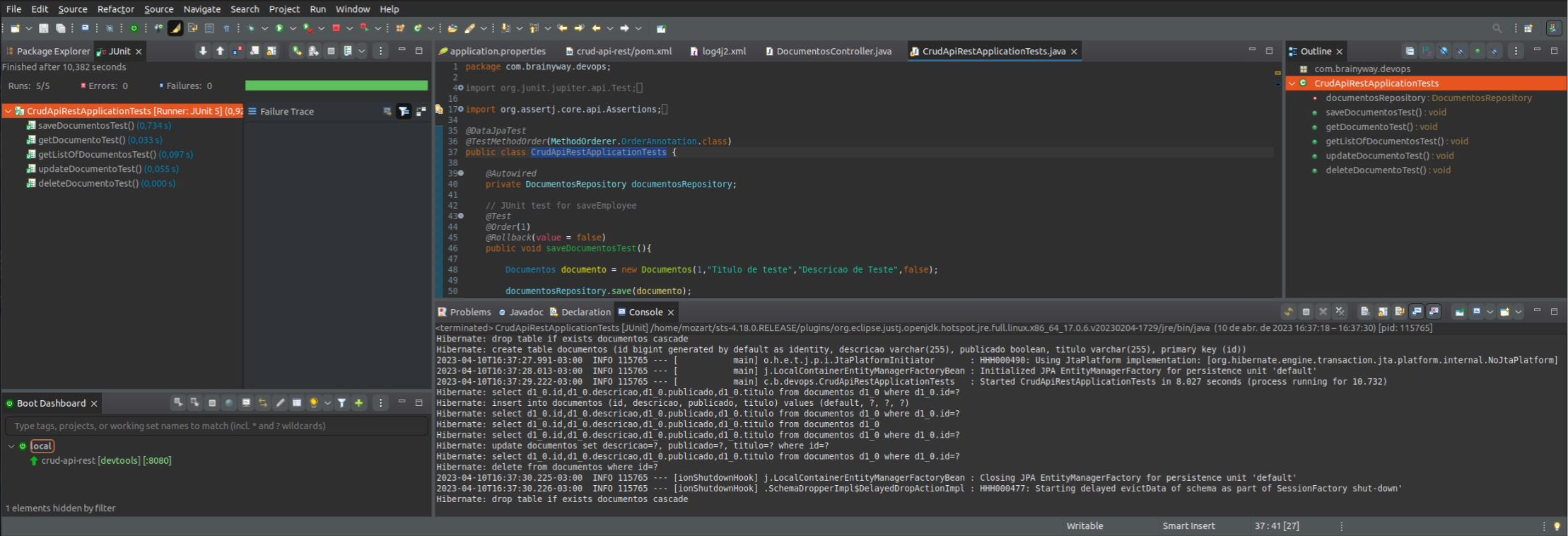
Texto

Descrição gerada automaticamente

**Testes com JUNIT**

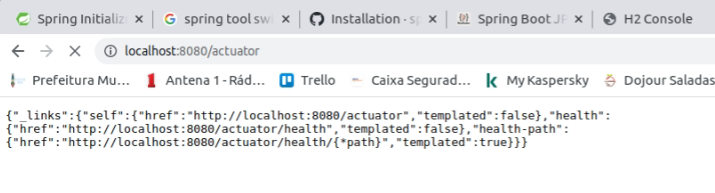
Foi criada uma classe para os testes com o JUNIT. A classe CrudApiRestApplicationTests. A seguir está o resultado dos testes

Print dos testes com JUNIT



**Monitoramento com o Actuator**

O actuator foi configurado e acessado pelo endereço: <http://localhost:8080/actuator> e <http://localhost:8080/actuator/health>



Interface gráfica do usuário, Texto, Aplicativo, chat ou mensagem de texto

Descrição gerada automaticamente

**Configurando o Log4j**

O Log4j foi configurado para mostrar os logs na console e no arquivo crud-api-rest.log conforme tela a seguir:

Interface gráfica do usuário, Texto

Descrição gerada automaticamente

Publicação do projeto no Github

O projeto foi publicado no Github e pode ser acessado pelo link: <https://github.com/mozartclaret/crud-api-rest>

Interface gráfica do usuário, Texto, Email

Descrição gerada automaticamente

**Monitoramento com o Micrometer**

Para acompanhar a performance da aplicação foi instalado o micrometer. Porém, ao ser incluída a configuração no arquivo application.properties, acontece um erro no logback que para de reconhecer as tags <Appenders> e <Loggers>

As configurações incluídas foram:

Inclusão da dependência no pom.xml

<dependency>

<groupId>io.micrometer</groupId>

<artifactId>micrometer-core</artifactId>

<version>1.7.3</version>

</dependency>

<dependency>

<groupId>io.micrometer</groupId>

<artifactId>micrometer-registry-prometheus</artifactId>

<version>1.7.3</version>

</dependency>

Inclusão dos parâmetros em application.properties:

# Configure o registro para o Prometheus

management.metrics.export.prometheus.enabled=true

# Configure o nome da aplicação

management.metrics.tags.application=crud-api-rest

**Configurando o zipkin**

A configuração do zipkin é feita adicionando-se e dependência no pom.xml e depois atualizando o application.properties. Porém, de igual maneira, acontece um erro no logback que para de reconhecer as tags <Appenders> e <Loggers>

Os parâmetros adicionados em application.properties foram:

spring.zipkin.base-url=http://localhost:9411

spring.sleuth.sampler.probability=1.0

**Terraform**

Foi criada uma instância de um computador com Ubuntu na amazon utilizando o Terraform, corforma o processo abaixo:

mozart@Devmachine:~/devops-terraform-aws-instance$ sudo curl "https://awscli.amazonaws.com/awscli-exe-linux-x86\_64.zip" -o "awscliv2.zip"

mozart@Devmachine:~/devops-terraform-aws-instance$ unzip awscliv2.zip

You can now run: /usr/local/bin/aws --version

mozart@Devmachine:~$ aws --version

aws-cli/2.11.11 Python/3.11.2 Linux/5.19.0-38-generic exe/x86\_64.ubuntu.22 prompt/off

mozart@Devmachine:~/devops-terraform-aws-instance$ export AWS\_ACCESS\_KEY\_ID=AKIA22UVXAIM6QFC5TW3

mozart@Devmachine:~/devops-terraform-aws-instance$ export AWS\_SECRET\_ACCESS\_KEY=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

mozart@Devmachine:~$ mkdir devops-terraform-aws-instance

mozart@Devmachine:~$ cd devops-terraform-aws-instance/

mozart@Devmachine:~/devops-terraform-aws-instance$ touch main.tf

mozart@Devmachine:~/devops-terraform-aws-instance$ terraform init

Initializing the backend...

Initializing provider plugins...

- Finding hashicorp/aws versions matching "~> 4.16"...

- Installing hashicorp/aws v4.62.0...

- Installed hashicorp/aws v4.62.0 (signed by HashiCorp)

Terraform has created a lock file .terraform.lock.hcl to record the provider

selections it made above. Include this file in your version control repository

so that Terraform can guarantee to make the same selections by default when

you run "terraform init" in the future.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see

any changes that are required for your infrastructure. All Terraform commands

should now work.

If you ever set or change modules or backend configuration for Terraform,

rerun this command to reinitialize your working directory. If you forget, other

commands will detect it and remind you to do so if necessary.

mozart@Devmachine:~/devops-terraform-aws-instance$ terraform fmt

main.tf

mozart@Devmachine:~/devops-terraform-aws-instance$ terraform validate

Success! The configuration is valid.

mozart@Devmachine:~/devops-terraform-aws-instance$ terraform apply

mozart@Devmachine:~/devops-terraform-aws-instance$ terraform apply

Terraform used the selected providers to generate the following execution plan.

Resource actions are indicated with the following symbols:

+ create

Terraform will perform the following actions:

# aws\_instance.app\_server will be created

+ resource "aws\_instance" "app\_server" {

+ ami = "ami-830c94e3"

+ arn = (known after apply)

+ associate\_public\_ip\_address = (known after apply)

+ availability\_zone = (known after apply)

+ cpu\_core\_count = (known after apply)

+ cpu\_threads\_per\_core = (known after apply)

+ disable\_api\_stop = (known after apply)

+ disable\_api\_termination = (known after apply)

+ ebs\_optimized = (known after apply)

+ get\_password\_data = false

+ host\_id = (known after apply)

+ host\_resource\_group\_arn = (known after apply)

+ iam\_instance\_profile = (known after apply)

+ id = (known after apply)

+ instance\_initiated\_shutdown\_behavior = (known after apply)

+ instance\_state = (known after apply)

+ instance\_type = "t2.micro"

+ ipv6\_address\_count = (known after apply)

+ ipv6\_addresses = (known after apply)

+ key\_name = (known after apply)

+ monitoring = (known after apply)

+ outpost\_arn = (known after apply)

+ password\_data = (known after apply)

+ placement\_group = (known after apply)

+ placement\_partition\_number = (known after apply)

+ primary\_network\_interface\_id = (known after apply)

+ private\_dns = (known after apply)

+ private\_ip = (known after apply)

+ public\_dns = (known after apply)

+ public\_ip = (known after apply)

+ secondary\_private\_ips = (known after apply)

+ security\_groups = (known after apply)

+ source\_dest\_check = true

+ subnet\_id = (known after apply)

+ tags = {

+ "Name" = "ExampleAppServerInstance"

}

+ tags\_all = {

+ "Name" = "ExampleAppServerInstance"

}

+ tenancy = (known after apply)

+ user\_data = (known after apply)

+ user\_data\_base64 = (known after apply)

+ user\_data\_replace\_on\_change = false

+ vpc\_security\_group\_ids = (known after apply)

}

Plan: 1 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?

Terraform will perform the actions described above.

Only 'yes' will be accepted to approve.

Enter a value: yes

aws\_instance.app\_server: Creating...

aws\_instance.app\_server: Still creating... [10s elapsed]

aws\_instance.app\_server: Still creating... [20s elapsed]

aws\_instance.app\_server: Still creating... [30s elapsed]

aws\_instance.app\_server: Still creating... [40s elapsed]

aws\_instance.app\_server: Still creating... [50s elapsed]

aws\_instance.app\_server: Still creating... [1m0s elapsed]

aws\_instance.app\_server: Still creating... [1m10s elapsed]

aws\_instance.app\_server: Creation complete after 1m16s [id=i-0878d5d5cd2e37550]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.

mozart@Devmachine:~/devops-terraform-aws-instance$ terraform show

# aws\_instance.app\_server:

resource "aws\_instance" "app\_server" {

ami = "ami-830c94e3"

arn = "arn:aws:ec2:us-west-2:744417067545:instance/i-0878d5d5cd2e37550"

associate\_public\_ip\_address = true

availability\_zone = "us-west-2a"

cpu\_core\_count = 1

cpu\_threads\_per\_core = 1

disable\_api\_stop = false

disable\_api\_termination = false

ebs\_optimized = false

get\_password\_data = false

hibernation = false

id = "i-0878d5d5cd2e37550"

instance\_initiated\_shutdown\_behavior = "stop"

instance\_state = "running"

instance\_type = "t2.micro"

ipv6\_address\_count = 0

ipv6\_addresses = []

monitoring = false

placement\_partition\_number = 0

primary\_network\_interface\_id = "eni-0747c6c5a897774fa"

private\_dns = "ip-172-31-17-33.us-west-2.compute.internal"

private\_ip = "172.31.17.33"

public\_dns = "ec2-34-213-117-211.us-west-2.compute.amazonaws.com"

public\_ip = "34.213.117.211"

secondary\_private\_ips = []

security\_groups = [

"default",

]

source\_dest\_check = true

subnet\_id = "subnet-0d65f66283b77a6a7"

tags = {

"Name" = "ExampleAppServerInstance"

}

tags\_all = {

"Name" = "ExampleAppServerInstance"

}

tenancy = "default"

user\_data\_replace\_on\_change = false

vpc\_security\_group\_ids = [

"sg-09fffb280e0522cee",

]

capacity\_reservation\_specification {

capacity\_reservation\_preference = "open"

}

credit\_specification {

cpu\_credits = "standard"

}

enclave\_options {

enabled = false

}

maintenance\_options {

auto\_recovery = "default"

}

metadata\_options {

http\_endpoint = "enabled"

http\_put\_response\_hop\_limit = 1

http\_tokens = "optional"

instance\_metadata\_tags = "disabled"

}

private\_dns\_name\_options {

enable\_resource\_name\_dns\_a\_record = false

enable\_resource\_name\_dns\_aaaa\_record = false

hostname\_type = "ip-name"

}

root\_block\_device {

delete\_on\_termination = true

device\_name = "/dev/sda1"

encrypted = false

iops = 0

tags = {}

throughput = 0

volume\_id = "vol-0d481fb1e5a2ec3a1"

volume\_size = 8

volume\_type = "standard"

}

}

mozart@Devmachine:~/devops-terraform-aws-instance$ terraform state list

aws\_instance.app\_server

mozart@Devmachine:~/devops-terraform-aws-instance$

Instância criada – Console EC2

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

Pipeline no dev.azure.com

# Trigger

trigger:

branches:

include:

- master

# Pipeline

pool:

vmImage: Ubuntu-20.04

# Stage

stages:

- stage: DeployTerraform

displayName: Deploy Terraform

jobs:

- job: DeployTerraform

displayName: 'Deploy infrastructure with Terraform'

# Steps

steps:

- task: Bash@3

displayName: "Terraform Install"

name: "terraform\_install"

inputs:

targetType: inline

script: "echo \"# Downloading Terraform v1.3.7\"\nwget https://releases.hashicorp.com/terraform/1.3.7/terraform\_1.3.8\_linux\_amd64.zip\necho \"# Unzipping files\"\nunzip terraform\_1.3.8\_linux\_amd64.zip\n \necho \"# Configuring terraform and checking version\"\nexport TERRAFORM\_PATH=$(which terraform)\nls -lha\nsudo mv terraform $TERRAFORM\_PATH\n\nterraform -v\n"

- task: Bash@3

displayName: 'Deploy Service'

inputs:

targetType: 'inline'

Script: |

export AWS\_ACCESS\_KEY\_ID=744417067545

export AWS\_SECRET\_ACCESS\_KEY=f6966ef8a6e558577d386745f48c570d3108c22c7142ec8073ed9b0d90031198

mkdir devops-terraform-aws-instance

cd devops-terraform-aws-instance/

touch main.tf

terraform init

terraform fmt

terraform validate

terraform apply

Pipeline agendado no Azure Devops

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente