



## Desenvolvendo serviços de mensageria com RabbitMQ

### O que é RabbitMQ?

RabbitMQ é um servidor de mensageria de código aberto (open source), que faz uso do protocolo AMQP (Advanced Message Queuing Protocol). O rabbit é compatível com muitas linguagens de programação e permite lidar com o tráfego de mensagens de forma simples e confiável. Vale falar que também possui uma interface de administração nativa e é multiplataforma.

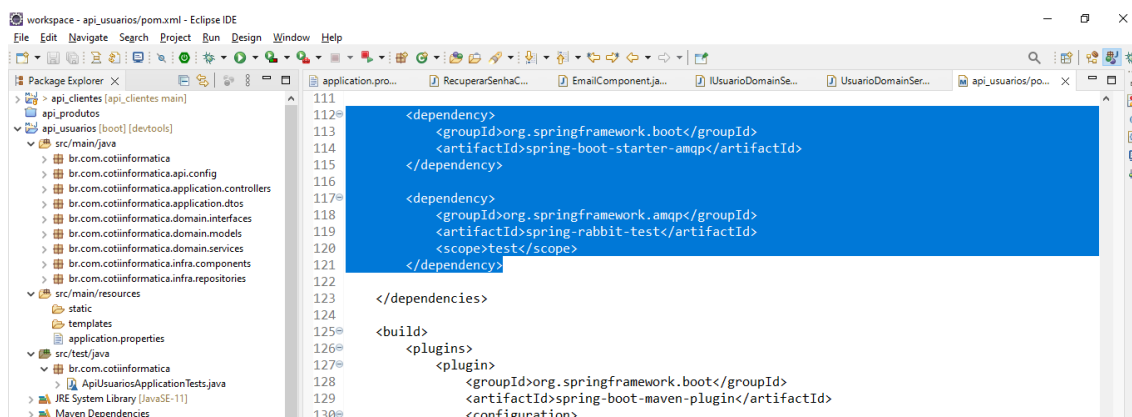
Um message broker é um sistema que permite que diferentes componentes, como aplicações e aplicativos, se comuniquem entre si, trocando informações. Para isso, geralmente utilizam uma estrutura de fila de mensagens, que será a responsável por armazenar e ordenar mensagens enquanto os consumidores (como aplicativos ou aplicações) não as processam.

### /pom.xml

#### Instalando RabbitMQ

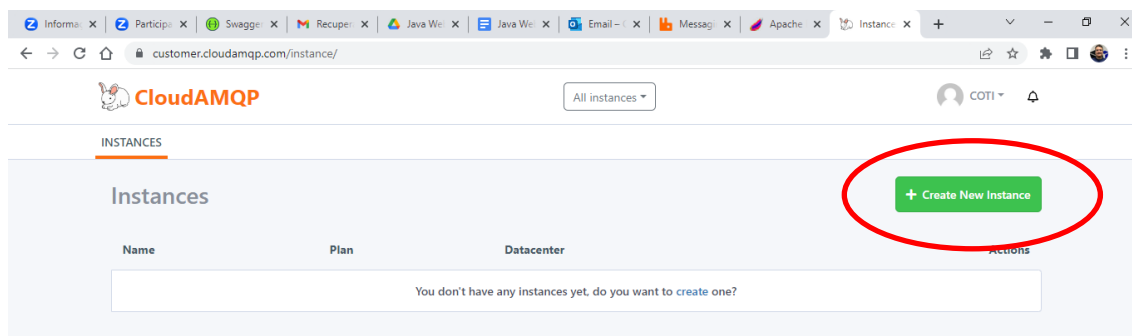
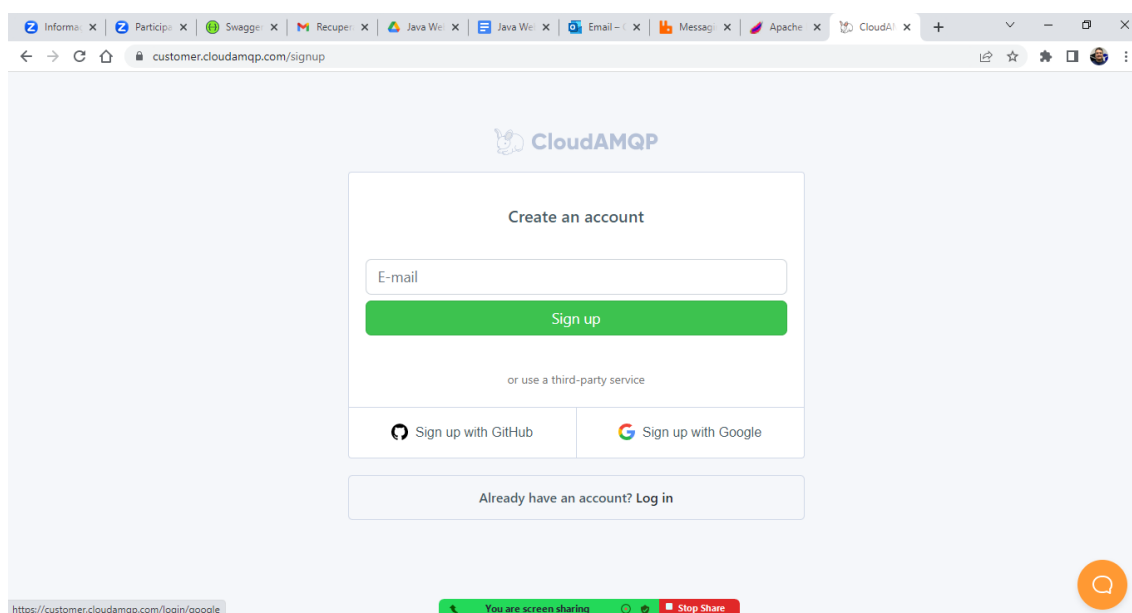
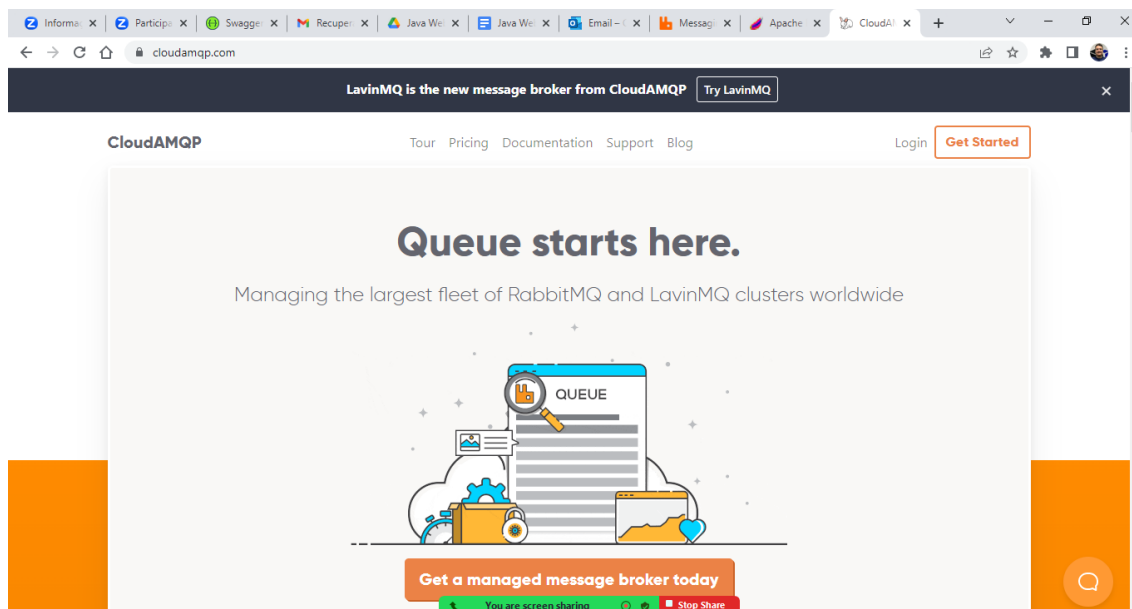
```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-amqp</artifactId>
</dependency>

<dependency>
  <groupId>org.springframework.amqp</groupId>
  <artifactId>spring-rabbit-test</artifactId>
  <scope>test</scope>
</dependency>
```

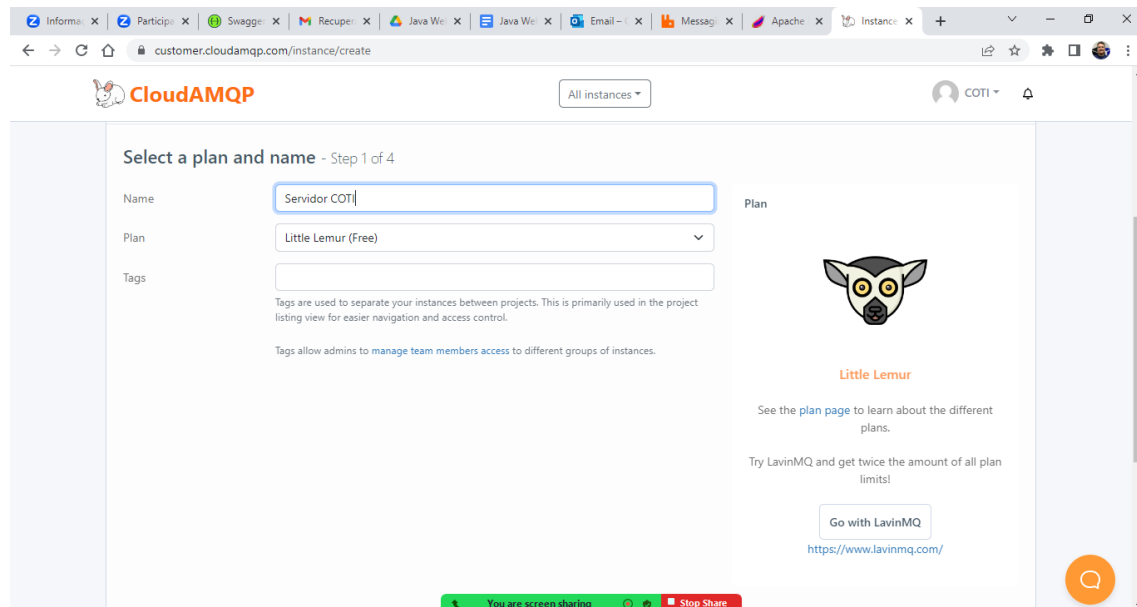


<https://www.cloudamqp.com/>

Servidor em nuvem para criação de serviços de mensageria através do RabbitMQ (Message Brokers)



## Criando um servidor de mensageria:



CloudAMQP

Select a plan and name - Step 1 of 4

Name:

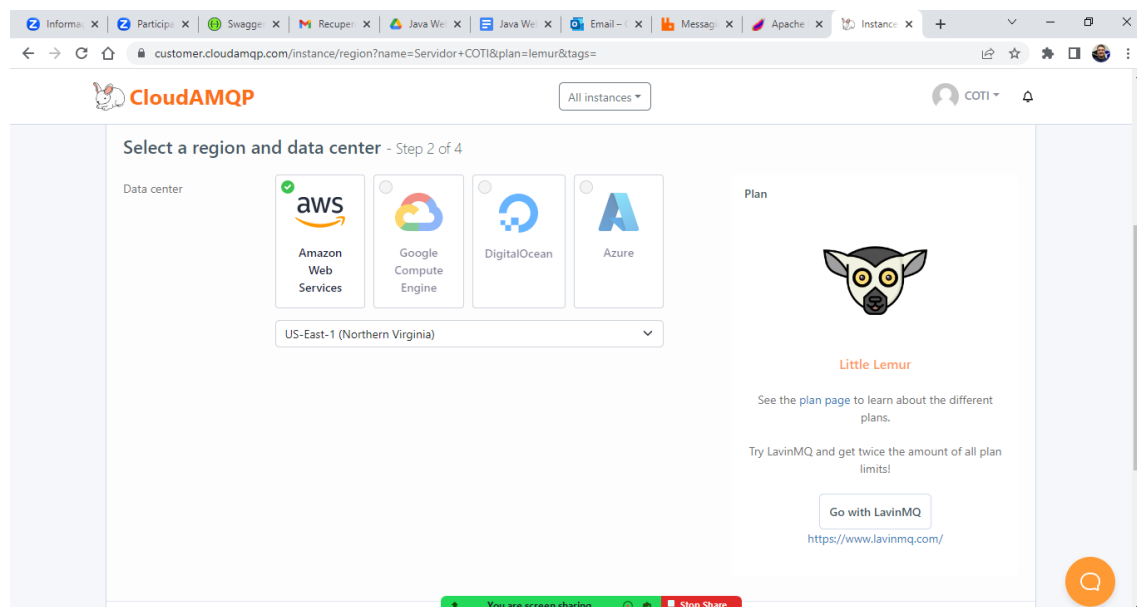
Plan:

Tags:

Plan: Little Lemur

Go with LavinMQ

<https://www.lavinmq.com/>



CloudAMQP

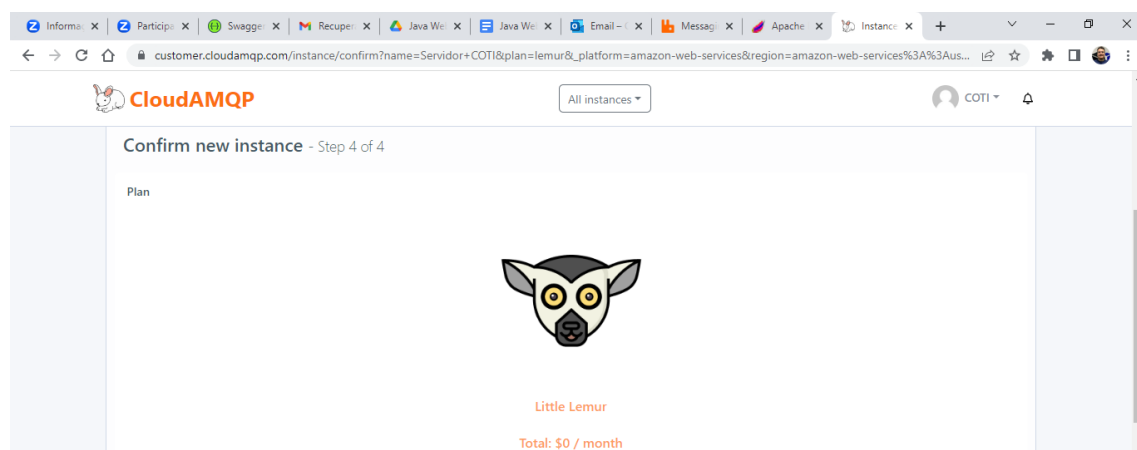
Select a region and data center - Step 2 of 4

Data center:

Plan: Little Lemur

Go with LavinMQ

<https://www.lavinmq.com/>

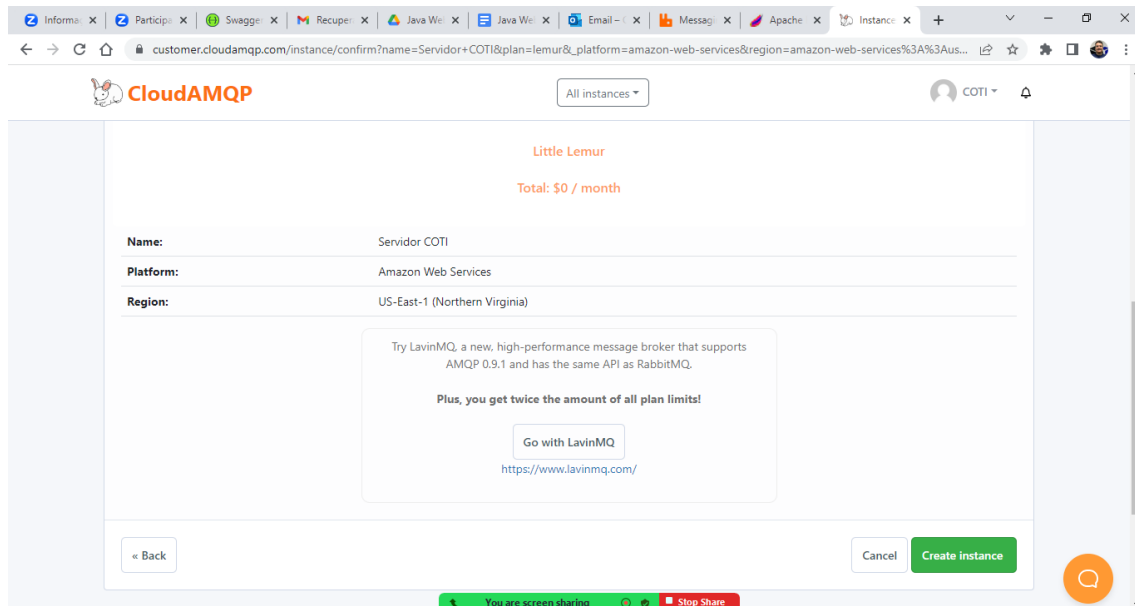


CloudAMQP

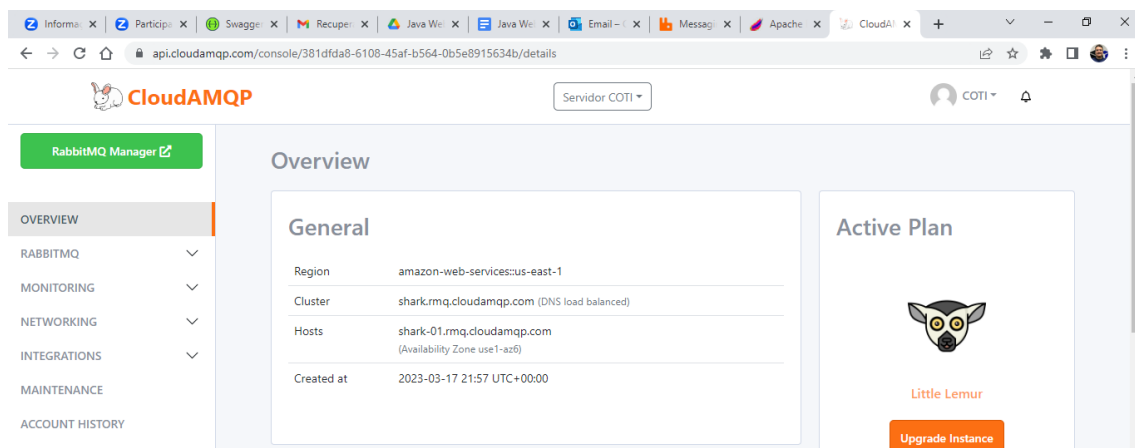
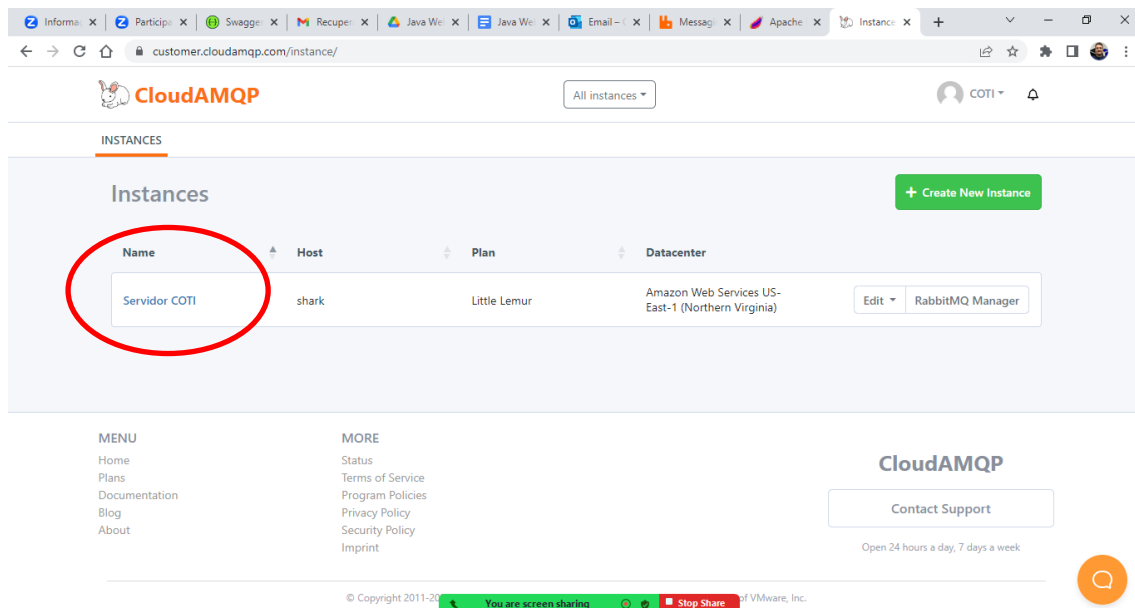
Confirm new instance - Step 4 of 4

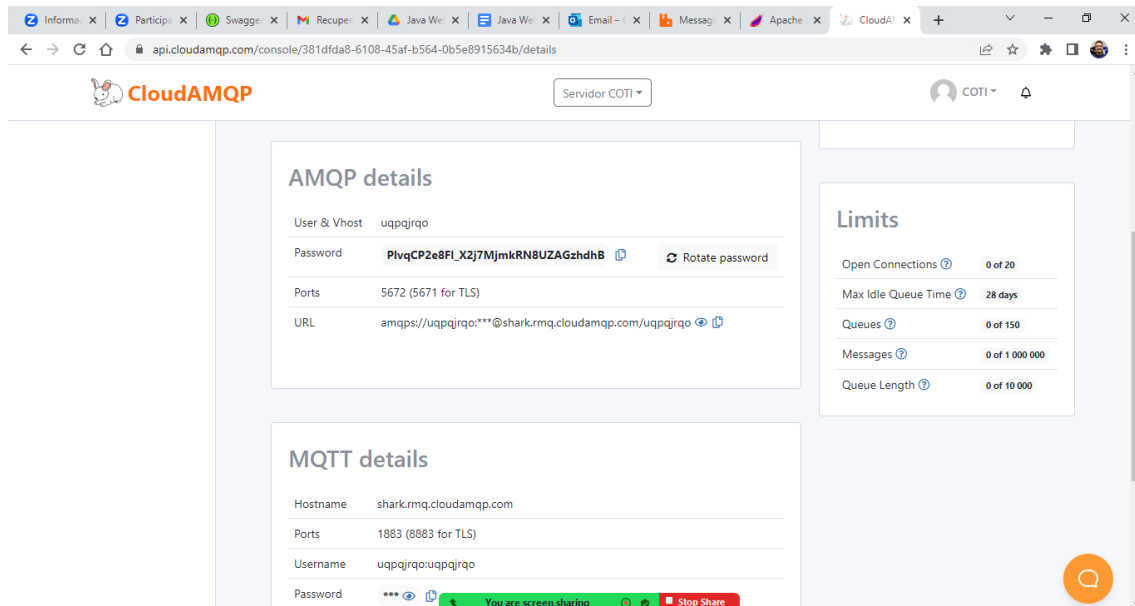
Plan: Little Lemur

Total: \$0 / month

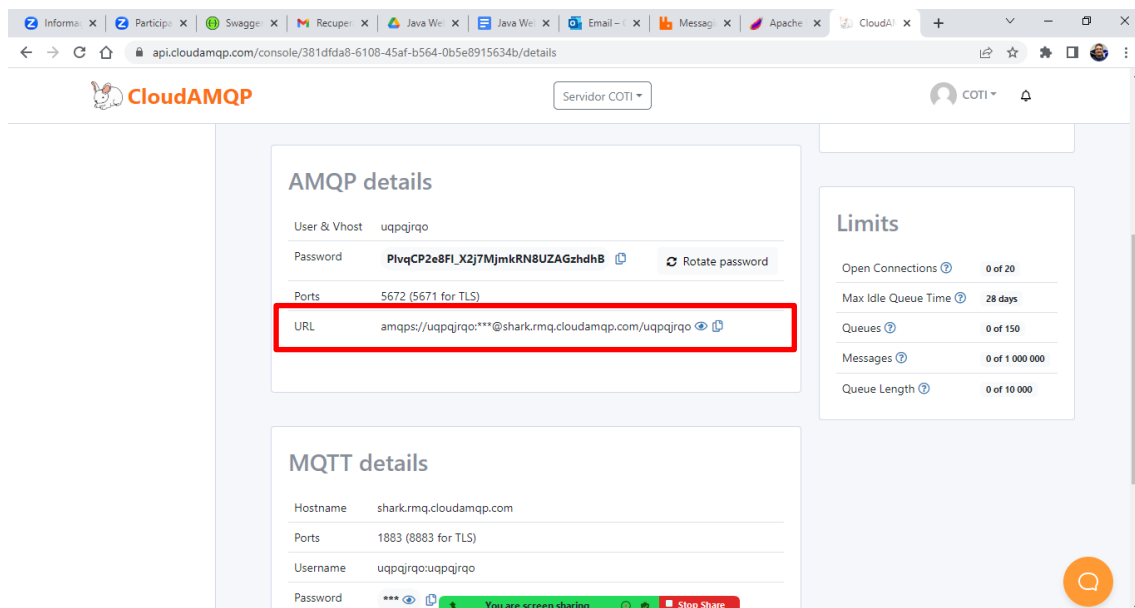


## Acessando o servidor:





Precisamos mapear as informações de conexão no servidor do RabbitMQ, para isso vamos usar o arquivo **/application.properties**



`server.port=8083`

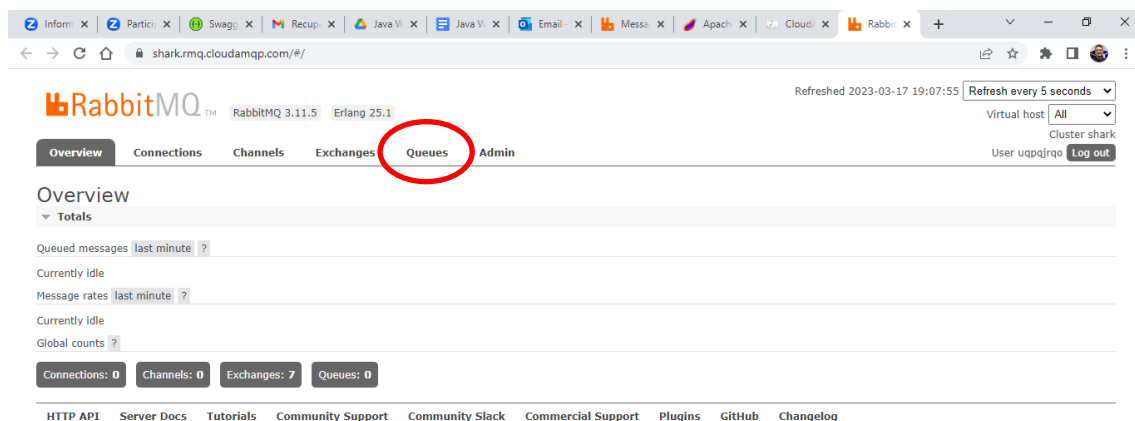
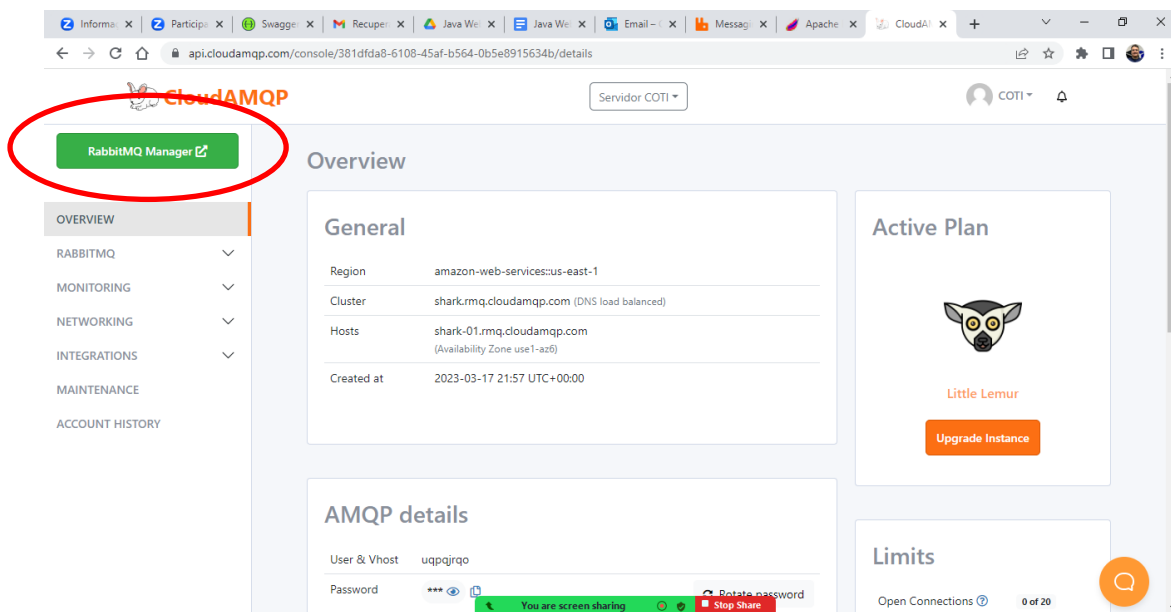
```
spring.datasource.url=jdbc:postgresql://database-coti.cybi4ptzkfaq.us-east-2.rds.amazonaws.com:5432/bd_apiusuarios
spring.datasource.driver-class-name=org.postgresql.Driver
spring.datasource.username=postgres
spring.datasource.password=coti2023
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.PostgreSQLDialect
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.format_sql=true
```

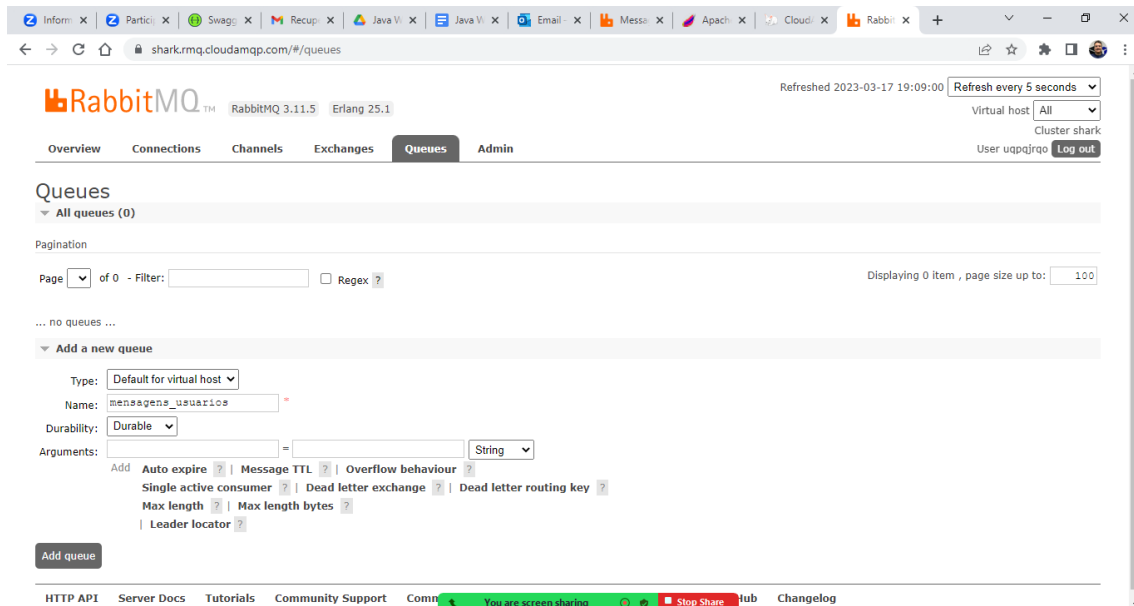
```
spring.mail.host=smtp-mail.outlook.com  
spring.mail.port=587  
spring.mail.username=csharpcoti@outlook.com  
spring.mail.password=@Admin12345  
spring.mail.properties.mail.smtp.auth=true  
spring.mail.properties.mail.smtp.starttls.enable=true
```

```
spring.rabbitmq.host=amqps://uqpqjrqr:PlvqCP2e8FI_X2j7MjmkRN8UZAGzhdhB  
@shark.rmcloudamqp.com/uqpqjrqr  
queue.name=mensagens_usuarios
```

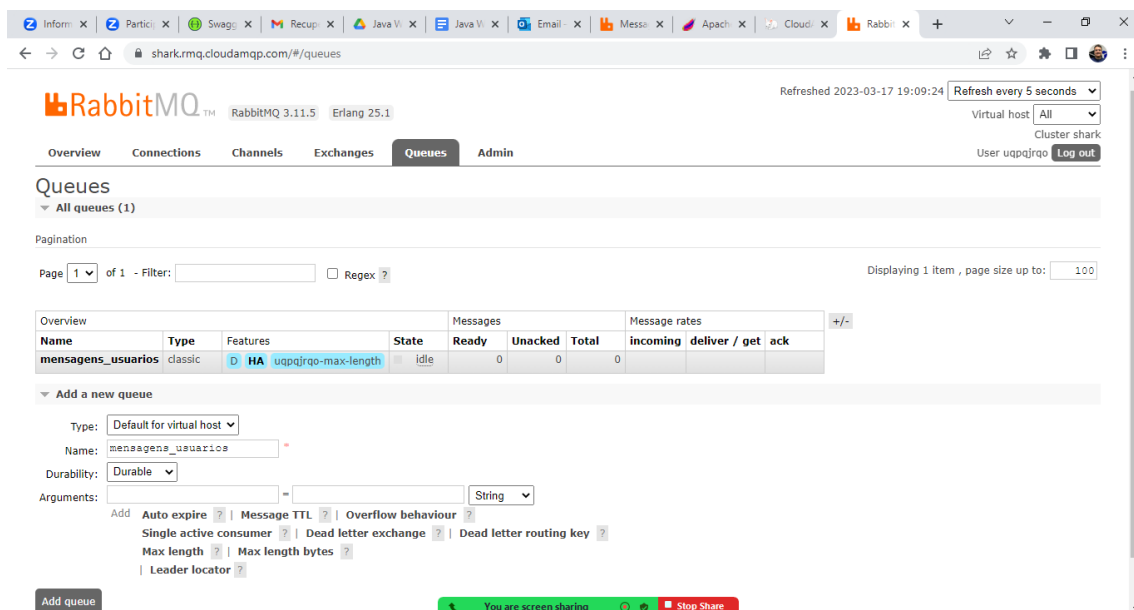
**Agora, precisamos criar a fila no servidor para onde serão enviadas as mensagens:**

`queue.name=mensagens_usuarios`





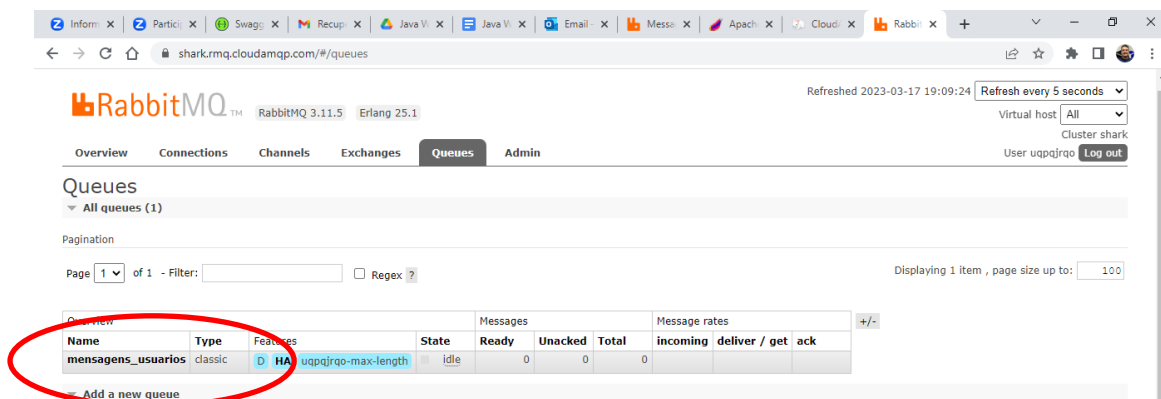
RabbitMQ CloudAMQP interface showing the 'Queues' tab. The page indicates 'All queues (0)'. The 'Add a new queue' form is visible, with fields for Name (mensagens\_usuarios), Durability (Durable), and Arguments. The 'Add queue' button is at the bottom.



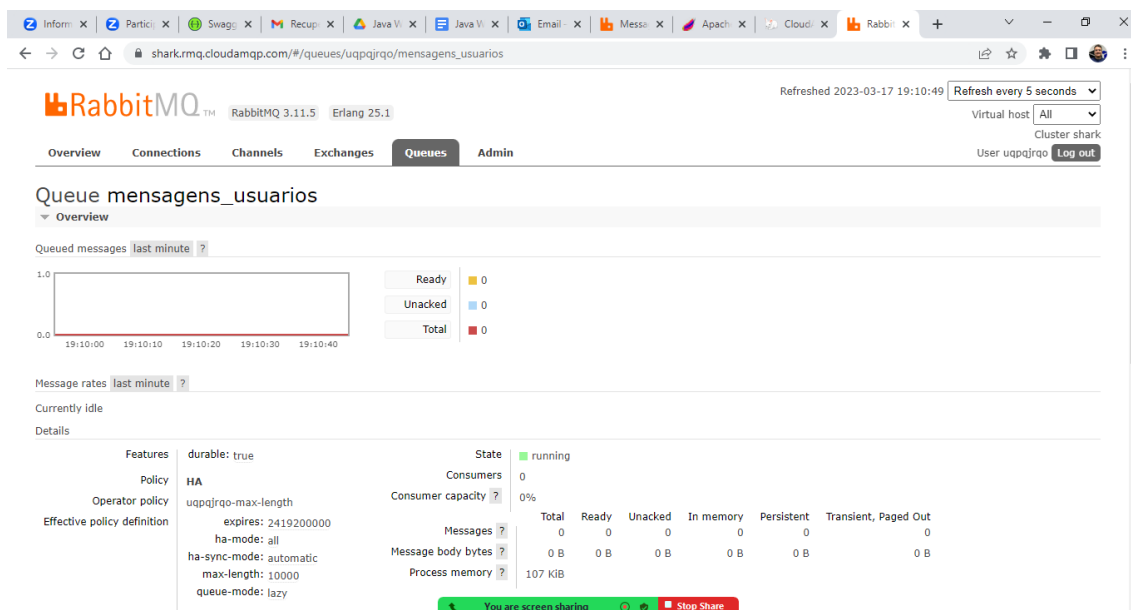
RabbitMQ CloudAMQP interface showing the 'Queues' tab. The page indicates 'All queues (1)'. A table lists the queue 'mensagens\_usuarios' with details like Type (classic), State (idle), and Message rates. The 'Add a new queue' form is also visible below the table.

Overview				Messages			Message rates			
Name	Type	Features	State	Ready	Unacked	Total	incoming	deliver	get	ack
mensagens_usuarios	classic	D HA uapqjrq-max-length	idle	0	0	0				

## Acessando a fila:



RabbitMQ CloudAMQP interface showing the 'Queues' tab. The page indicates 'All queues (1)'. A table lists the queue 'mensagens\_usuarios'. The queue name 'mensagens\_usuarios' in the table is circled in red. The 'Add a new queue' button is also visible.



Configurando o projeto Spring Boot para usar o RabbitMQ  
Criando uma simples configuração:

```
package br.com.cotiinformatica;
```

```
import org.springframework.amqp.rabbit.annotation.EnableRabbit;
import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.SpringBootApplication;
```

**@EnableRabbit** ←

@SpringBootApplication

```
public class ApiUsuariosApplication {

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

Criando uma classe para configurar o RabbitMQ:

/api/config/**RabbitMQConfig.java**

```
package br.com.cotiinformatica.api.config;
```

```
import org.springframework.amqp.core.Queue;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
```



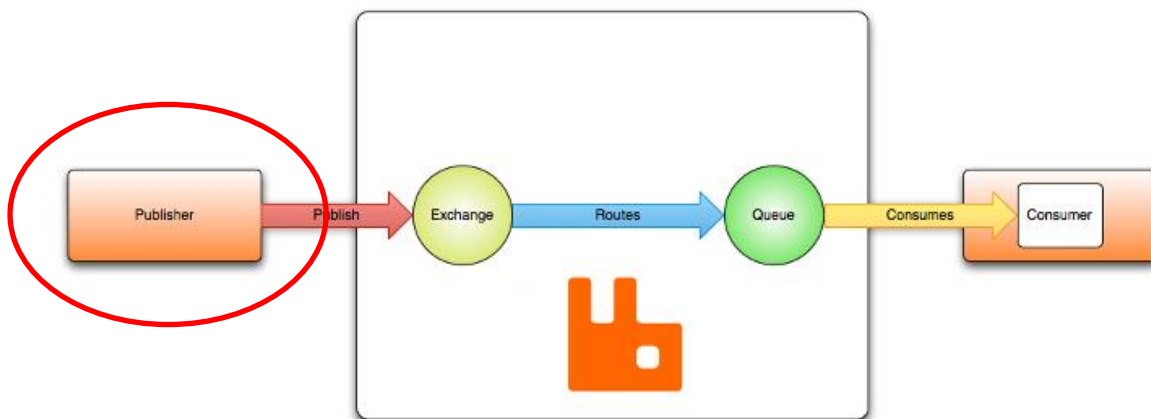
```
@Configuration
public class RabbitMQConfig {

    /**
     * Ler o nome da fila que será acessada mapeado no arquivo
     * application.properties
     */
    @Value("${queue.name}")
    private String queueName;

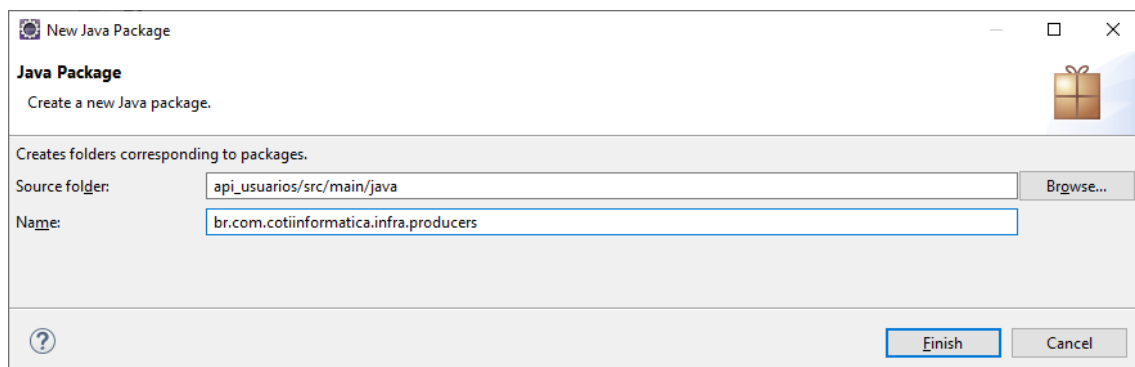
    /**
     * Método para conexão na fila
     */
    @Bean
    public Queue queue() {
        return new Queue(queueName, true);
    }
}
```

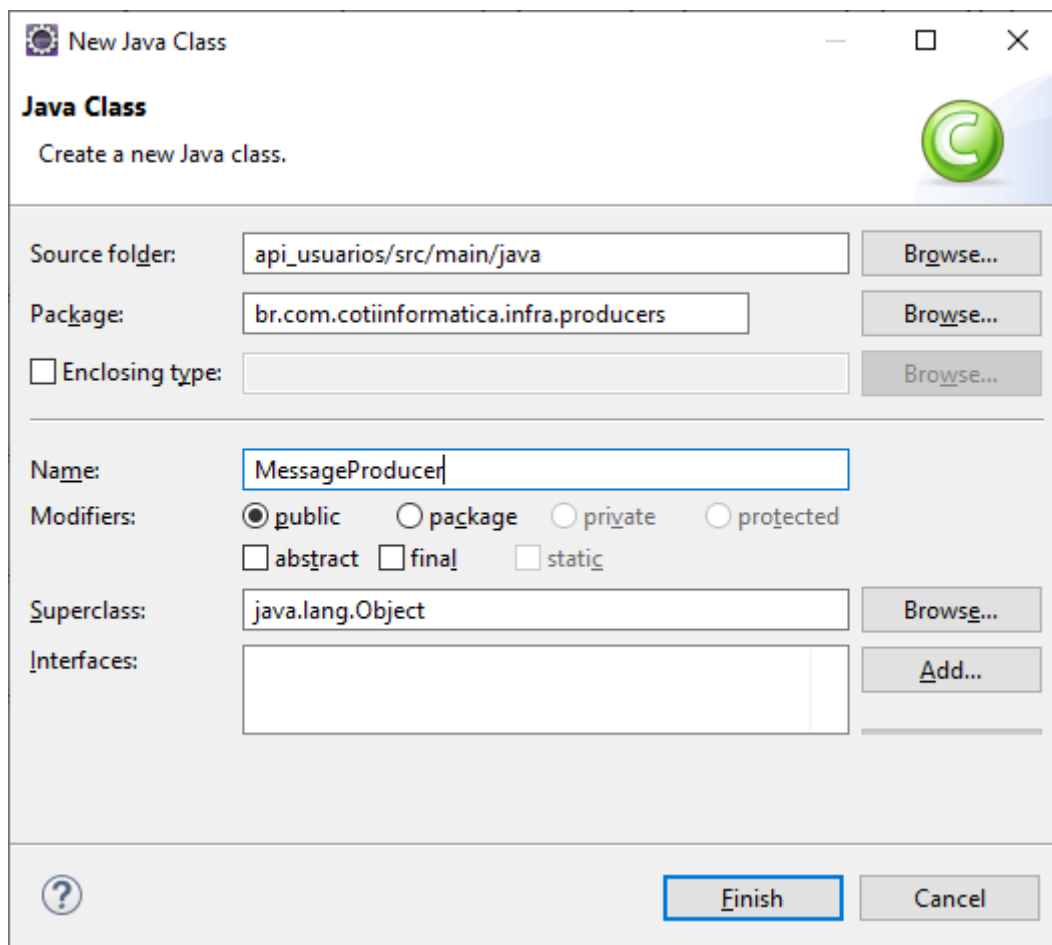
## PRODUCER / PUBLISHER

Consiste em um programa responsável por gravar um conteúdo na fila da mensageria. Ou seja, este programa cria novos itens na fila.



/infra/producers/**MessageProducer.java**





**New Java Class**

**Java Class**  
Create a new Java class.

Source folder:

Package:

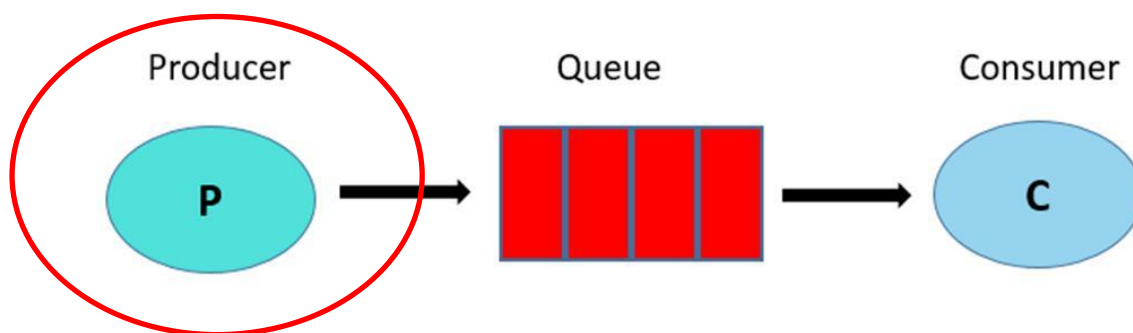
☐ Enclosing type:

Name:

Modifiers: ☒ public ☐ package ☐ private ☐ protected  
☐ abstract ☐ final ☐ static

Superclass:

Interfaces:



```
package br.com.cotiinformatica.infra.producers;
```

```
import org.springframework.amqp.core.Queue;
import org.springframework.amqp.rabbit.core.RabbitTemplate;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
```

```
@Service
```

```
public class MessageProducer {
```

```
    @Autowired
```

```
    private RabbitTemplate rabbitTemplate;
```

```
@Autowired
private Queue queue;

/*
 * Método acessado para gravar um item na FILA
 * PRODUCER / PUBLISHER -> Adicionar itens na fila
 */
public void send(String message) {
    rabbitTemplate.convertAndSend
        (this.queue.getName(), message);
}
}
```

---

**Criando uma classe DTO para modelar os dados necessários para fazer um envio de email: (DATA TRANSFER OBJECT)**

```
package br.com.cotiinformatica.application.dtos;
```

```
import lombok.AllArgsConstructor;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;
import lombok.ToString;
```

```
@Setter
@Getter
@NoArgsConstructor
@AllArgsConstructor
@ToString
public class EmailMessageDTO {

    private String emailTo;
    private String subject;
    private String body;
}
```

---

**Voltando na camada de serviço de domínio:**

Escrevendo mensagens na **FILA** para 2 ações do projeto.

- Usuário cadastrado com sucesso
  - EmailMessageDTO contendo uma mensagem de boas-vindas para o usuário.
- Recuperação de senha do usuário
  - EmailMessageDTO contendo uma mensagem de recuperação de senha.

## /UsuarioDomainService.java

```
package br.com.cotiinformatica.domain.services;

import java.util.Date;

import org.modelmapper.ModelMapper;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import com.fasterxml.jackson.databind.ObjectMapper;
import com.github.javafaker.Faker;

import br.com.cotiinformatica.application.dtos.EmailMessageDTO;
import br.com.cotiinformatica.application.dtos.GetUsuarioDTO;
import br.com.cotiinformatica.application.dtos.PostAutenticarDTO;
import br.com.cotiinformatica.application.dtos.PostCriarContaDTO;
import br.com.cotiinformatica.application.dtos.PostRecuperarSenhaDTO;
import br.com.cotiinformatica.application.dtos.ResponseAutenticarDTO;
import br.com.cotiinformatica.application.dtos.ResponseCriarContaDTO;
import br.com.cotiinformatica.application.dtos.ResponseRecuperarSenhaDTO;
import br.com.cotiinformatica.domain.interfaces.IUsuarioDomainService;
import br.com.cotiinformatica.domain.models.Usuario;
import br.com.cotiinformatica.infra.components.MD5Component;
import br.com.cotiinformatica.infra.components.TokenComponent;
import br.com.cotiinformatica.infra.producers.MessageProducer;
import br.com.cotiinformatica.infra.repositories.IUsuarioRepository;

@Service
public class UsuarioDomainService implements IUsuarioDomainService {

    @Autowired // injeção de dependência
    private IUsuarioRepository usuarioRepository;

    @Autowired // injeção de dependência
    private MD5Component md5Component;

    @Autowired // injeção de dependência
    private TokenComponent tokenComponent;

    @Autowired // injeção de dependência
    private MessageProducer messageProducer;

    @Autowired // injeção de dependência
    private ObjectMapper objectMapper;

    @Override
    public ResponseAutenticarDTO autenticar(PostAutenticarDTO dto) {

        // procurar o usuário no banco de dados através do email e da senha
        Usuario usuario = usuarioRepository.findByEmailAndSenha(
            dto.getEmail(), md5Component.encrypt(dto.getSenha()));

        // verificar se o usuário não foi encontrado
        if (usuario == null)
            throw new IllegalArgumentException(
                "Acesso negado. Usuário inválido.");

        // transferir os dados do usuário para o DTO
        ModelMapper modelMapper = new ModelMapper();
```

```
GetUsuarioDTO usuarioDTO = modelMapper.map
(usuario, GetUsuarioDTO.class);

// gerando o token JWT para o usuário
String token = null;
try {
    token = tokenComponent.generateToken(usuario.getEmail());
} catch (Exception e) {
    e.printStackTrace();
}

ResponseAutenticarDTO response = new ResponseAutenticarDTO();
response.setStatus(200);
response.setMensagem("Usuário autenticado com sucesso.");
response.setToken(token);
response.setUsuario(usuarioDTO);

return response;
}

@Override
public ResponseCriarContaDTO criarConta(PostCriarContaDTO dto) {

    // verificar se já existe um usuário cadastrado com o email informado
    if (usuarioRepository.findByEmail(dto.getEmail()) != null)
        throw new IllegalArgumentException
            ("O email informado já está cadastrado. Tente outro.");

    // transferir os dados do DTO para a classe de modelo de entidade
    ModelMapper modelMapper = new ModelMapper();
    Usuario usuario = modelMapper.map(dto, Usuario.class);

    usuario.setSenha(md5Component.encrypt(usuario.getSenha()));
    usuario.setDataHoraCriacao(new Date());

    // gravando no banco de dados
    usuarioRepository.save(usuario);

    // enviando mensagem de boas vindas
    EmailMessageDTO emailMessageDTO = new EmailMessageDTO();
    emailMessageDTO.setEmailTo(usuario.getEmail());
    emailMessageDTO.setSubject("Seja bem vindo a API
        de Usuários - COTI Informática");
    emailMessageDTO.setBody(
        "<div>" +
        "<p>Parabéns, " + usuario.getNome()
            + ". Sua conta de
            usuário foi criada com sucesso.</p>" +
        "<p>Utilize o email e senha cadastrados para
            acessar sua conta." +
        "<p>Att,<br/>Equipe COTI Informática</p>"
    );

    // enviando a mensagem para a fila..
    try {
        messageProducer.send(objectMapper.writeValueAsString
            (emailMessageDTO));
    }
    catch (Exception e) {
        e.printStackTrace();
    }
}
```

```
ResponseCriarContaDTO response = new ResponseCriarContaDTO();
response.setStatus(201);
response.setMensagem("Usuário cadastrado com sucesso");
response.setDataHoraCadastro(new Date());
response.setUsuario(modelMapper.map(usuario, GetUsuarioDTO.class));

return response;
}

@Override
public ResponseRecuperarSenhaDTO recuperarSenha(PostRecuperarSenhaDTO dto) {

    // procurar o usuário no banco de dados através do email
    Usuario usuario = usuarioRepository.findByEmail(dto.getEmail());

    // verificar se o usuário foi encontrado
    if (usuario == null)
        throw new IllegalArgumentException
            ("Email não encontrado. Usuário inválido.");

    // gerar uma nova senha para o usuário
    String novaSenha = new Faker().internet().password(8, 10, true);

    //enviando mensagem de recuperação de senha
    EmailMessageDTO emailMessageDTO = new EmailMessageDTO();
    emailMessageDTO.setEmailTo(usuario.getEmail());
    emailMessageDTO.setSubject("Recuperação de Senha
        (API de Usuários) - COTI Informática");
    emailMessageDTO.setBody(
        "<div>" +
        "<p>Parabéns, " + usuario.getNome()
        + ". Uma nova senha foi gerada com sucesso.</p>" +
        "<p>Utilize a senha <strong>" + novaSenha
        + "</strong> para acessar sua conta." +
        "<p>Att,<br/>Equipe COTI Informática</p>"
    );

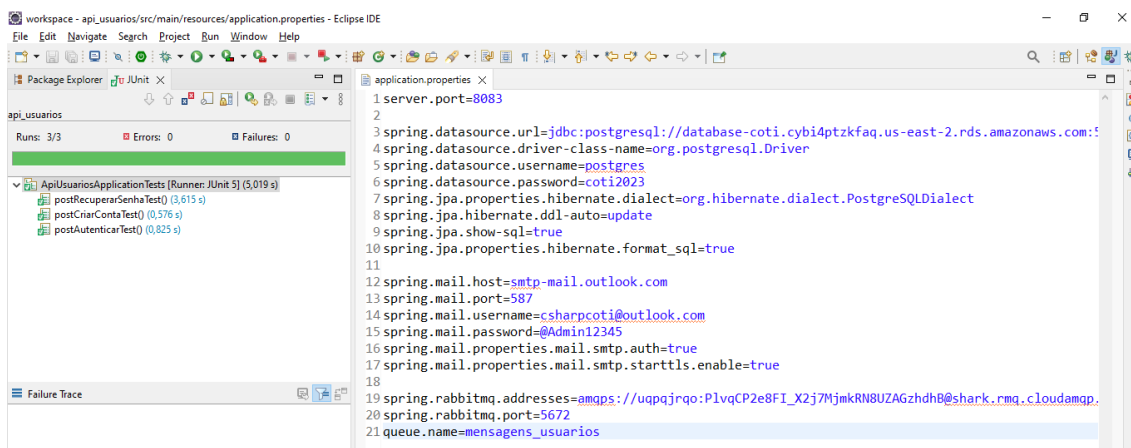
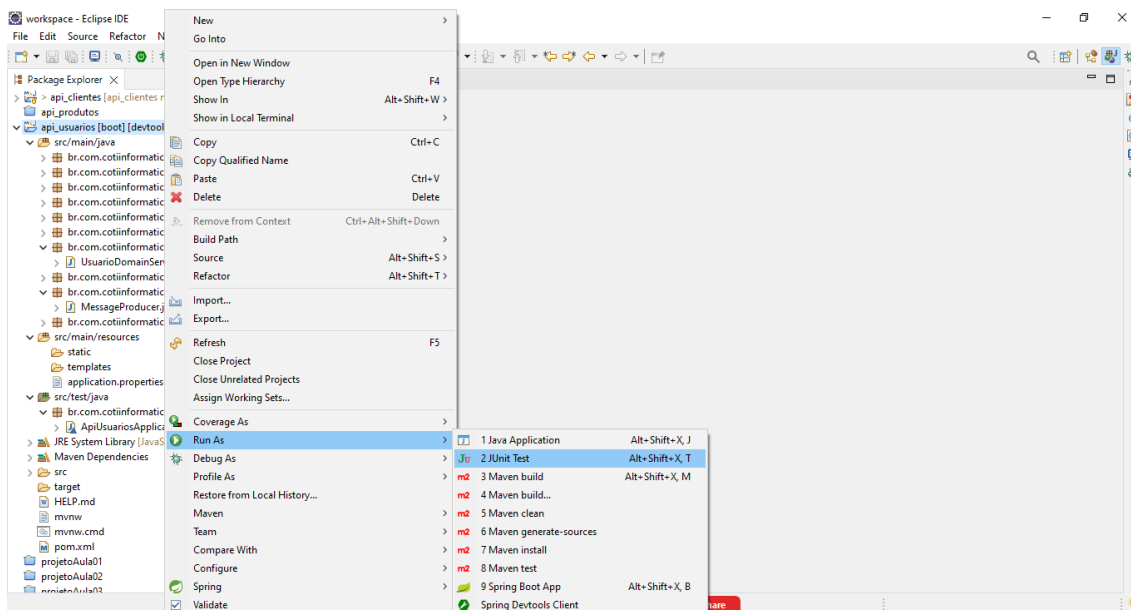
    //enviando a mensagem para a fila..
    try {
        messageProducer.send
            (objectMapper.writeValueAsString(emailMessageDTO));
    }
    catch(Exception e) {
        e.printStackTrace();
    }

    // atualizando a senha do usuário no banco de dados
    usuario.setSenha(md5Component.encrypt(novaSenha));
    usuarioRepository.save(usuario);

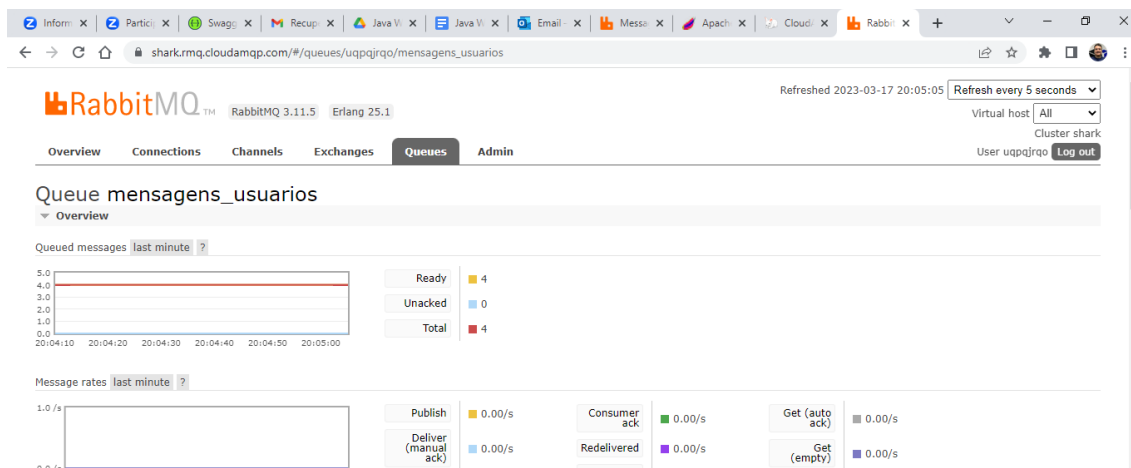
    ResponseRecuperarSenhaDTO response
        = new ResponseRecuperarSenhaDTO();
    response.setStatus(200);
    response.setMensagem("Recuperação de senha realizado com sucesso.");

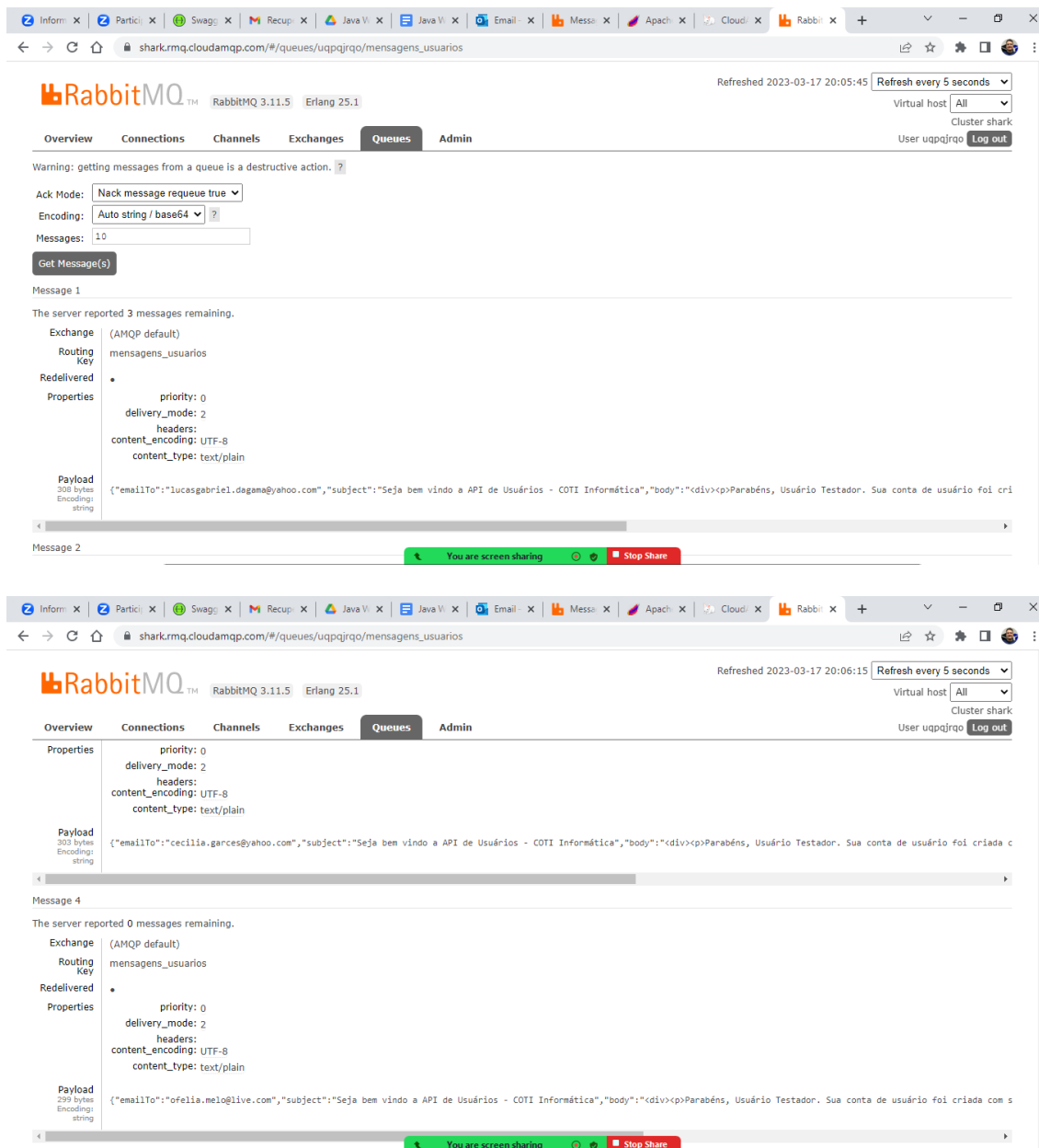
    return response;
}
}
```

## Executando os testes:



## Na fila do servidor do RabbitMQ:





The first screenshot shows Message 1 with the following details:

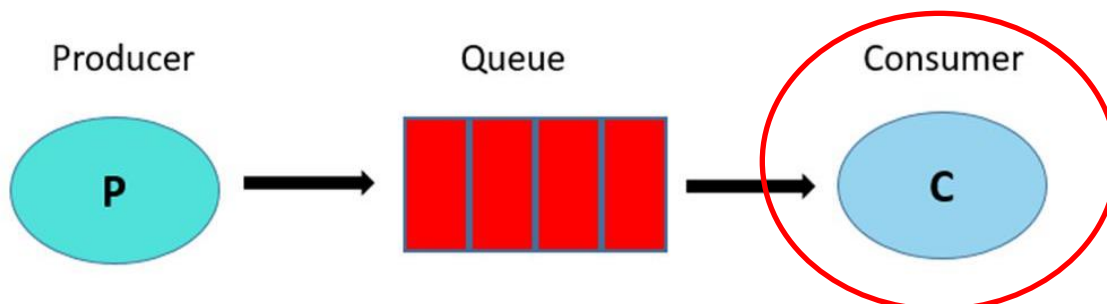
- Exchange: (AMQP default)
- Routing Key: mensagens\_usuarios
- Redelivered: 0
- Properties:
  - priority: 0
  - delivery\_mode: 2
  - headers: content\_encoding: UTF-8, content\_type: text/plain
- Payload: 308 bytes, Encoding: string

The second screenshot shows Message 4 with the following details:

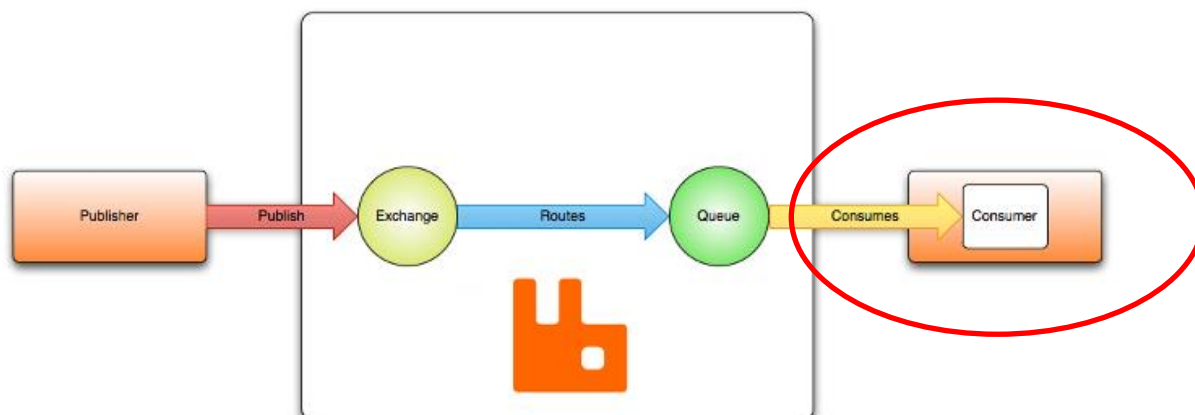
- Exchange: (AMQP default)
- Routing Key: mensagens\_usuarios
- Redelivered: 0
- Properties:
  - priority: 0
  - delivery\_mode: 2
  - headers: content\_encoding: UTF-8, content\_type: text/plain
- Payload: 299 bytes, Encoding: string

## CONSUMER

Programa que irá ler e processar os itens da FILA. Neste caso será uma classe para varrer e processar o envio de cada mensagem contida na fila.







**New Java Package**

**Java Package**  
Create a new Java package.

Creates folders corresponding to packages.

Source folder:

Name:

**New Java Class**

**Java Class**  
Create a new Java class.

Source folder:

Package:

☐ Enclosing type:

Name:

Modifiers: ☒ public ☐ package ☐ private ☐ protected  
☐ abstract ☐ final ☐ static

Superclass:

Interfaces:

```
package br.com.cotiinformatica.infra.consumers;

import org.springframework.amqp.rabbit.annotation.RabbitListener;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.messaging.handler.annotation.Payload;
import org.springframework.stereotype.Service;

import com.fasterxml.jackson.databind.ObjectMapper;

import br.com.cotiinformatica.application.dtos.EmailMessageDTO;
import br.com.cotiinformatica.infra.components.EmailComponent;

@Service
public class MessageConsumer {

    @Autowired // Injeção de dependência
    EmailComponent emailComponent;

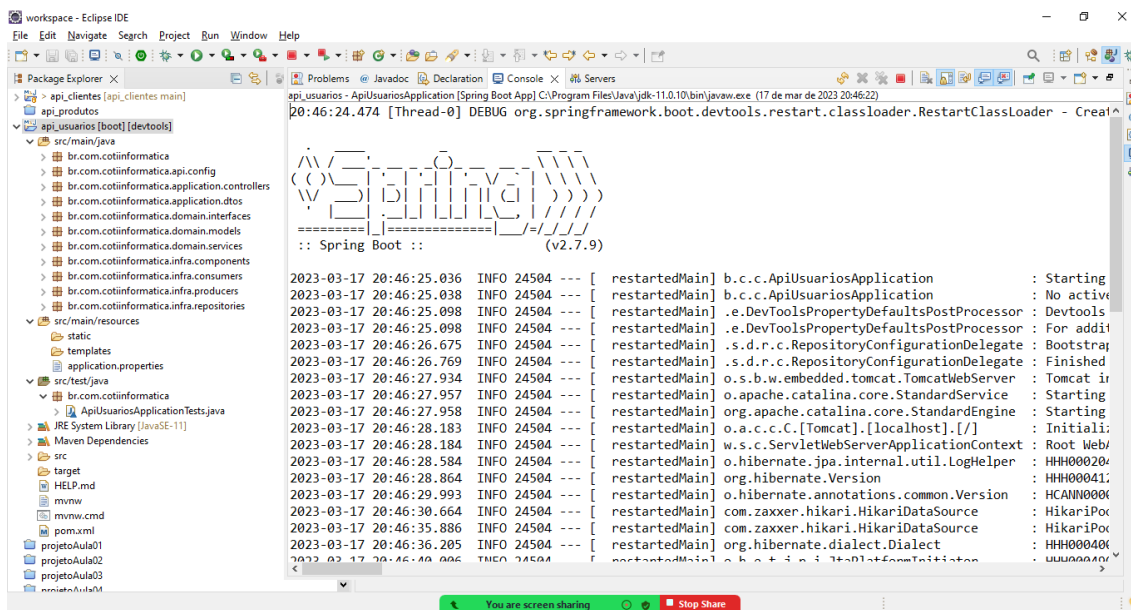
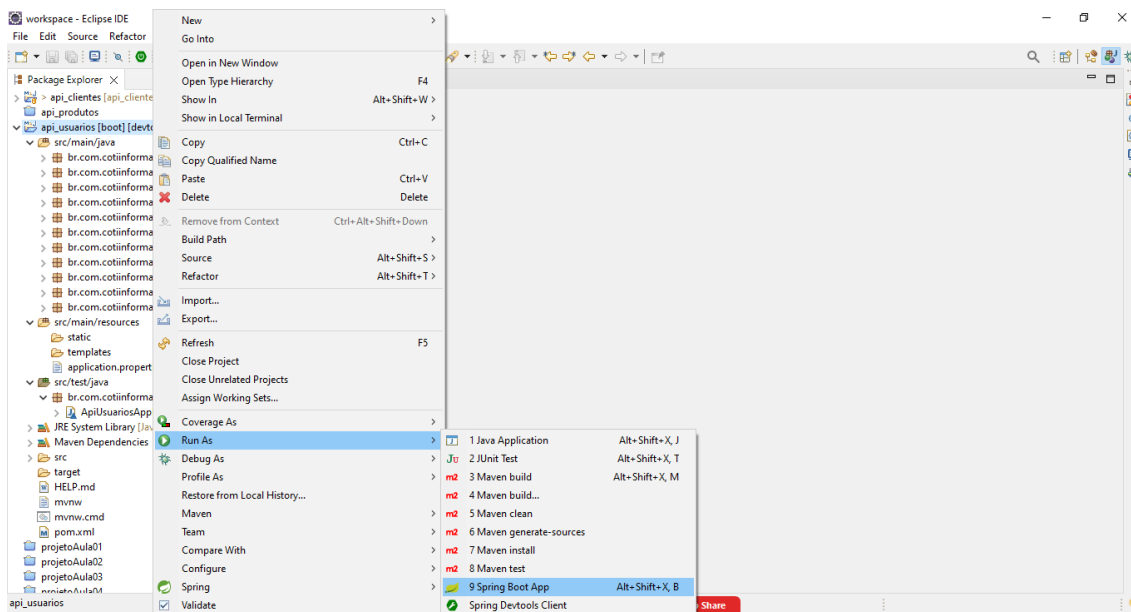
    @Autowired // Injeção de dependência
    ObjectMapper objectMapper;

    /**
     * Método para ler e processar a fila
     */
    @RabbitListener(queues = { "${queue.name}" })
    public void receive(@Payload String payload) throws Exception {

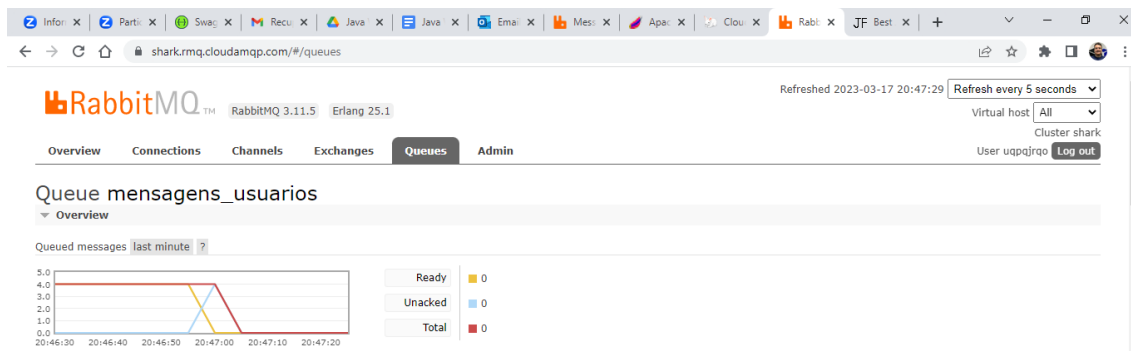
        // lendo o conteudo da mensagem
        // gravada na fila do RabbitMQ
        EmailMessageDTO emailMessageDTO = objectMapper.readValue
            (payload, EmailMessageDTO.class);

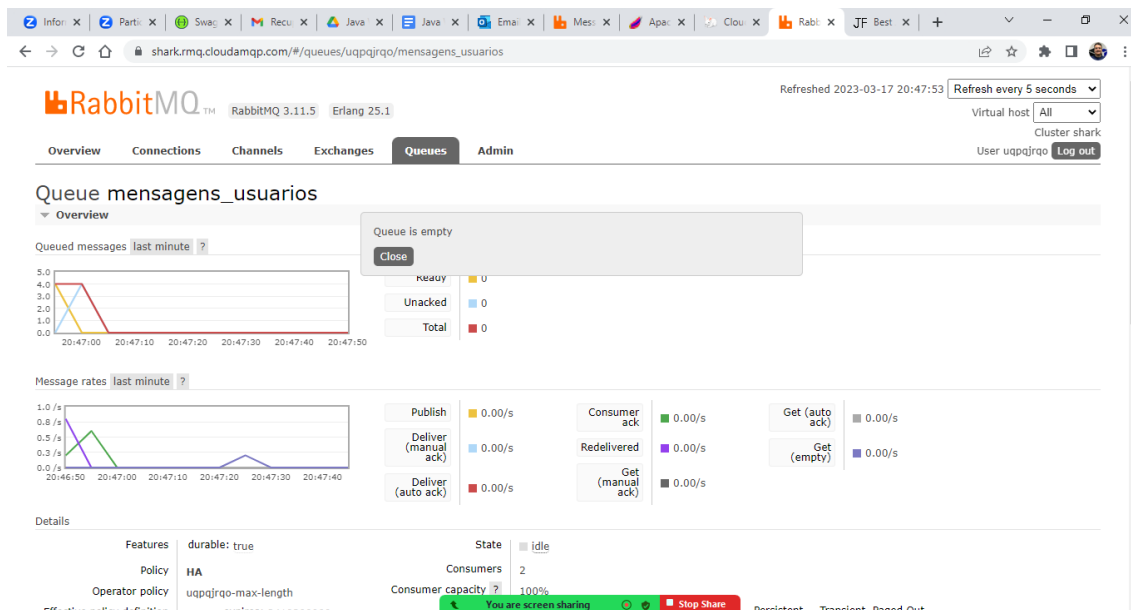
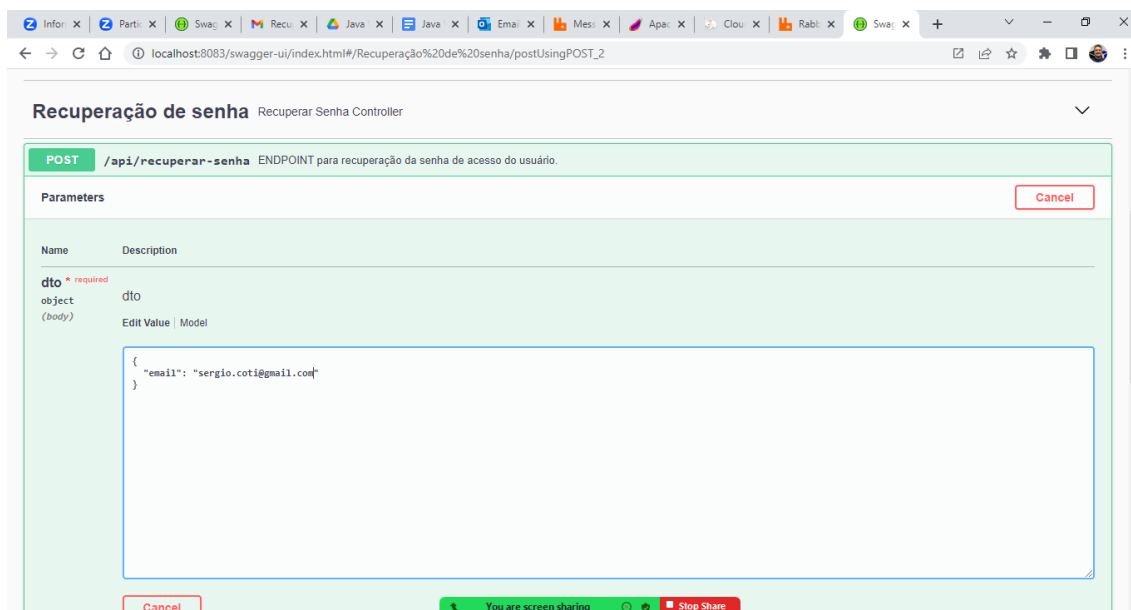
        // enviar o email
        emailComponent.sendMessage(emailMessageDTO.getEmailTo(),
            emailMessageDTO.getSubject(),
            emailMessageDTO.getBody());
    }
}
```

## Executando o projeto:



## Fila do RabbitMQ:



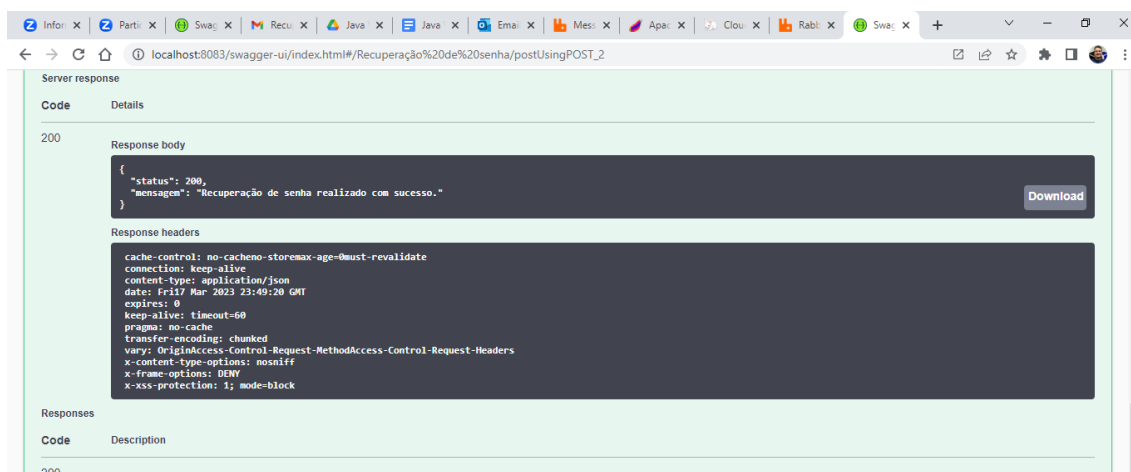



The image shows the Swagger UI for the password recovery endpoint. The endpoint is a POST request to '/api/recuperar-senha'. The parameters section shows a required 'dto' object with a 'body' type. The body is a JSON object with an 'email' field, which is currently set to 'sergio.coti@gmail.com'.

```

{
  "email": "sergio.coti@gmail.com"
}

```



The image shows the Swagger UI displaying the server response for the password recovery endpoint. The response is a 200 status code with a JSON body and several headers.

```

{
  "status": 200,
  "mensagem": "Recuperação de senha realizado com sucesso."
}

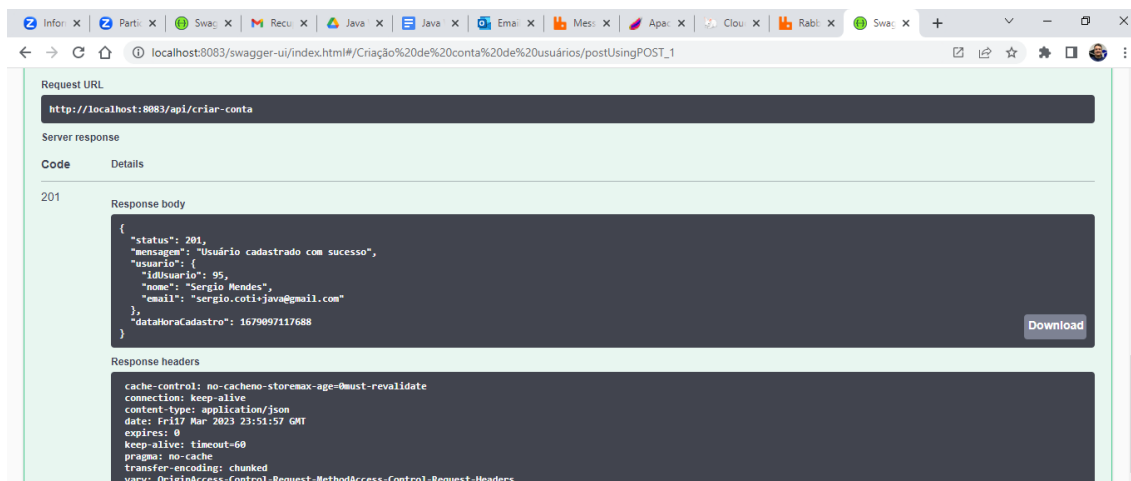
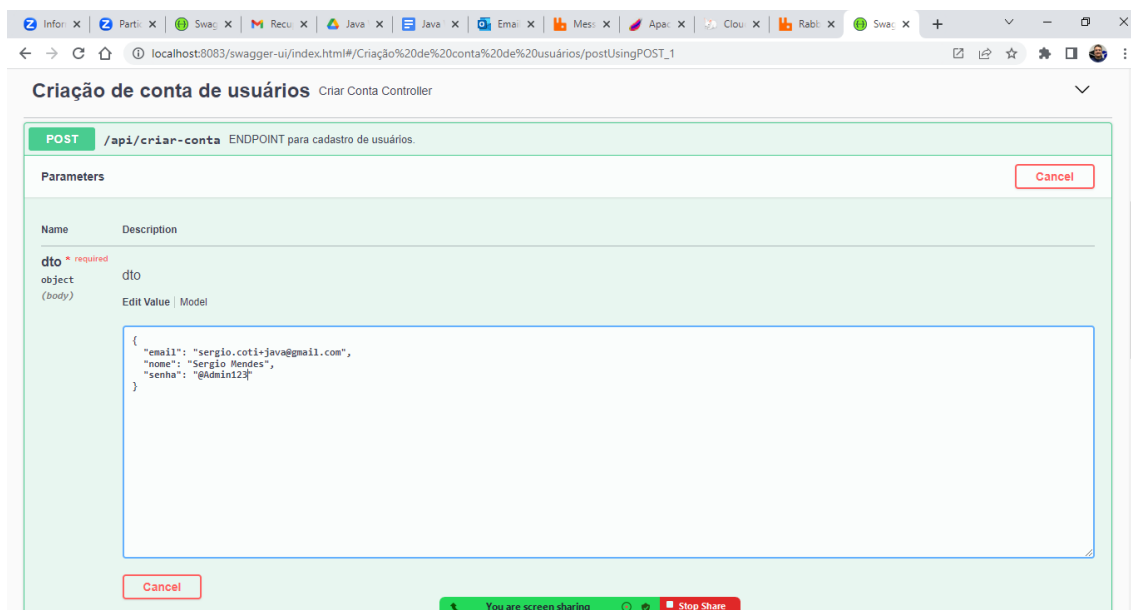
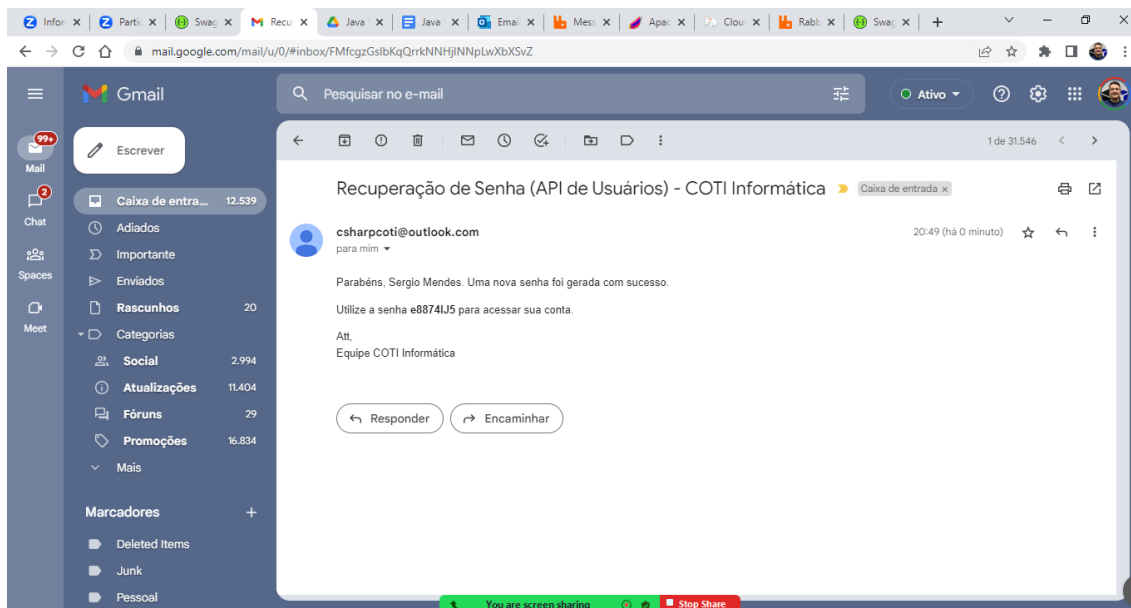
```

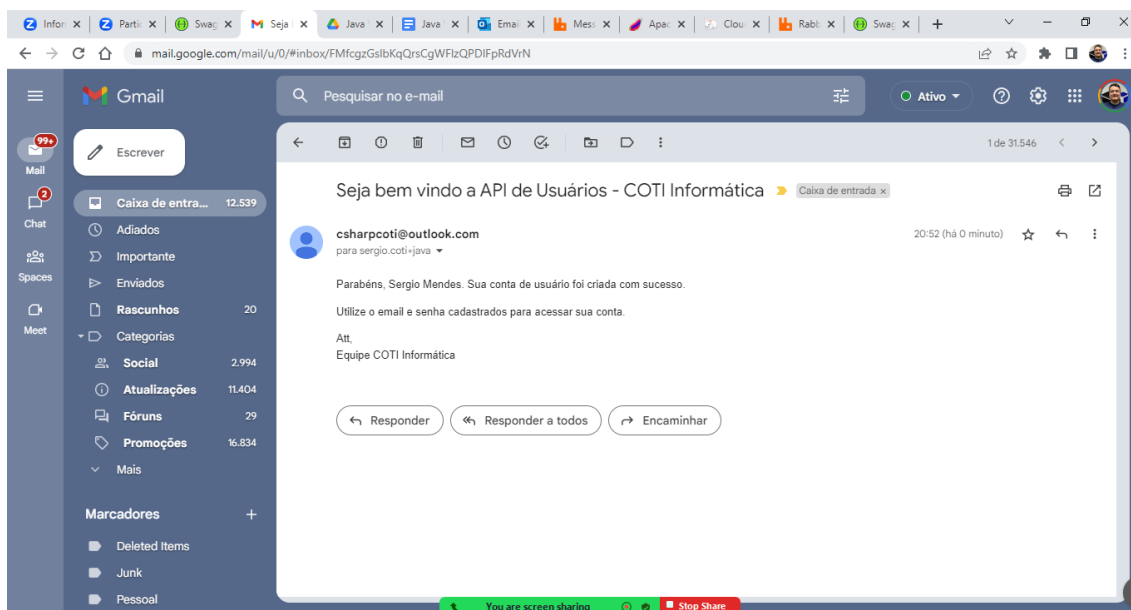
Response headers:

```

cache-control: no-cache, no-store, max-age=0, must-revalidate
connection: keep-alive
content-type: application/json
date: Fri, 17 Mar 2023 23:49:20 GMT
expires: 0
keep-alive: timeout=60
pragma: no-cache
transfer-encoding: chunked
vary: Origin, Access-Control-Request-Method, Access-Control-Request-Headers
x-content-type-options: nosniff
x-frame-options: DENY
x-xss-protection: 1; mode=block

```





**Publicando o projeto no AWS com essa atualização:**  
Gerando o DEPLOY do projeto:

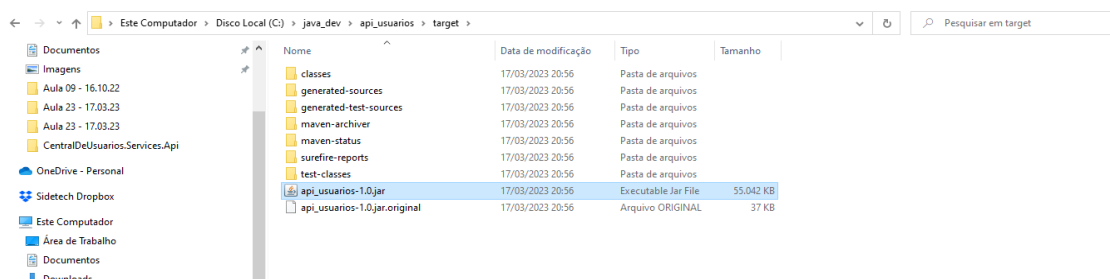
C:\java\_dev\api\_usuarios> **mvnw clean package**

```

for workers to finish.
2023-03-17 20:56:56.163 INFO 19896 --- [ionShutdownHook] o.s.a.r.l.SimpleMessageListenerContainer : Shutdown ignored - c
ontainer is already stopped
2023-03-17 20:56:56.477 INFO 19896 --- [ionShutdownHook] j.LocalContainerEntityManagerFactoryBean : Closing JPA EntityMa
nagerFactory for persistence unit 'default'
2023-03-17 20:56:56.480 INFO 19896 --- [ionShutdownHook] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Shutd
own initiated...
2023-03-17 20:56:56.497 INFO 19896 --- [ionShutdownHook] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Shutd
own completed.
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 3, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] --- maven-jar-plugin:3.2.2:jar (default-jar) @ api_usuarios ---
[INFO] Building jar: C:\java_dev\api_usuarios\target\api_usuarios-1.0.jar
[INFO] --- spring-boot-maven-plugin:2.7.9:repackage (repackage) @ api_usuarios ---
[INFO] Replacing main artifact with repackaged archive
[INFO]
[INFO] BUILD SUCCESS
[INFO]
[INFO] Total time: 40.022 s
[INFO] Finished at: 2023-03-17T20:56:58-03:00
[INFO]
C:\java_dev\api_usuarios>

```

**Deploy gerado:**

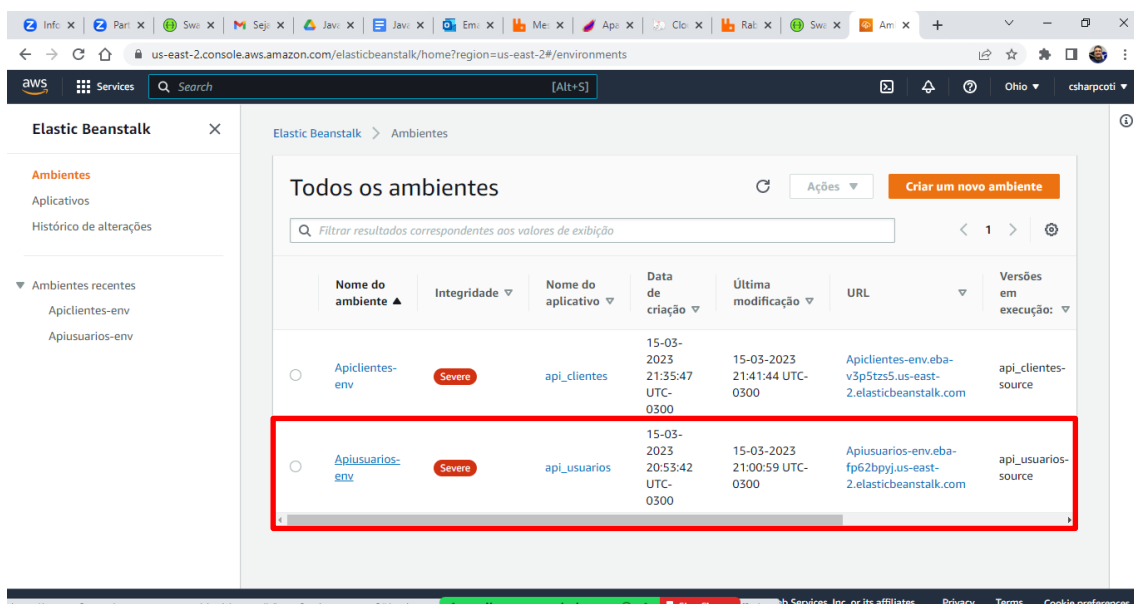


## Enviando para o AWS:

<https://aws.amazon.com/pt/>

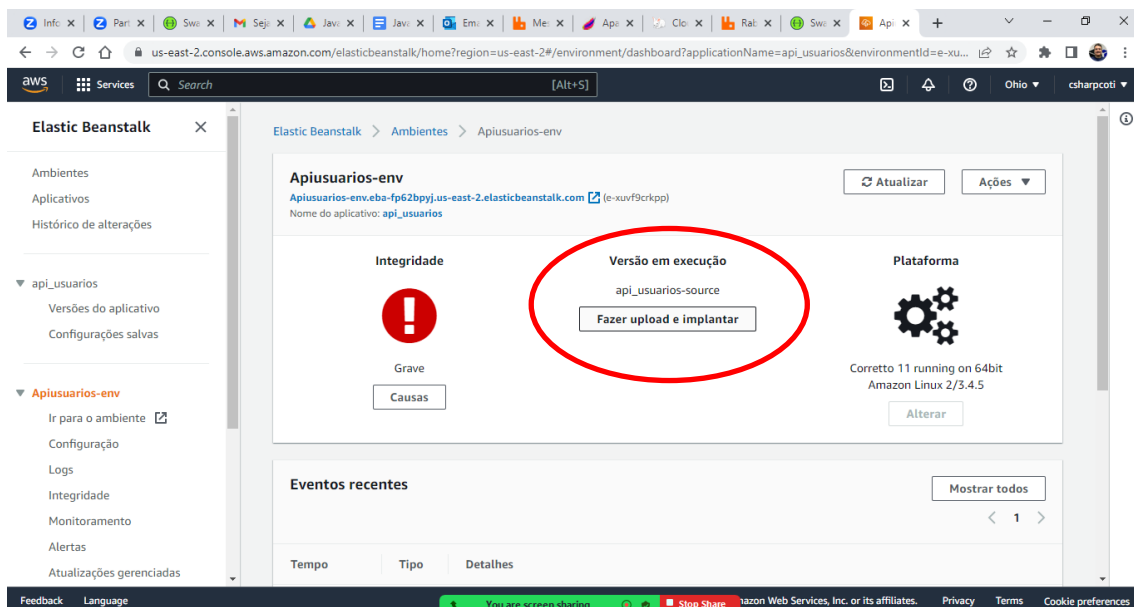
## Elastic BeanStalk

Serviço para publicação de aplicações web no AWS.



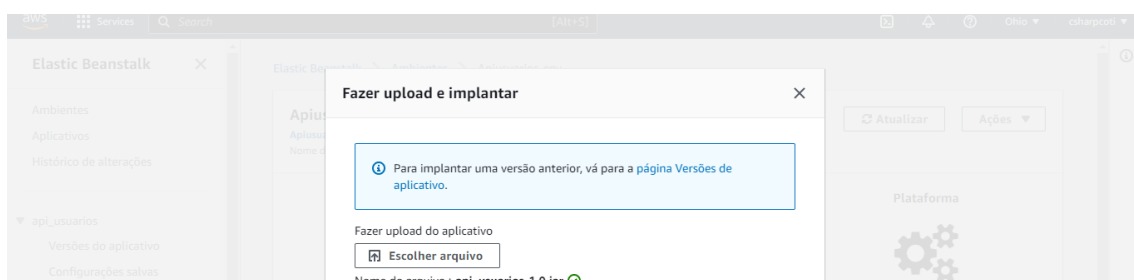
The screenshot shows the AWS Elastic Beanstalk console. The left sidebar contains navigation links: Elastic Beanstalk, Ambientes, Aplicativos, and Histórico de alterações. The main content area is titled 'Todos os ambientes' and displays a table of environments. The table has columns for Nome do ambiente, Integridade, Nome do aplicativo, Data de criação, Última modificação, URL, and Versões em execução. Two environments are listed: 'Apiclientes-env' and 'Apiusuarios-env'. The 'Apiusuarios-env' row is highlighted with a red box. Below the table, there is a section for 'Apiusuarios-env' showing its details, including a red exclamation mark icon indicating a 'Grave' (Severe) status, a 'Versão em execução' (Running version) of 'api\_usuarios-source', and a 'Plataforma' (Platform) of 'Correto 11 running on 64bit Amazon Linux 2/3.4.5'.

Nome do ambiente	Integridade	Nome do aplicativo	Data de criação	Última modificação	URL	Versões em execução
Apiclientes-env	Severe	api_clientes	15-03-2023 21:35:47 UTC-0300	15-03-2023 21:41:44 UTC-0300	Apiclientes-env.eba-v3p5tzs5.us-east-2.elasticbeanstalk.com	api_clientes-source
Apiusuarios-env	Severe	api_usuarios	15-03-2023 20:53:42 UTC-0300	15-03-2023 21:00:59 UTC-0300	Apiusuarios-env.eba-fp62bpyj.us-east-2.elasticbeanstalk.com	api_usuarios-source

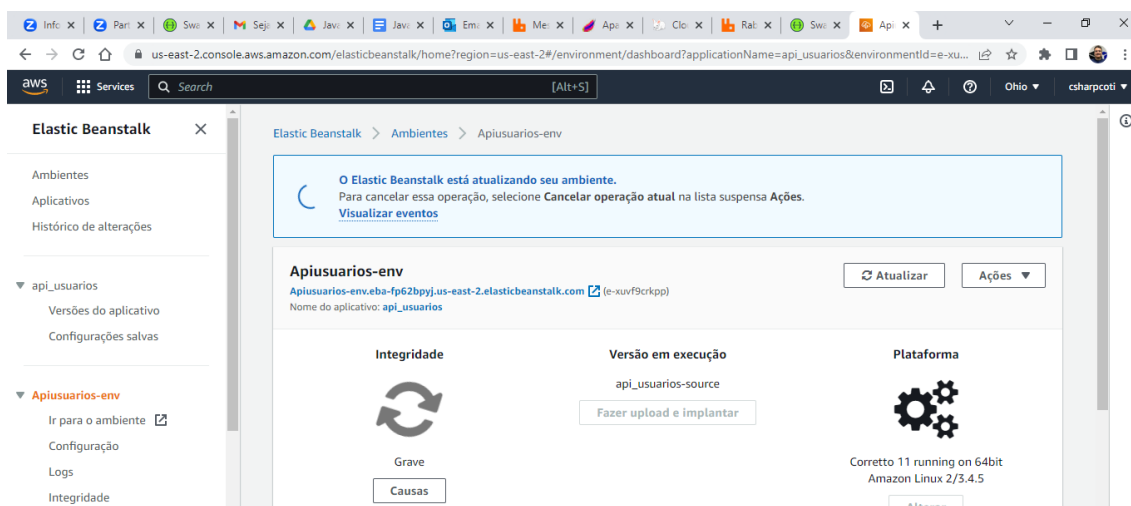
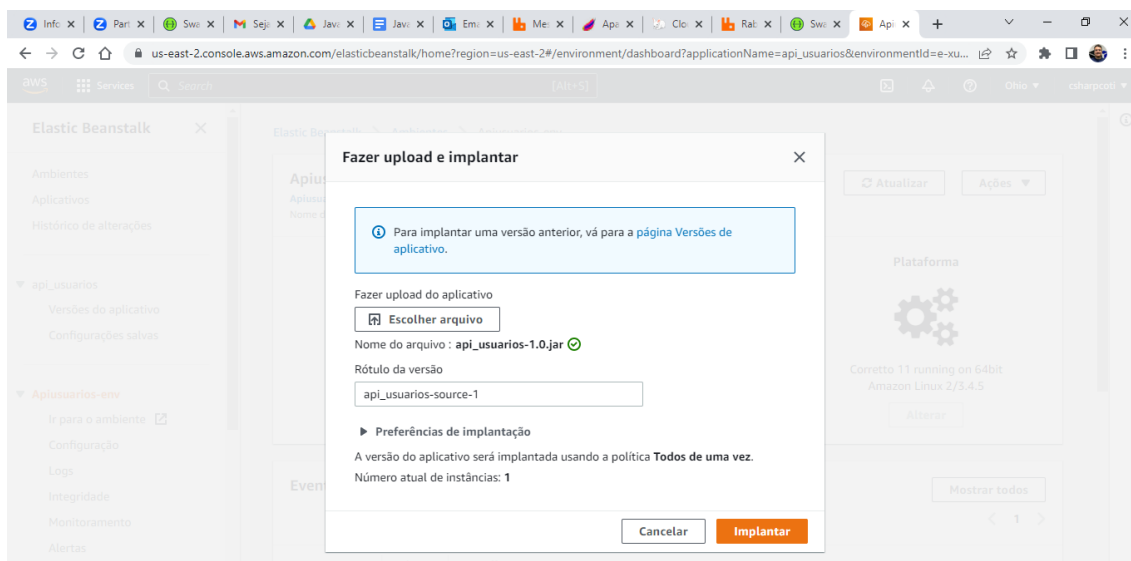


The screenshot shows the details of the 'Apiusuarios-env' environment. The left sidebar contains navigation links: Elastic Beanstalk, Ambientes, Aplicativos, and Histórico de alterações. The main content area is titled 'Apiusuarios-env' and displays the environment's details. The 'Versão em execução' (Running version) is highlighted with a red circle. Below the details, there is a section for 'Eventos recentes' (Recent events) with a table showing the event history.

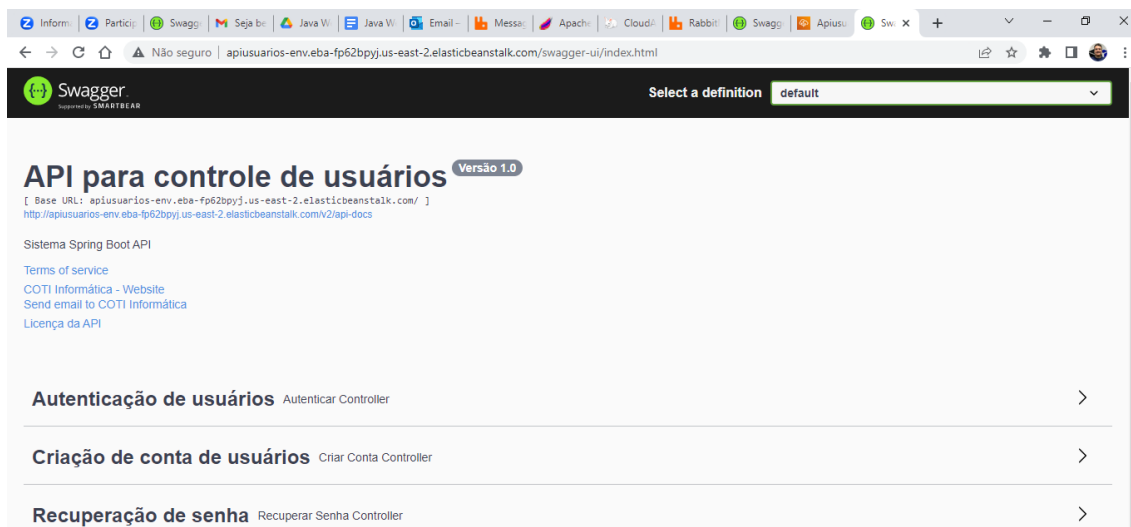
Tempo	Tipo	Detalhes



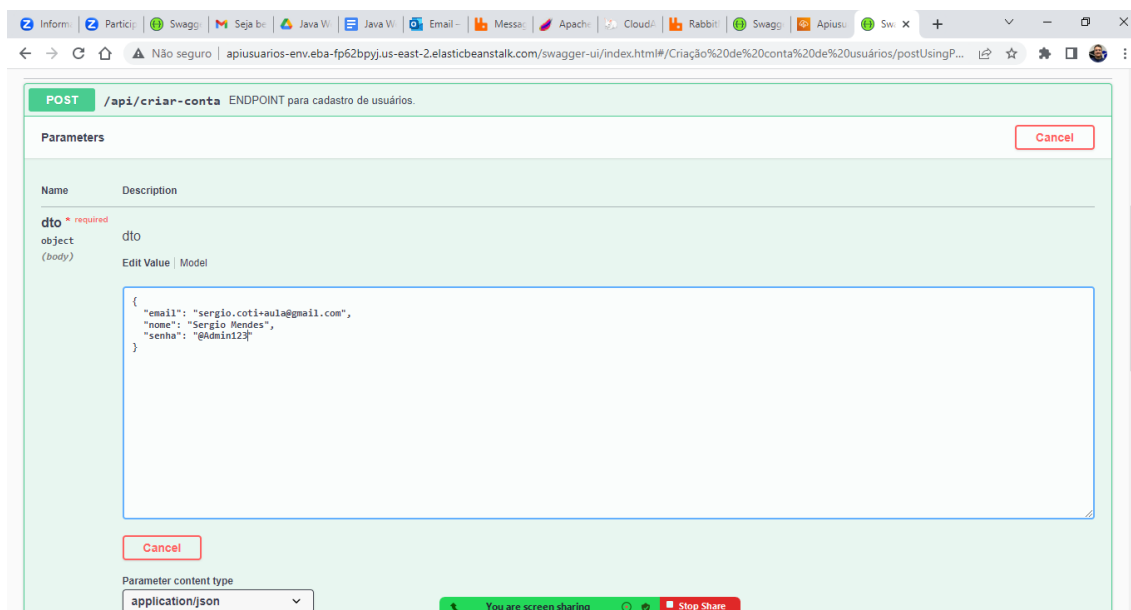
The screenshot shows the 'Fazer upload e implantar' (Upload and deploy) dialog box. The dialog box contains a message: 'Para implantar uma versão anterior, vá para a página Versões de aplicativo.' (To deploy a previous version, go to the application versions page). Below the message, there is a section for 'Fazer upload do aplicativo' (Upload application) with a button labeled 'Escolher arquivo' (Choose file). The file name 'Nome do arquivo: api\_usuarios-1.0.jar' is displayed below the button.



<http://apiusuarios-env.eba-fp62bpyj.us-east-2.elasticbeanstalk.com/swagger-ui/index.html>







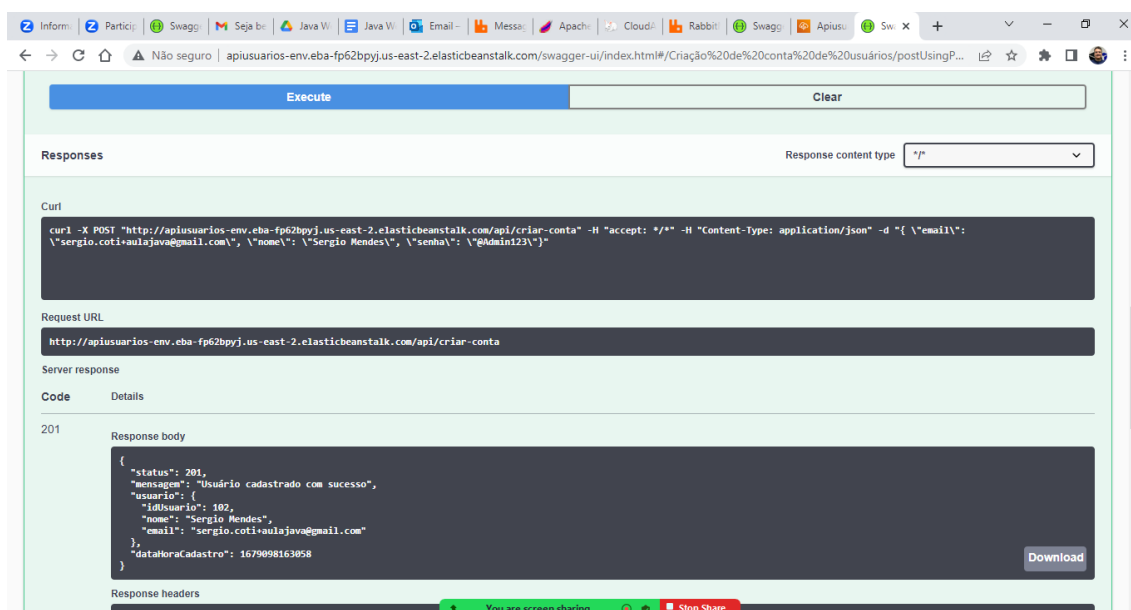
**POST** /api/criar-conta ENDPOINT para cadastro de usuários.

**Parameters**

Name	Description
dto	dto
object (body)	Edit Value   Model

```
{
  "email": "sergio.coti+aula@gmail.com",
  "nome": "Sergio Mendes",
  "senha": "@Admin123"
}
```

Parameter content type: application/json



**Execute** **Clear**

**Responses** Response content type: \*/\*

**Curl**

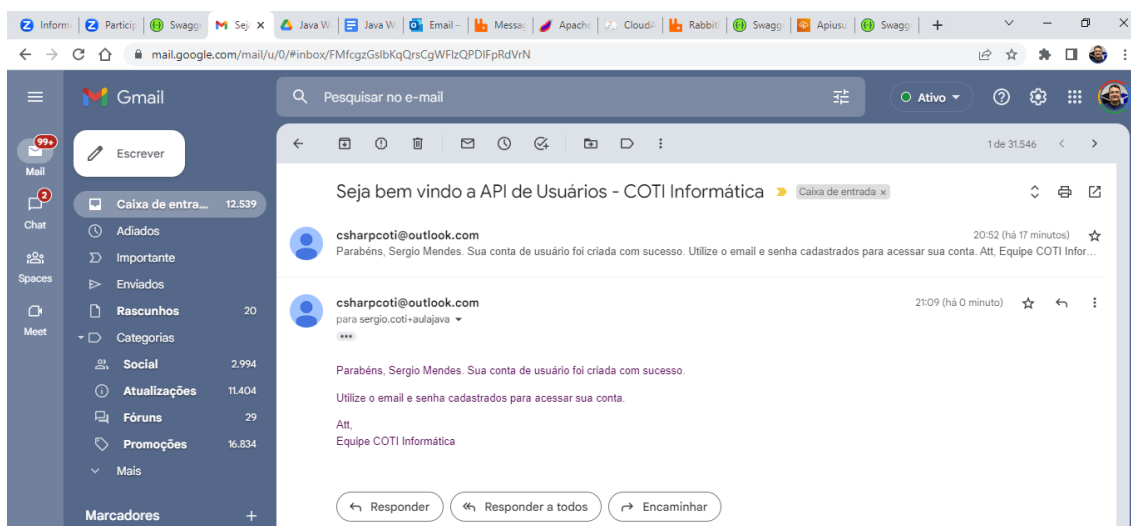
```
curl -X POST "http://apiusuarios-env.eba-fp62bpj.us-east-2.elasticbeanstalk.com/api/criar-conta" -H "accept: */*" -H "Content-Type: application/json" -d '{"email": "sergio.coti+aulajava@gmail.com", "nome": "Sergio Mendes", "senha": "@Admin123"}'
```

**Request URL**

```
http://apiusuarios-env.eba-fp62bpj.us-east-2.elasticbeanstalk.com/api/criar-conta
```

**Server response**

Code	Details
201	<p><b>Response body</b></p> <pre>{   "status": 201,   "mensagem": "Usuário cadastrado com sucesso",   "usuario": {     "idUsuario": 102,     "nome": "Sergio Mendes",     "email": "sergio.coti+aulajava@gmail.com"   },   "dataHoraCadastro": "1679098163058" }</pre> <p><b>Response headers</b></p>



**Gmail** Pesquisar no e-mail

Seja bem vindo a API de Usuários - COTI Informática

**csharpcoti@outlook.com** 20:52 (há 17 minutos)

Parabéns, Sergio Mendes. Sua conta de usuário foi criada com sucesso. Utilize o email e senha cadastrados para acessar sua conta. Att, Equipe COTI Inform...

**csharpcoti@outlook.com** 21:09 (há 0 minuto)

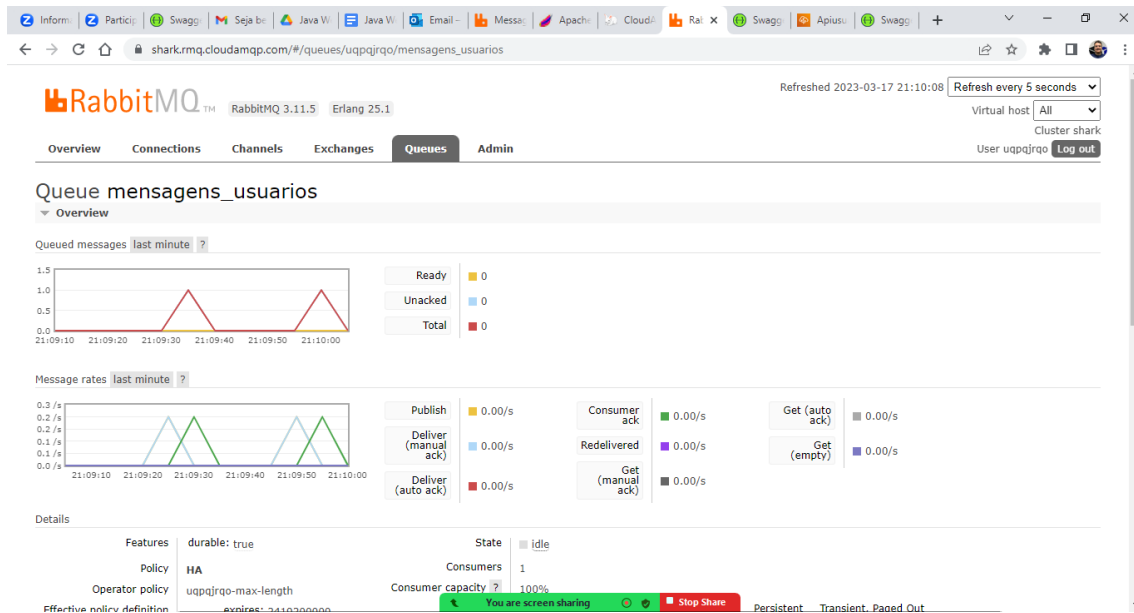
pará sergio.coti+aulajava

Parabéns, Sergio Mendes. Sua conta de usuário foi criada com sucesso.

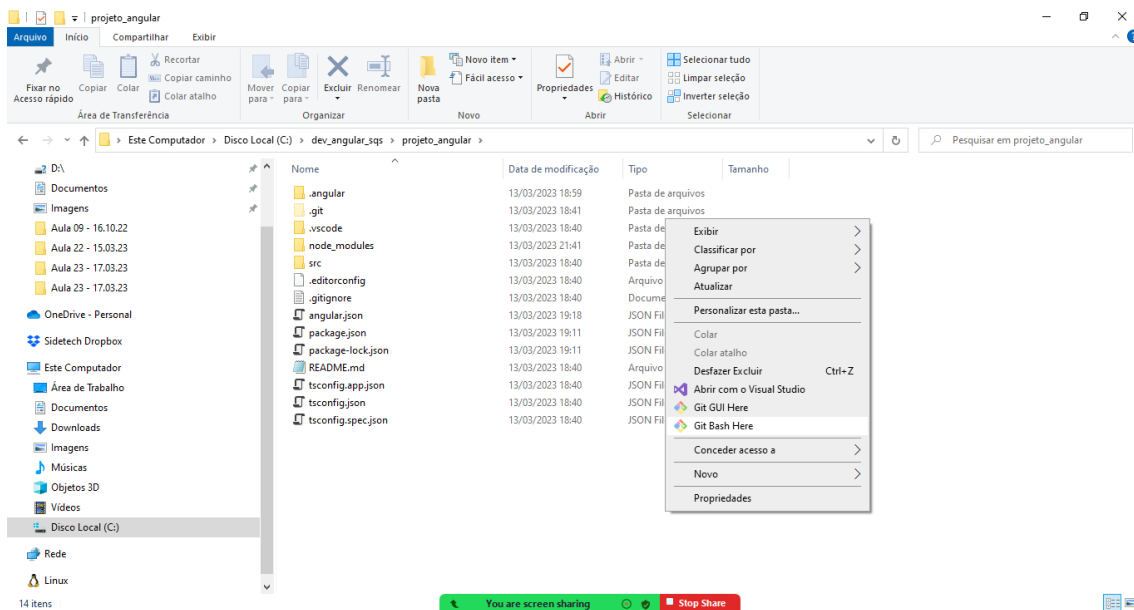
Utilize o email e senha cadastrados para acessar sua conta.

Att,  
Equipe COTI Informática

Responder Responder a todos Encaminhar

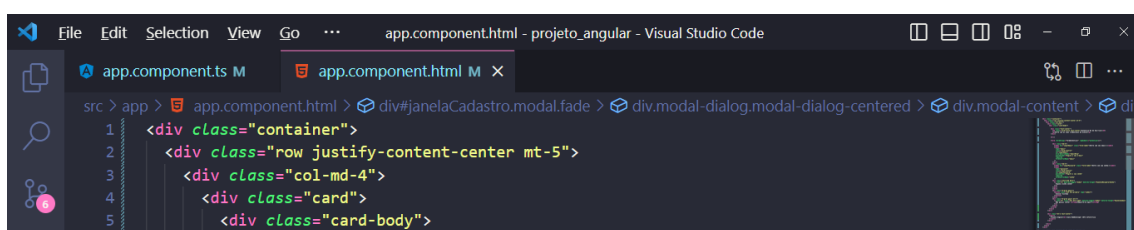


## Abrindo o projeto Angular:

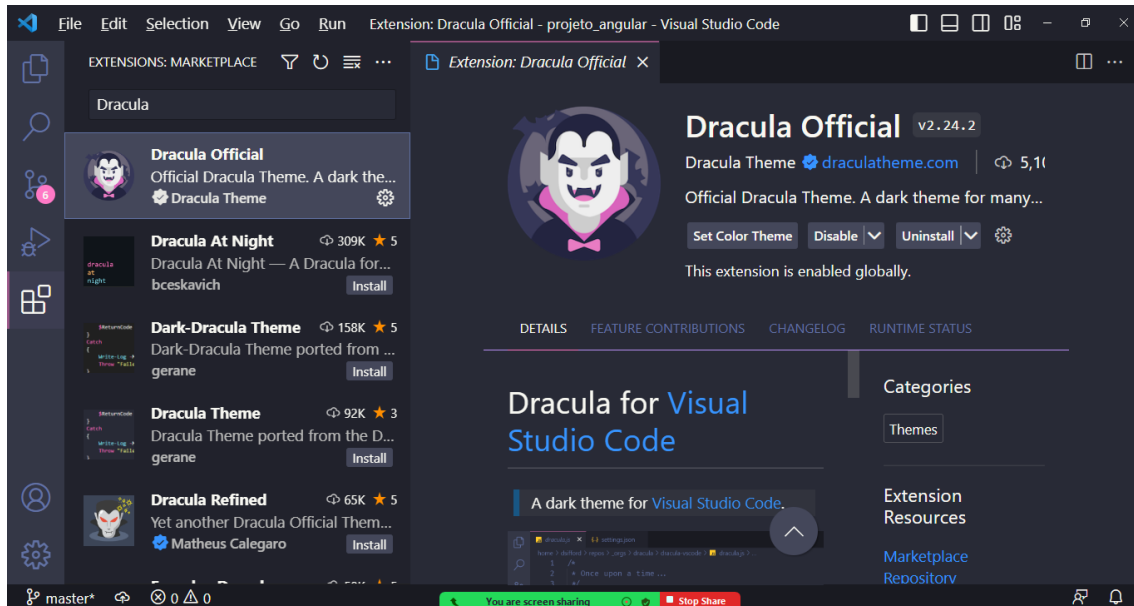


## Abrindo o Visual Studio Code:

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



## Instalando o tema "Dracula":



## Executando o projeto:

```
Samsung@DESKTOP-P9F6D9F MINGW64
/c/dev_angular_sqs/projeto_angular (master)
$ ng s
```

```
Samsung@DESKTOP-P9F6D9F MINGW64 /c/dev_angular_sqs/projeto_angular (master)
$ ng s
Your global Angular CLI version (15.2.3) is greater than your local version (15.2.2). The local Angular CLI version is used.

To disable this warning use "ng config -g cli.warnings.versionMismatch false".
- Generating browser application bundles (phase: setup)...
✓ Browser application bundle generation complete.

Initial Chunk Files | Names | Raw Size
vendor.js           | vendor | 2.09 MB
styles.css, styles.js | styles | 398.73 kB
polyfills.js        | polyfills | 314.28 kB
scripts.js          | scripts | 78.57 kB
main.js             | main | 17.38 kB
runtime.js          | runtime | 6.53 kB

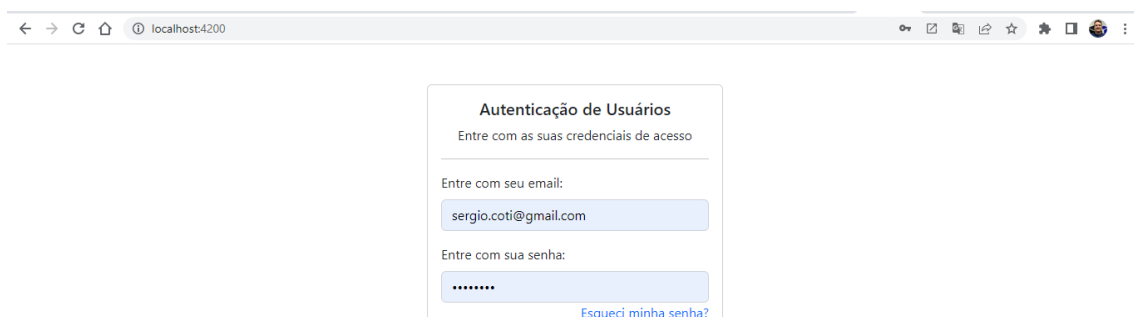
| Initial Total | 2.89 MB

Build at: 2023-03-18T00:17:27.946Z - Hash: 41c8c2440f9e3ec1 - Time: 5539ms

** Angular Live Development Server is listening on localhost:4200, open your browser on http://localhost:4200/ **

✓ Compiled successfully.
```

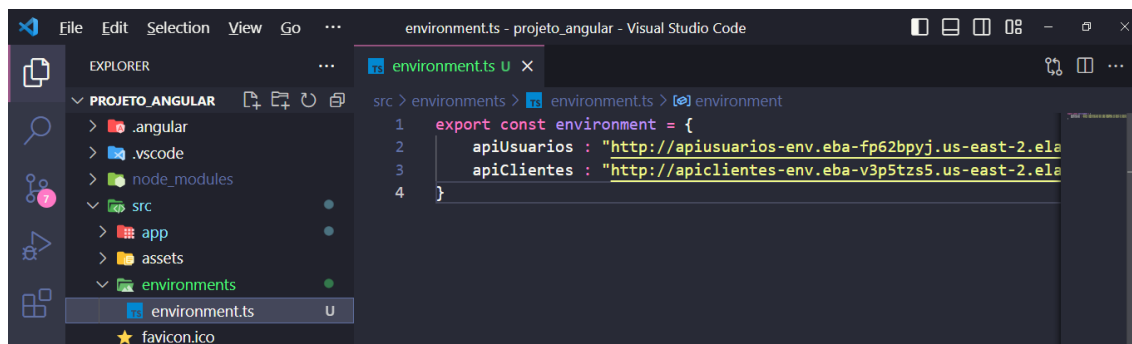
<http://localhost:4200/>



Criando um arquivo de configuração no projeto para definirmos os endereços dos servidores das APIs:

/src/environments/**environment.ts**

```
export const environment = {  
  apiUsuarios : "http://apiusuarios-env.eba-fp62bpyj.  
                .us-east-2.elasticbeanstalk.com/",  
  apiClientes : "http://apiclientes-env.eba-v3p5tzs5.  
                .us-east-2.elasticbeanstalk.com/"  
}
```



Acessando os endereços no componente:

/app.component.ts

```
import { Component } from '@angular/core';  
import { FormGroup, FormControl } from '@angular/forms';  
import { HttpClient } from '@angular/common/http';  
import { environment } from 'src/environments/environment';  
  
@Component({  
  selector: 'app-root',  
  templateUrl: './app.component.html',  
  styleUrls: ['./app.component.css']  
})  
export class AppComponent {  
  
  mensagem_criarconta: string = '';  
  
  constructor(  
    private httpClient: HttpClient //injeção de dependência  
  ) {  
  
  }  
}
```

```
//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 {
  //capturando os valores preenchidos no formulário
  var dados = this.formCriarConta.value;
  //executando a chamada para a API
  this.httpClient.post(environment.apiUrlUsuarios
    + '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;
        }
      }
    });
}

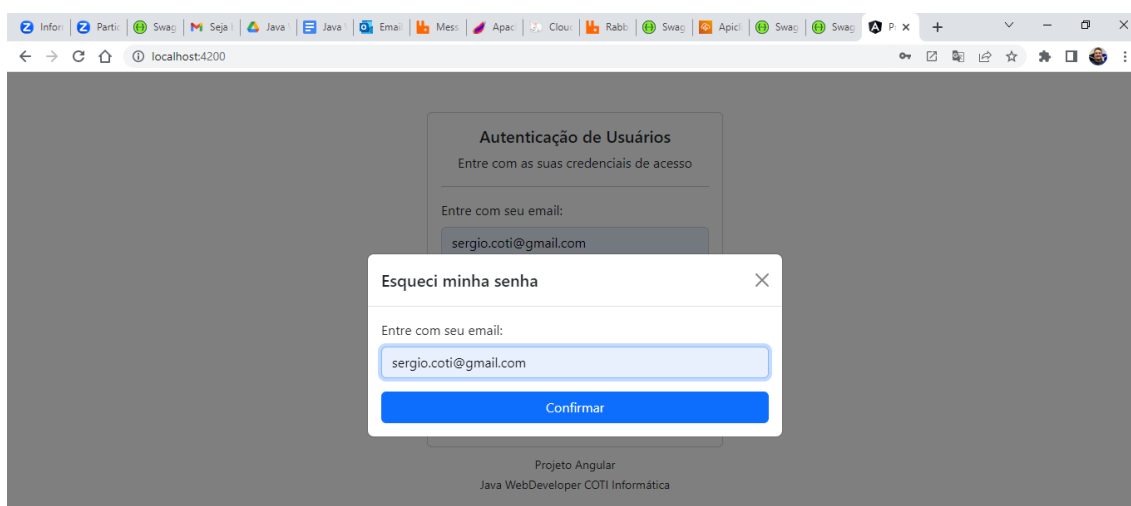
//função para capturar o SUBMIT do formulário
autenticar(): void {
  //capturando os valores preenchidos no formulário
```

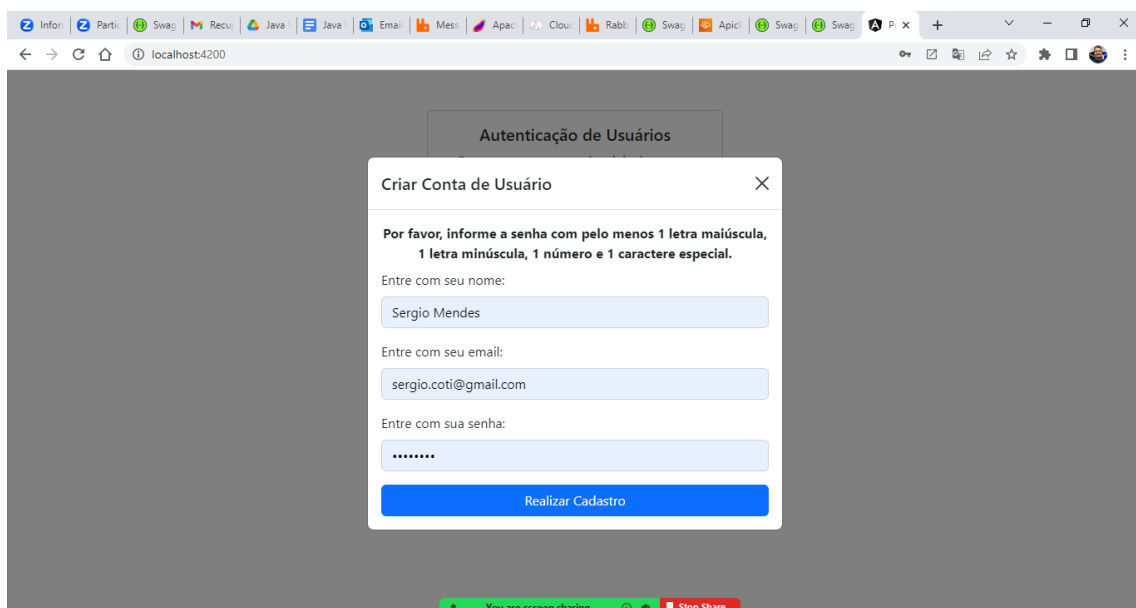
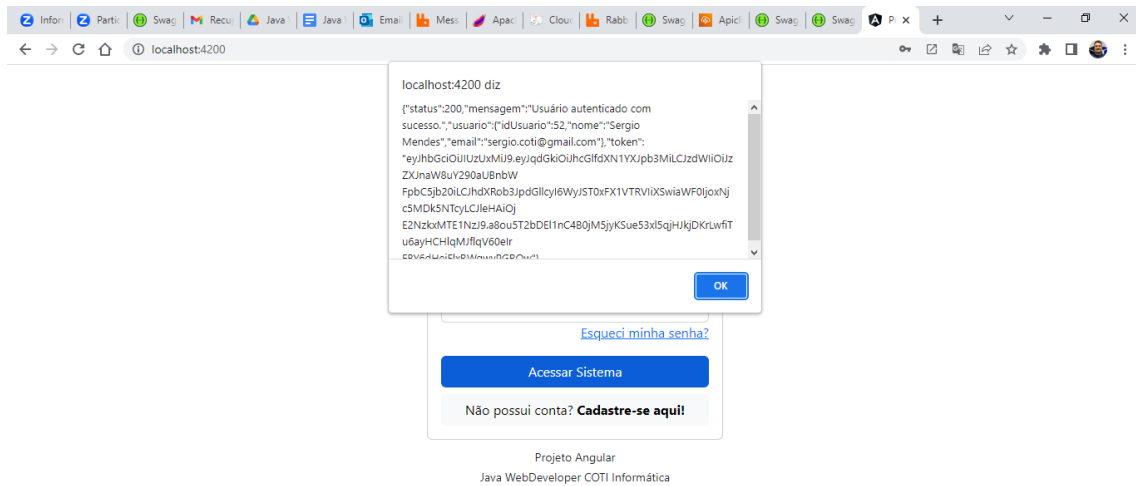
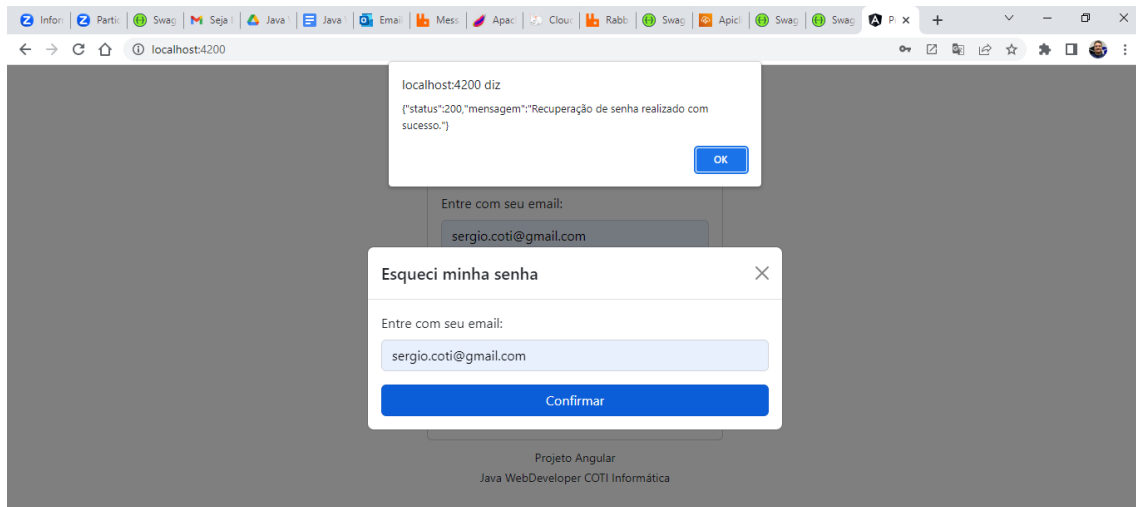
```

var dados = this.formAutenticar.value;
//executando a chamada para a API
this.httpClient.post(environment.apiUrlUsuarios
                    + 'api/autenticar', dados)
    .subscribe({ //capturando a resposta da API
        next: (data) => { //sucesso!
            alert(JSON.stringify(data));
        },
        error: (e) => { //erro!
            alert(JSON.stringify(e.error));
        }
    });
}

//função para capturar o SUBMIT do formulário
recuperarSenha(): void {
    //capturando os valores preenchidos no formulário
    var dados = this.formRecuperarSenha.value;
    //executando a chamada para a API
    this.httpClient.post(environment.apiUrlUsuarios
                        + 'api/recuperar-senha', dados)
        .subscribe({ //capturando a resposta da API
            next: (data) => { //sucesso!
                alert(JSON.stringify(data));
            },
            error: (e) => { //erro!
                alert(JSON.stringify(e.error));
            }
        });
}
}

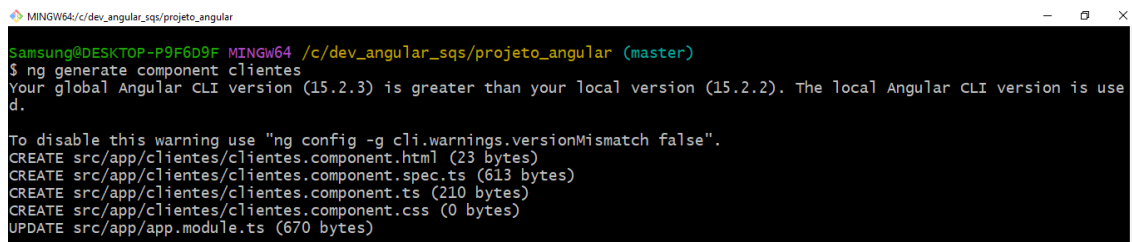
```



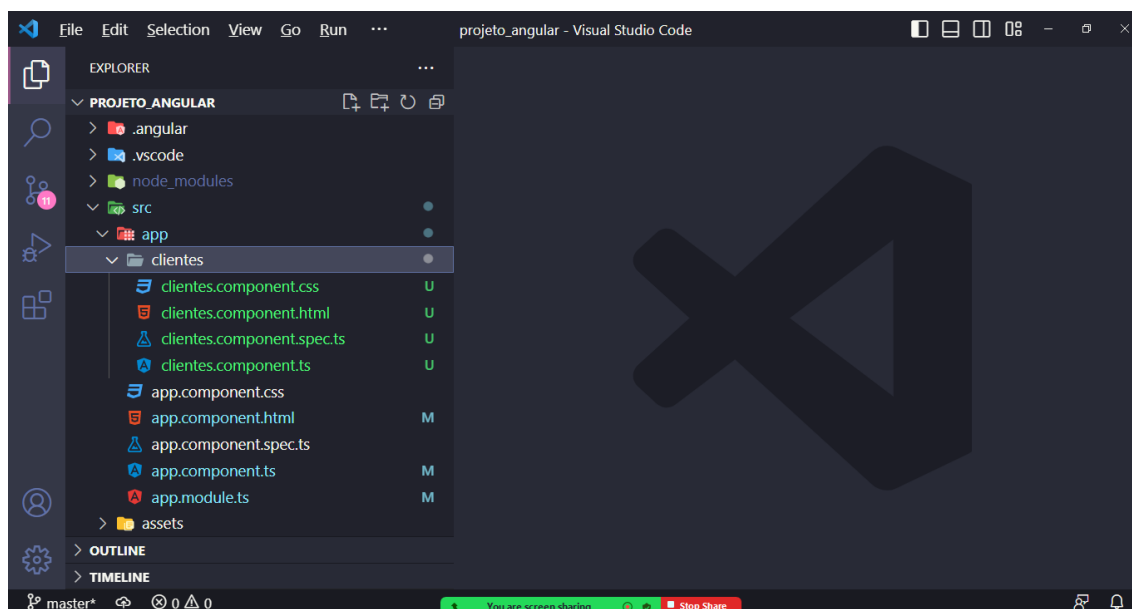


## Criando um componente para desenhar a página de gerenciamento de clientes:

```
Samsung@DESKTOP-P9F6D9F MINGW64  
/c/dev_angular_sqs/projeto_angular (master)  
$ ng generate component clientes
```



```
MINGW64/c/dev_angular_sqs/projeto_angular  
Samsung@DESKTOP-P9F6D9F MINGW64 /c/dev_angular_sqs/projeto_angular (master)  
$ ng generate component clientes  
Your global Angular CLI version (15.2.3) is greater than your local version (15.2.2). The local Angular CLI version is used.  
  
To disable this warning use "ng config -g cli.warnings.versionMismatch false".  
CREATE src/app/clientes/clientes.component.html (23 bytes)  
CREATE src/app/clientes/clientes.component.spec.ts (613 bytes)  
CREATE src/app/clientes/clientes.component.ts (210 bytes)  
CREATE src/app/clientes/clientes.component.css (0 bytes)  
UPDATE src/app/app.module.ts (670 bytes)
```



Continua...