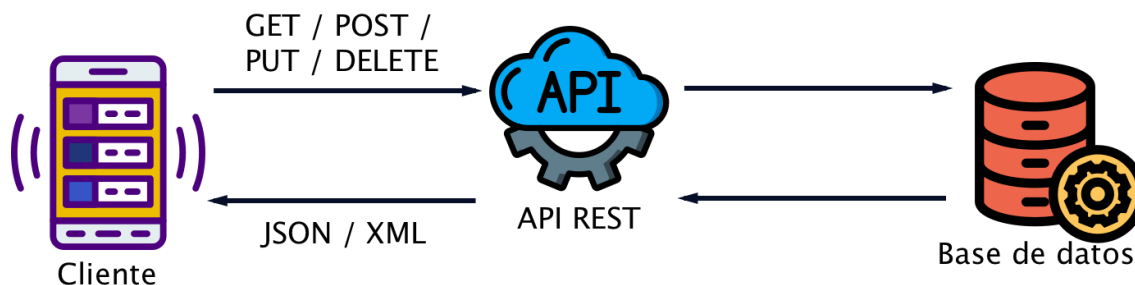


Api para controle de usuários:

Construindo um micro serviço (API REST) para controle de usuários, englobando cadastro, autenticação, recuperação de senha e alteração dos dados do usuário.



Inicialmente, a API terá os seguintes ENDPOINTS:

POST /api/autenticar

ENDPOINT para autenticação do usuário

REQUEST BODY:

- Email
- Senha

RESPONSE:

- Status
- Mensagem
- Usuario
 - Id do Usuário
 - Nome
 - Email
- Token (chave de autenticação)

POST /api/criar-conta

ENDPOINT para cadastro do usuário

REQUEST BODY:

- Nome
- Email
- Senha

RESPONSE:

- Status
- Mensagem
- Usuario
 - Id do Usuário
 - Nome
 - Email

POST /api/recuperar-senha

ENDPOINT para recuperação da senha do usuário

REQUEST BODY:

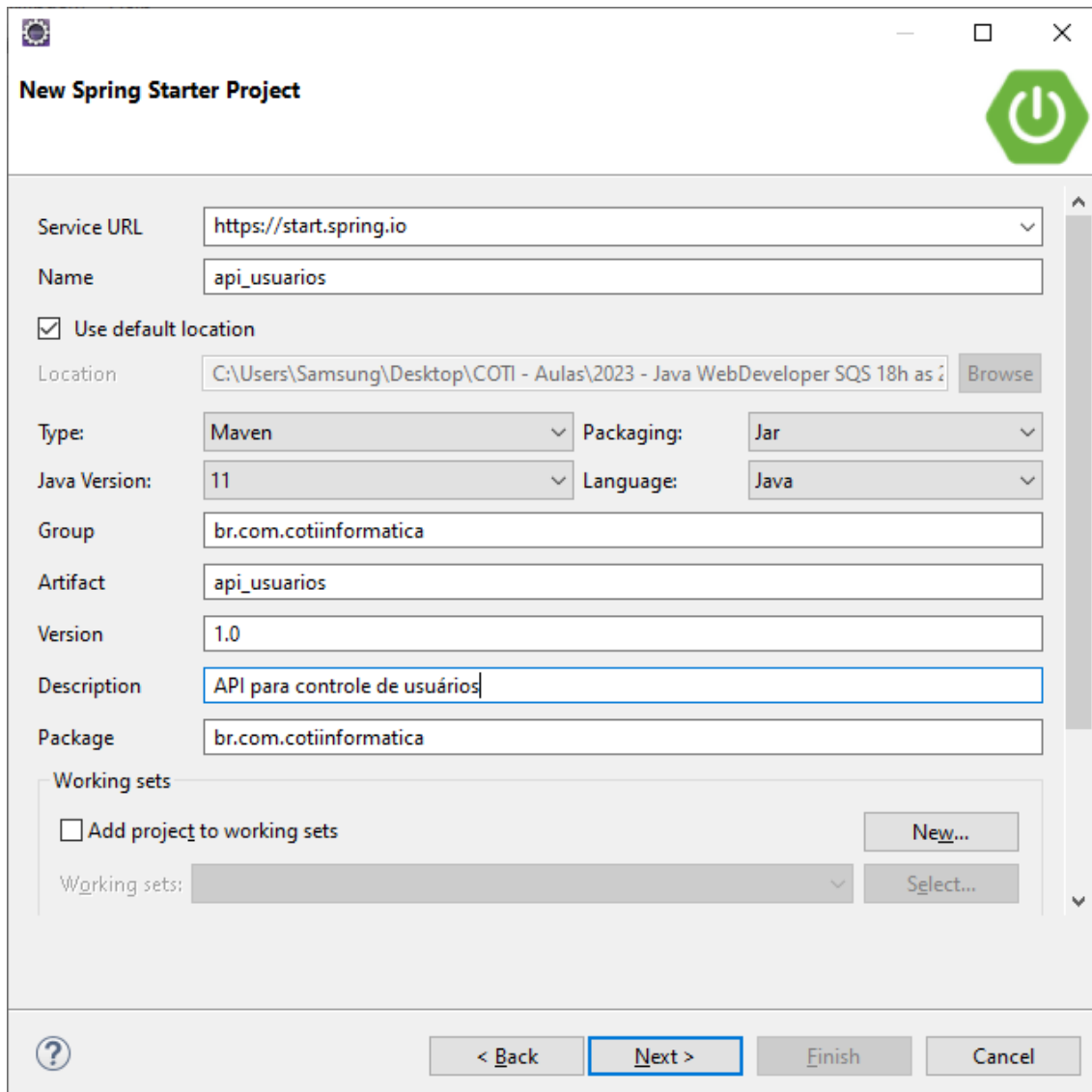
. Email

RESPONSE:

- Status
- Mensagem

Novo projeto:

- File / New / Other
- Spring starter project



New Spring Starter Project

Service URL:

Name:

☒ Use default location

Location:

Type: Packaging:

Java Version: Language:

Group:

Artifact:

Version:

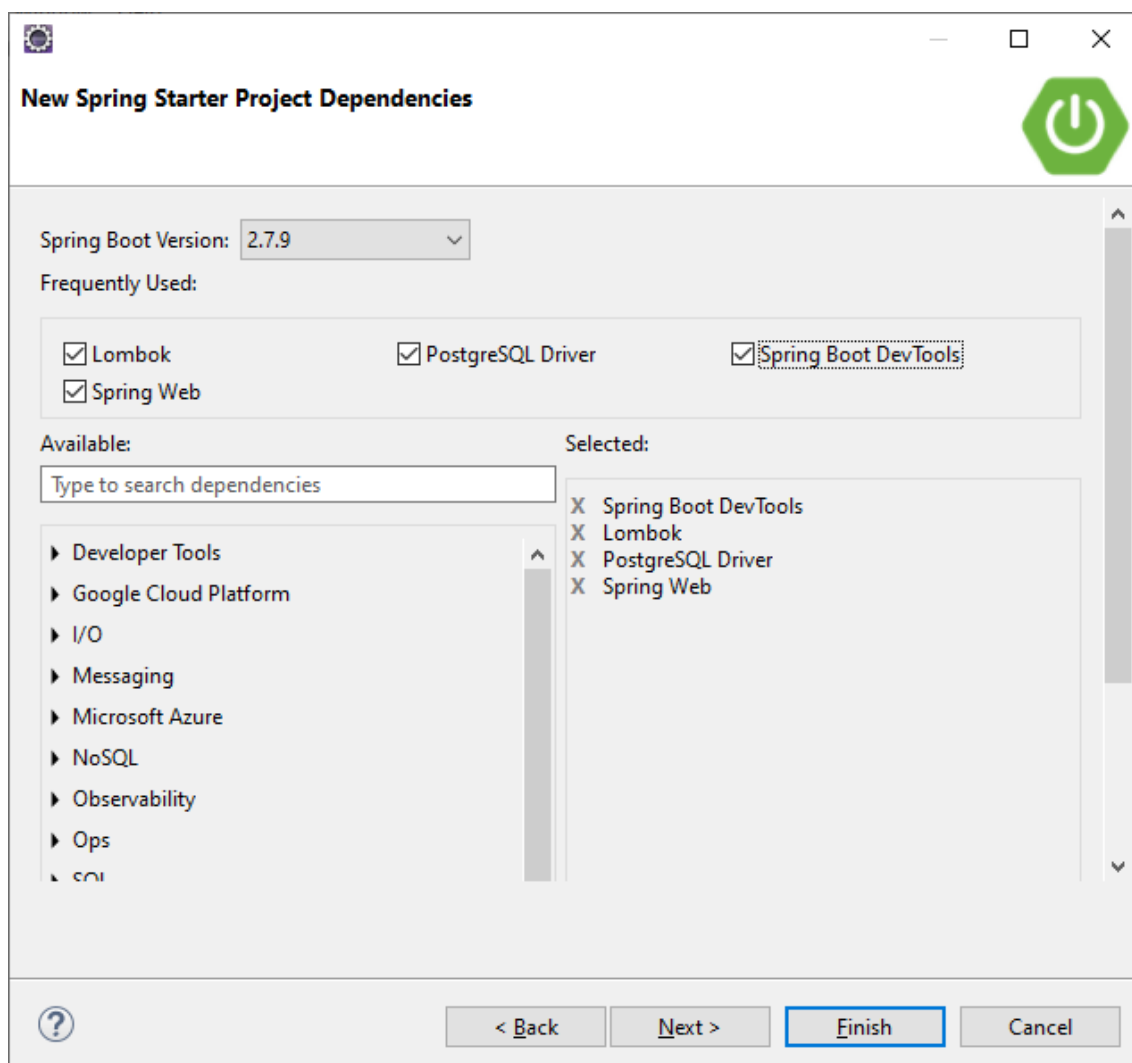
Description:

Package:

Working sets

☐ Add project to working sets

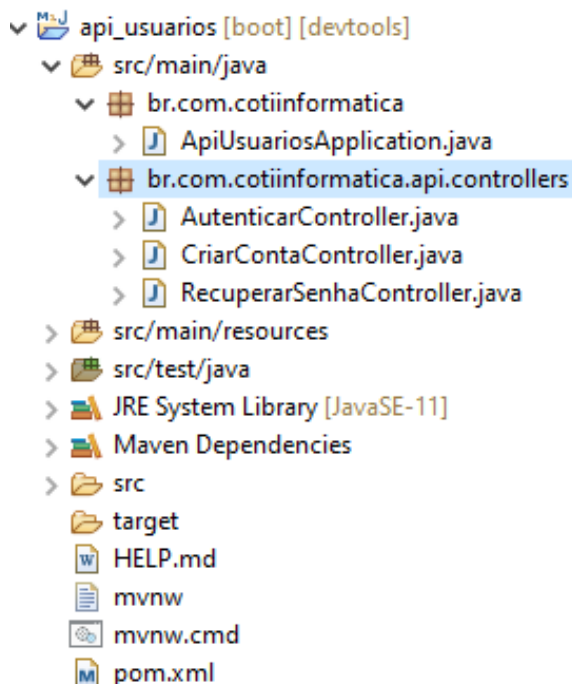
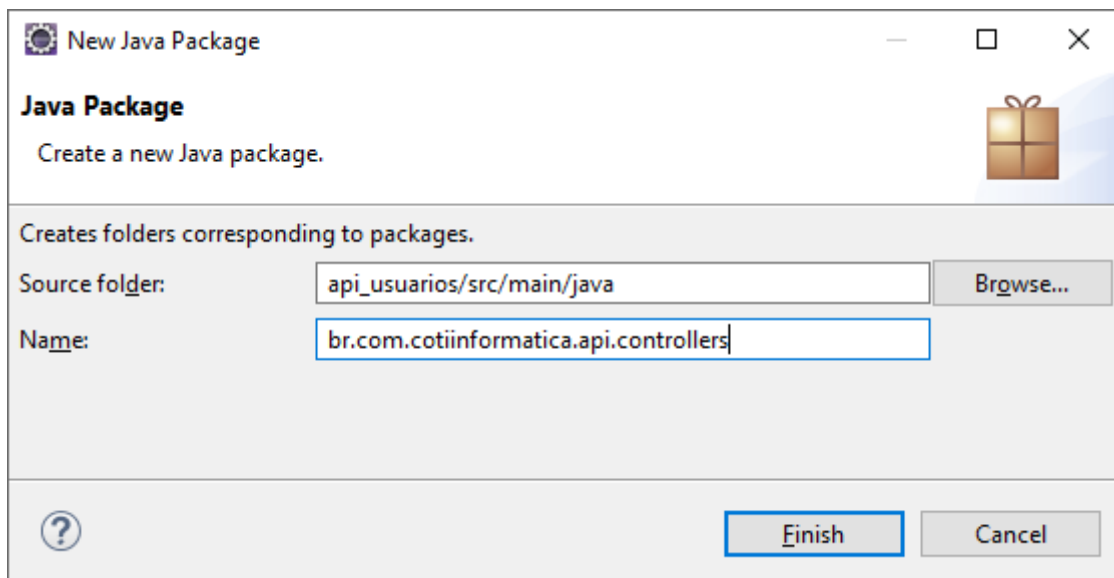
Working sets:



Projeto criado:

- api_usuarios [boot] [devtools]
 - src/main/java
 - br.com.cotiinformatica
 - ApiUsuariosApplication.java
 - src/main/resources
 - src/test/java
 - JRE System Library [JavaSE-11]
 - Maven Dependencies
 - src
 - target
 - HELP.md
 - mvnw
 - mvnw.cmd
 - pom.xml

Criando os endpoints do projeto:



/AutenticarController.java

Criando o ENDPOINT para autenticação do usuário.

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

import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class AutenticarController {
```

```
@PostMapping("/api/autenticar")
public ResponseEntity<String> post() {
    return null;
}

package br.com.cotiinformatica.api.controllers;

import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class CriarContaController {

    @PostMapping("/api/criar-conta")
    public ResponseEntity<String> post() {
        return null;
    }
}

package br.com.cotiinformatica.api.controllers;

import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class RecuperarSenhaController {

    @PostMapping("/api/recuperar-senha")
    public ResponseEntity<String> post() {
        return null;
    }
}
```

DATA TRANSFER OBJECT (DTOs)

Nome dado às classes JAVABEANS utilizadas para transferência de dados entre a API e a sua aplicação cliente. Ou seja, os dados de REQUISIÇÃO e RESPOSTA.

```
api_usuarios [boot] [devtools]
├── src/main/java
│   ├── br.com.cotiinformatica
│   ├── br.com.cotiinformatica.api.controllers
│   └── br.com.cotiinformatica.application.dtos
│       ├── PostAutenticarDTO.java
│       ├── PostCriarContaDTO.java
│       └── PostRecuperarSenhaDTO.java
```

```
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 PostAutenticarDTO {  
  
    private String email;  
    private String senha;  
}
```

```
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 PostCriarContaDTO {  
  
    private String nome;  
    private String email;  
    private String senha;  
}
```

```
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 PostRecuperarSenhaDTO {

    private String email;

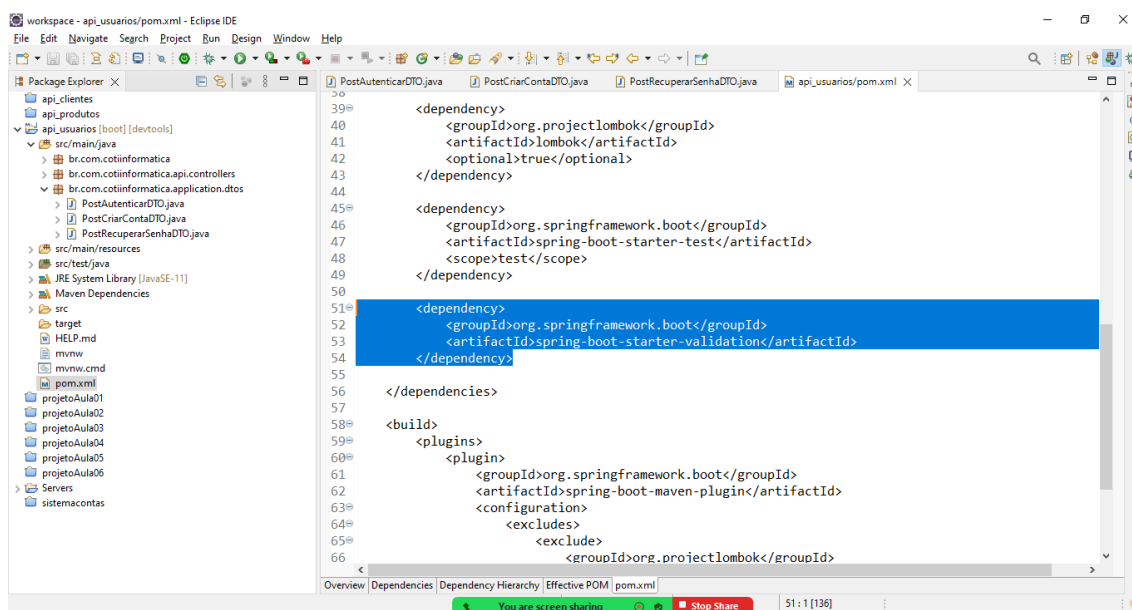
}
```

Bean Validations

Biblioteca para mapeamento de validações de dados em classes JavaBean. Vamos utilizar esta biblioteca para validar os dados dos DTOs que a API irá receber nas requisições dos seus ENDPOINTS.

/pom.xml

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



/dtos/PostAutenticarDTO.java

Mapeando as validações.

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

import javax.validation.constraints.Email;
import javax.validation.constraints.NotBlank;
import javax.validation.constraints.Size;

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

```
import lombok.Setter;
import lombok.ToString;

@Setter
@Getter
@NoArgsConstructor
@AllArgsConstructor
@ToString
public class PostAutenticarDTO {

    @Email(message = "Por favor, informe
                    um endereço de email válido.")
    @NotBlank(message = "Por favor, informe o email
                    de acesso do usuário.")
    private String email;

    @Size(min = 8, max = 20, message = "Informe a senha
                    com 8 a 20 caracteres.")
    @NotBlank(message = "Por favor, informe a senha
                    de acesso do usuário.")
    private String senha;
}
```

/dtos/**PostCriarContaDTO.java**

Mapeando as validações.

Para validação de senha forte:

- Pelo menos uma letra minúscula: `(?=.*[a-z])`
- Pelo menos uma letra maiúscula: `(?=.*[A-Z])`
- Pelo menos um dígito: `(?=.*\d)`
- Pelo menos um caractere especial (entre `@$!%*?&`): `(?=.*[@$!%*?&])`
- O comprimento da senha deve ser de pelo menos 8 caracteres: `[A-Za-z\d@$!%*?&]{8,}`

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

```
import javax.validation.constraints.Email;
import javax.validation.constraints.NotBlank;
import javax.validation.constraints.Pattern;
import javax.validation.constraints.Size;
```

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



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

    @Pattern(regexp = "^[A-Za-zÀ-Üà-ü\\s]{6,150}$",
            message = "Por favor, informe um nome
            válido de 6 a 150 caracteres.")
    @NotBlank(message = "Por favor, informe o nome do usuário.")
    private String nome;

    @Email(message = "Por favor, informe
            um endereço de email válido.")
    @NotBlank(message = "Por favor, informe o email do usuário.")
    private String email;

    @Pattern(regexp = "^(?=.*[a-z])(?=.*[A-Z])(?=.*\\d)
            (?!.*[@$_!*?&])[A-Za-z\\d@$!%*?&]{8,}$",
            message = "Por favor, informe a senha com pelo menos
            1 letra maiúscula, 1 letra minúscula, 1 número
            e 1 caractere especial.")
    @Size(min = 8, max = 20, message = "Informe a senha
            com 8 a 20 caracteres.")
    @NotBlank(message = "Por favor, informe a senha do usuário.")
    private String senha;
}
```

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

```
import javax.validation.constraints.Email;
import javax.validation.constraints.NotBlank;
```

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

```
@Setter
@Getter
@NoArgsConstructor
@AllArgsConstructor
@ToString
```

```
public class PostRecuperarSenhaDTO {  
  
    @Email(message = "Por favor, informe  
                um endereço de email válido.")  
    @NotBlank(message = "Por favor, informe  
                o email do usuário.")  
    private String email;  
  
}
```

Voltando nos controladores para usarmos os DTOs:

```
package br.com.cotiinformatica.api.controllers;  
  
import javax.validation.Valid;  
  
import org.springframework.http.ResponseEntity;  
import org.springframework.web.bind.annotation.PostMapping;  
import org.springframework.web.bind.annotation.RequestBody;  
import org.springframework.web.bind.annotation.RestController;  
  
import br.com.cotiinformatica.application.dtos.PostAutenticarDTO;  
  
@RestController  
public class AutenticarController {  
  
    @PostMapping("/api/autenticar")  
    public ResponseEntity<String> post  
        (@Valid @RequestBody PostAutenticarDTO dto) {  
        return null;  
    }  
  
}  
  
package br.com.cotiinformatica.api.controllers;  
  
import javax.validation.Valid;  
  
import org.springframework.http.ResponseEntity;  
import org.springframework.web.bind.annotation.PostMapping;  
import org.springframework.web.bind.annotation.RequestBody;  
import org.springframework.web.bind.annotation.RestController;  
  
import br.com.cotiinformatica.application.dtos.PostCriarContaDTO;  
  
@RestController  
public class CriarContaController {  
  
    @PostMapping("/api/criar-conta")  
    public ResponseEntity<String> post  
        (@Valid @RequestBody PostCriarContaDTO dto) {  
        return null;  
    }  
  
}
```

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

import javax.validation.Valid;

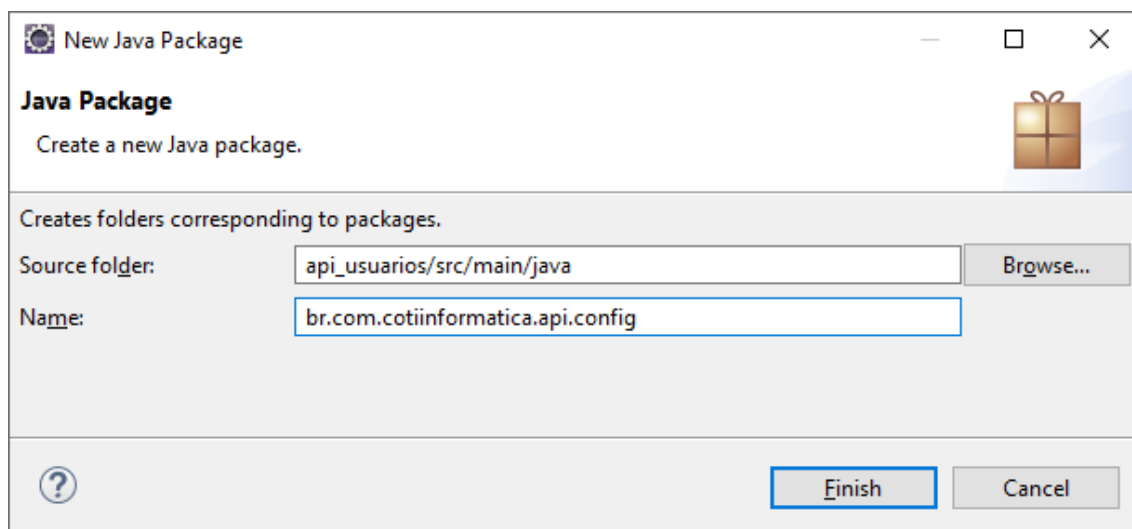
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RestController;

import br.com.cotiinformatica.application.dtos.PostRecuperarSenhaDTO;

@RestController
public class RecuperarSenhaController {

    @PostMapping("/api/recuperar-senha")
    public ResponseEntity<String> post
        (@Valid @RequestBody PostRecuperarSenhaDTO dto) {
        return null;
    }
}
```

Criando uma classe para tratar e formatar o conteúdo dos erros retornados pela API derivados do BEAN VALIDATIONS.



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

import java.util.LinkedHashMap;
import java.util.List;
import java.util.Map;
import java.util.stream.Collectors;

import org.springframework.http.HttpHeaders;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.MethodArgumentNotValidException;
import org.springframework.web.bind.annotation.ControllerAdvice;
import org.springframework.web.context.request.WebRequest;
```

```
import
org.springframework.web.servlet.mvc.method.annotation.ResponseEntityEx
ceptionHandler;

@ControllerAdvice
public class ErrorHandlerConfig extends ResponseEntityExceptionHandler
{

    @Override
    protected ResponseEntity<Object> handleMethodArgumentNotValid
        (MethodArgumentNotValidException ex,
         HttpHeaders headers, HttpStatus status,
         WebRequest request) {

        Map<String, Object> body = new LinkedHashMap<>();
        body.put("status", status.value());

        List<String> errors = ex.getBindingResult()
            .getFieldErrors().stream().map
            (x -> x.getDefaultMessage())
            .collect(Collectors.toList());

        body.put("errors", errors);

        return new ResponseEntity<>(body, headers, status);
    }
}
```

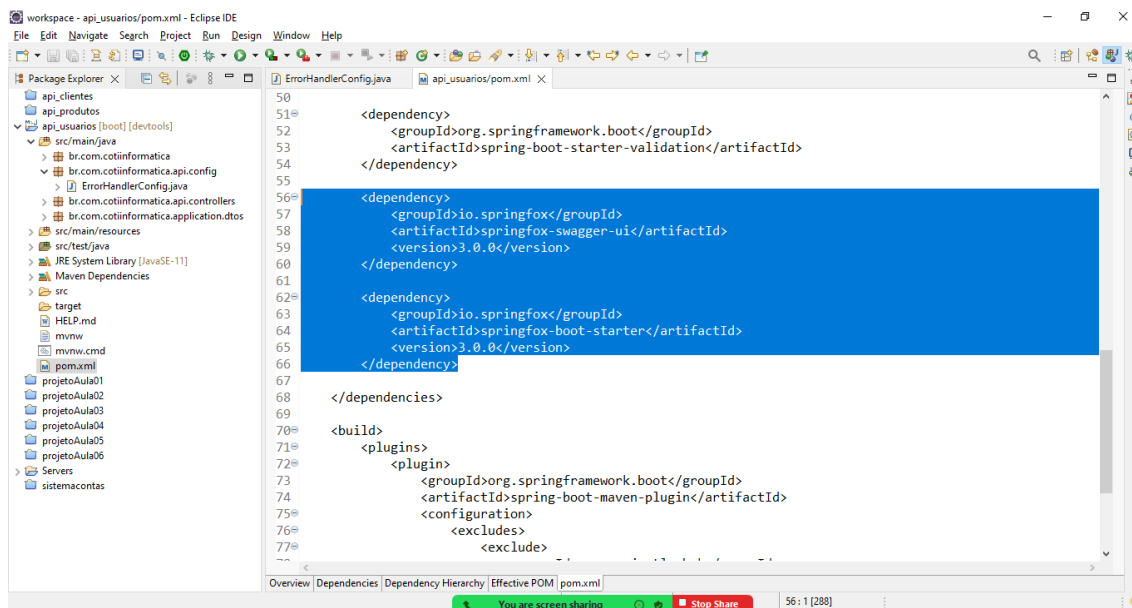
Configurando o Swagger:

Gerando a documentação da API.

- /pom.xml

```
<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>
```



Criando uma classe para configurarmos a documentação do Swagger:

```
package br.com.cotiinformatica.api.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 para controle de usuários",  
        "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()  
    );  
}  
}
```

Documentando os códigos dos controladores:

```
package br.com.cotiinformatica.api.controllers;  
  
import javax.validation.Valid;  
  
import org.springframework.http.ResponseEntity;  
import org.springframework.web.bind.annotation.PostMapping;  
import org.springframework.web.bind.annotation.RequestBody;  
import org.springframework.web.bind.annotation.RestController;  
  
import br.com.cotiinformatica.application.dtos.PostAutenticarDTO;  
import io.swagger.annotations.Api;  
import io.swagger.annotations.ApiOperation;  
  
@Api(tags = "Autenticação de usuários")  
@RestController  
public class AutenticarController {  
  
    @ApiOperation("ENDPOINT para autenticação  
de usuários e obtenção de Token.")  
    @PostMapping("/api/autenticar")  
    public ResponseEntity<String> post(@Valid @RequestBody PostAutenticarDTO dto) {  
        return null;  
    }  
}  
  
package br.com.cotiinformatica.api.controllers;  
  
import javax.validation.Valid;  
  
import org.springframework.http.ResponseEntity;  
import org.springframework.web.bind.annotation.PostMapping;  
import org.springframework.web.bind.annotation.RequestBody;  
import org.springframework.web.bind.annotation.RestController;  
  
import br.com.cotiinformatica.application.dtos.PostCriarContaDTO;  
import io.swagger.annotations.Api;  
import io.swagger.annotations.ApiOperation;
```

@Api(tags = "Criação de conta de usuários")

@RestController

public class CriarContaController {

@ApiOperation("ENDPOINT para cadastro de usuários.")

@PostMapping("/api/criar-conta")

public ResponseEntity<String> post(@Valid @RequestBody PostCriarContaDTO dto)

{

return null;

}

}

package br.com.cotiinformatica.api.controllers;

import javax.validation.Valid;

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.PostMapping;

import org.springframework.web.bind.annotation.RequestBody;

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

import br.com.cotiinformatica.application.dtos.PostRecuperarSenhaDTO;

import io.swagger.annotations.Api;

import io.swagger.annotations.ApiOperation;

@Api(tags = "Recuperação de senha")

@RestController

public class RecuperarSenhaController {

@ApiOperation("ENDPOINT para recuperação da senha de acesso do usuário.")

@PostMapping("/api/recuperar-senha")

public ResponseEntity<String> post

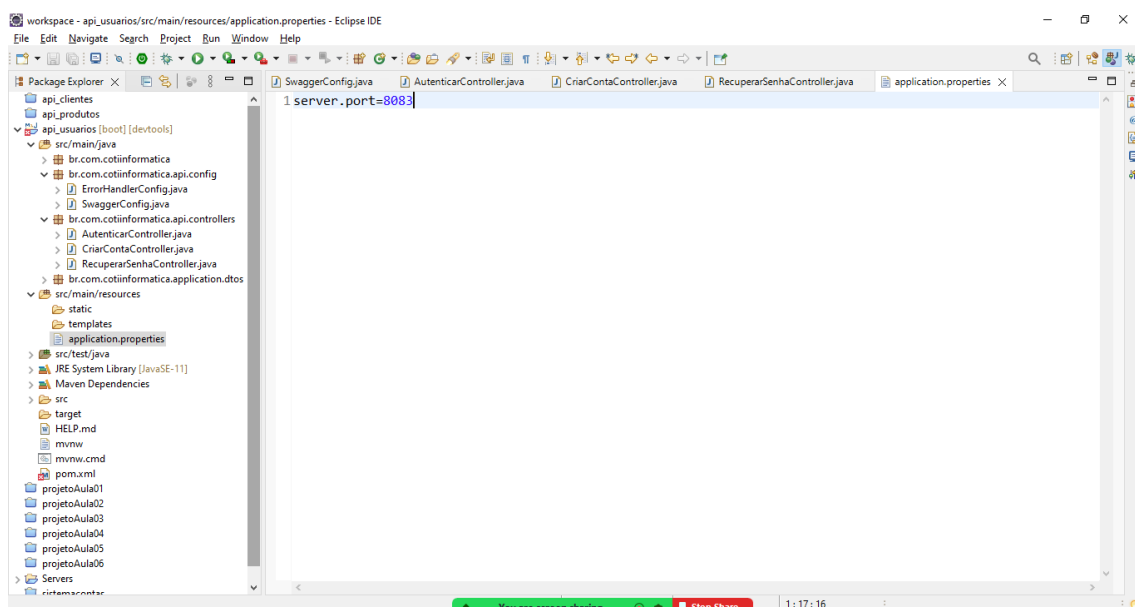
(@Valid @RequestBody PostRecuperarSenhaDTO dto) {

return null;

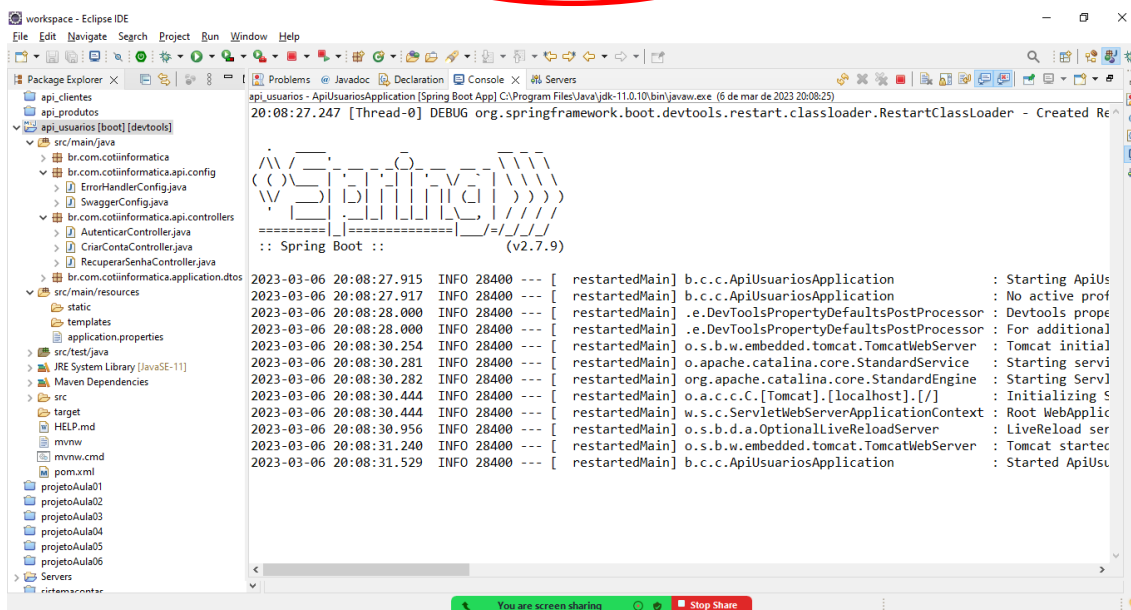
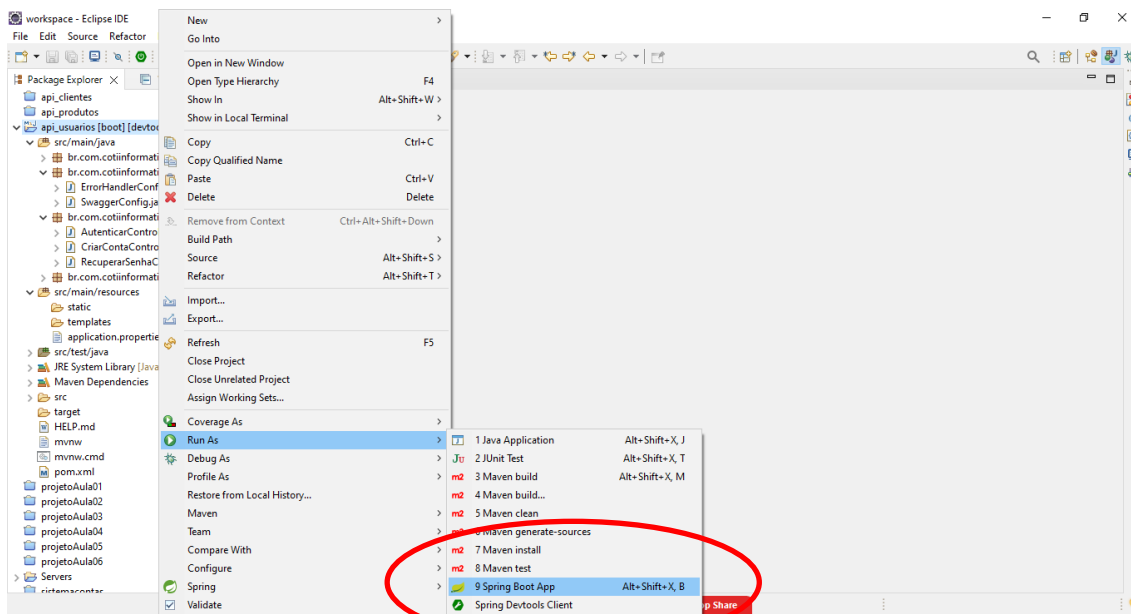
}

}

/application.properties



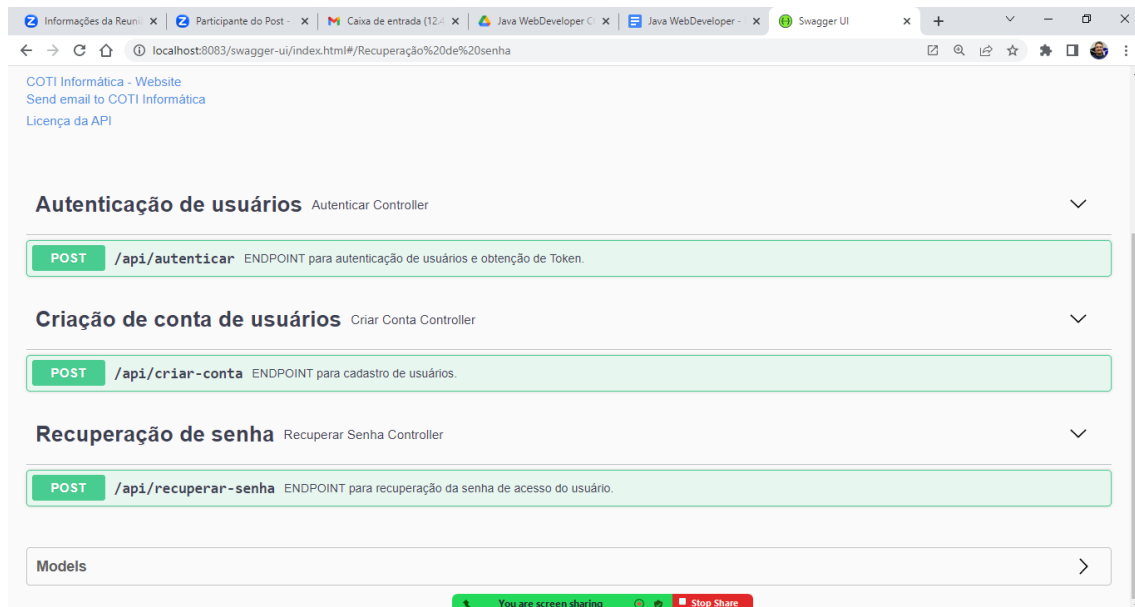
Executando:



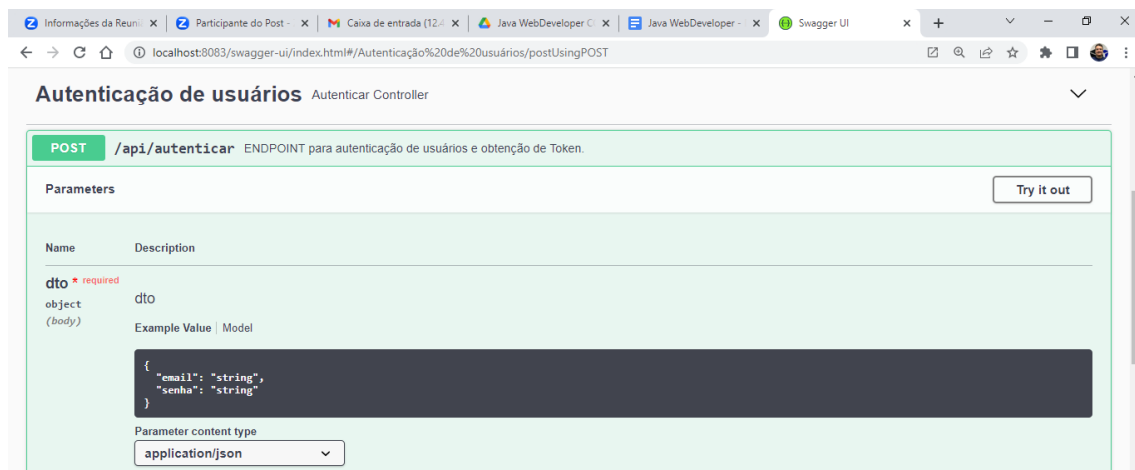


Swagger UI interface showing the API for user control. The title is "API para controle de usuários" (Versão 1.0). The base URL is "http://localhost:8083/v2/api-docs". The API is described as "Sistema Spring Boot API". Links for "Terms of service", "COTI Informática - Website", "Send email to COTI Informática", and "Licença da API" are provided. The API endpoints are listed:

- Autenticação de usuários** (Autenticar Controller)
- Criação de conta de usuários** (Criar Conta Controller)
- Recuperação de senha** (Recuperar Senha Controller)



Swagger UI interface showing the details of the POST endpoint for user authentication. The endpoint is "/api/autenticar" (ENDPOINT para autenticação de usuários e obtenção de Token). The method is POST. The description is "ENDPOINT para autenticação de usuários e obtenção de Token." The endpoint is also listed under "Criação de conta de usuários" and "Recuperação de senha".

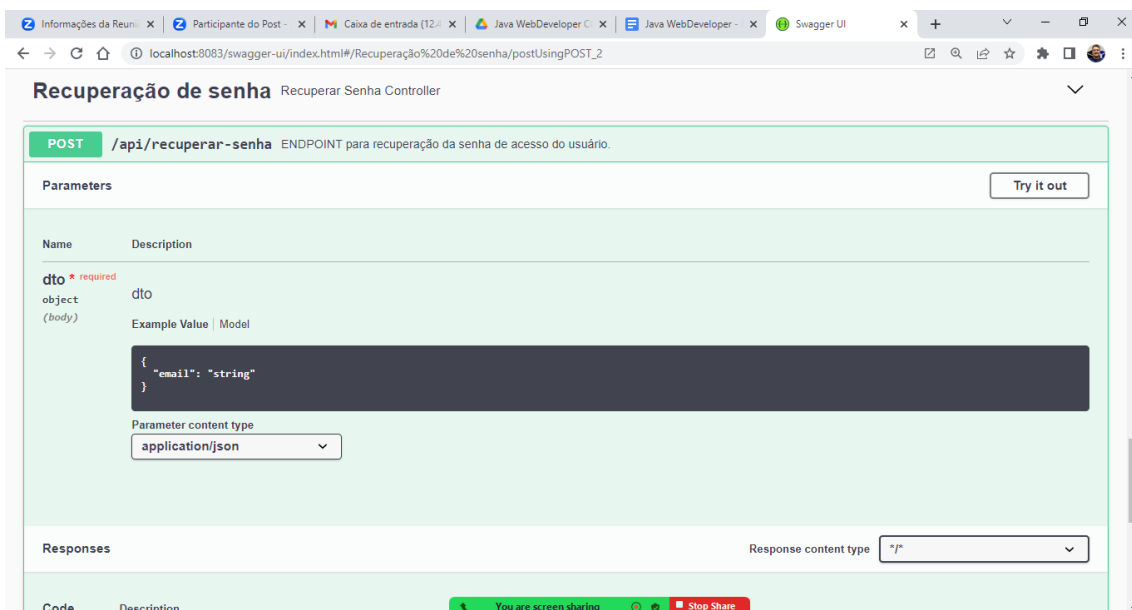
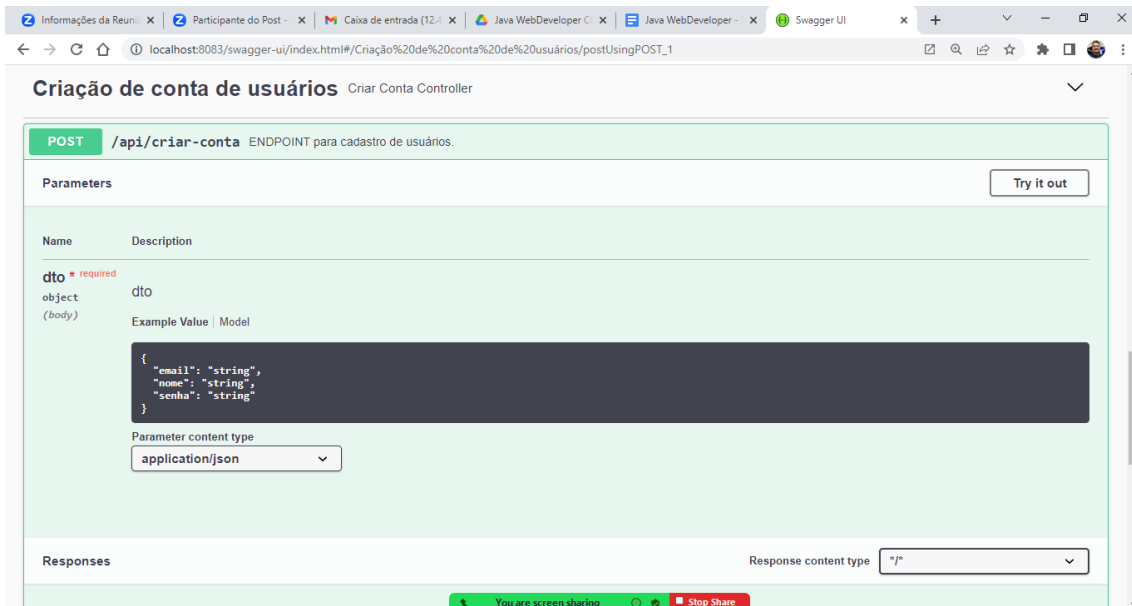


Swagger UI interface showing the parameters for the POST endpoint for user authentication. The endpoint is "/api/autenticar" (ENDPOINT para autenticação de usuários e obtenção de Token). The method is POST. The parameters are:

- dto** (required): object (body). Example Value:

```
{ "email": "string", "senha": "string" }
```

. Parameter content type: application/json.



Criando DTOS para modelar os dados que a API irá retornar após uma requisição:
RESPONSES.

```
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 GetUsuarioDTO {

    private Integer idUsuario;
    private String nome;
    private String email;
}

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 ResponseAutenticarDTO {

    private Integer status;
    private String mensagem;
    private GetUsuarioDTO usuario;
    private String token;
}

package br.com.cotiinformatica.application.dtos;

import java.util.Date;

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

@Setter
@Getter
@NoArgsConstructor
@AllArgsConstructor
@ToString
public class ResponseCriarContaDTO {

    private Integer status;
    private String mensagem;
    private GetUsuarioDTO usuario;
    private Date dataHoraCadastro;
}
```

```
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 ResponseRecuperarSenhaDTO {
```

```
    private Integer status;
```

```
    private String mensagem;
```

```
}
```

Voltando nos controladores:

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

```
import javax.validation.Valid;
```

```
import org.springframework.http.ResponseEntity;
```

```
import org.springframework.web.bind.annotation.PostMapping;
```

```
import org.springframework.web.bind.annotation.RequestBody;
```

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

```
import br.com.cotiinformatica.application.dtos.PostAutenticarDTO;
```

```
import br.com.cotiinformatica.application.dtos.ResponseAutenticarDTO;
```

```
import io.swagger.annotations.Api;
```

```
import io.swagger.annotations.ApiOperation;
```

```
@Api(tags = "Autenticação de usuários")
```

```
@RestController
```

```
public class AutenticarController {
```

```
    @ApiOperation("ENDPOINT para autenticação de usuários e obtenção de Token.")
```

```
    @PostMapping("/api/autenticar")
```

```
    public ResponseEntity<ResponseAutenticarDTO> post
```

```
        (@Valid @RequestBody PostAutenticarDTO dto) {
```

```
        return null;
```

```
    }
```

```
}
```

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

```
import javax.validation.Valid;
```

```
import org.springframework.http.ResponseEntity;
```

```
import org.springframework.web.bind.annotation.PostMapping;
```

```
import org.springframework.web.bind.annotation.RequestBody;
```

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

```
import br.com.cotiinformatica.application.dtos.PostCriarContaDTO;
import br.com.cotiinformatica.application.dtos.ResponseCriarContaDTO;
import io.swagger.annotations.Api;
import io.swagger.annotations.ApiOperation;

@Api(tags = "Criação de conta de usuários")
@RestController
public class CriarContaController {

    @ApiOperation("ENDPOINT para cadastro de usuários.")
    @PostMapping("/api/criar-conta")
    public ResponseEntity<ResponseCriarContaDTO> post
        (@Valid @RequestBody PostCriarContaDTO dto) {
        return null;
    }
}

package br.com.cotiinformatica.api.controllers;

import javax.validation.Valid;

import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RestController;

import br.com.cotiinformatica.application.dtos.PostRecuperarSenhaDTO;
import br.com.cotiinformatica.application.dtos.ResponseRecuperarSenhaDTO;
import io.swagger.annotations.Api;
import io.swagger.annotations.ApiOperation;

@Api(tags = "Recuperação de senha")
@RestController
public class RecuperarSenhaController {

    @ApiOperation("ENDPOINT para recuperação da senha de acesso do usuário.")
    @PostMapping("/api/recuperar-senha")
    public ResponseEntity<ResponseRecuperarSenhaDTO> post
        (@Valid @RequestBody PostRecuperarSenhaDTO dto) {
        return null;
    }
}
```

Desenvolvimento de testes no Spring Boot

É importante testarmos os ENDPOINTS das nossas APIs. E este pode ser feito através de programação Java. Esta programação de testes é algo que deve ser feito pelo próprio desenvolvedor.

Existe um processo utilizado em fabricas de software onde o programador deve ser responsável por testar as suas entregas, ou seja, desenvolver código capaz de testar de forma automatizadas as funcionalidade que ele está entregando.

Este processo é chamado de **TDD – Test Driven Development**

TDD – Test Driven Development

Desenvolvimento dirigido a testes.

O fluxo do processo TDD funciona da seguinte forma:

1. Escreva um teste que falhe

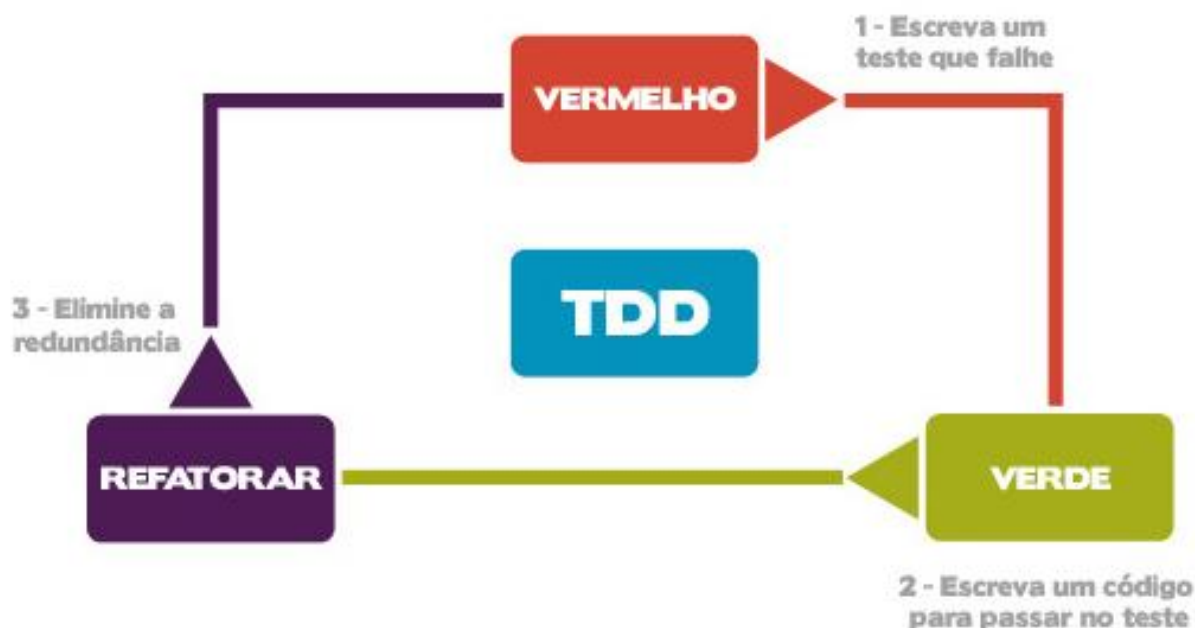
Aqui, o desenvolvedor irá criar um caso de teste para todas as funcionalidades que precisam ser validadas. No nosso caso, vamos criar casos de teste para os ENDPOINTS da API.
No momento a Api não está implementada, portanto, os testes irão retornar como resultado: **Falha**.








2. Escreva um código para passar no teste

Aqui, o desenvolvedor deverá implementar o programa necessário para satisfazer os critérios do teste. No nosso caso, iremos implementar a API de forma a garantir que os serviços possam passar nos testes previamente criados.

3. Elimine as redundâncias

Aqui, o desenvolvedor irá refatorar o seu código em busca de melhorias, eliminar as redundâncias de código, otimizar o programa etc.



- ✓  src/test/java
 - ✓  br.com.cotiinformatica
 - >  ApiUsuariosApplicationTests.java
 - >  JRE System Library [JavaSE-11]
 - >  Maven Dependencies
 - >  src
 -  target

Para cada **ENDPOINT** da nossa API, faremos pelo menos um método de teste:

POST /api/autenticar

POST /api/criar-conta

POST /api/recuperar-senha

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

```
import static org.assertj.core.api.Assertions.fail;
```

```
import org.junit.jupiter.api.Test;
```

```
import org.springframework.boot.test.context.SpringBootTest;
```

```
@SpringBootTest
```

```
class ApiUsuariosApplicationTests {
```

```
    @Test
```

```
    public void postAutenticarTest() {  
        fail("Não implementado.");  
    }
```

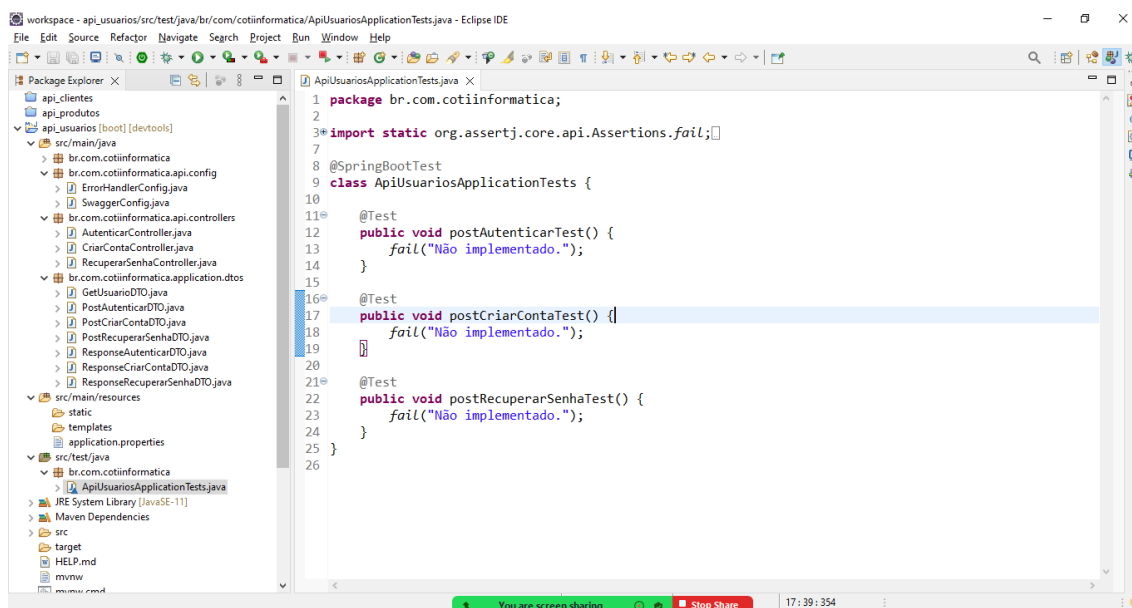
```
    @Test
```

```
    public void postCriarContaTest() {  
        fail("Não implementado.");  
    }
```

```
    @Test
```

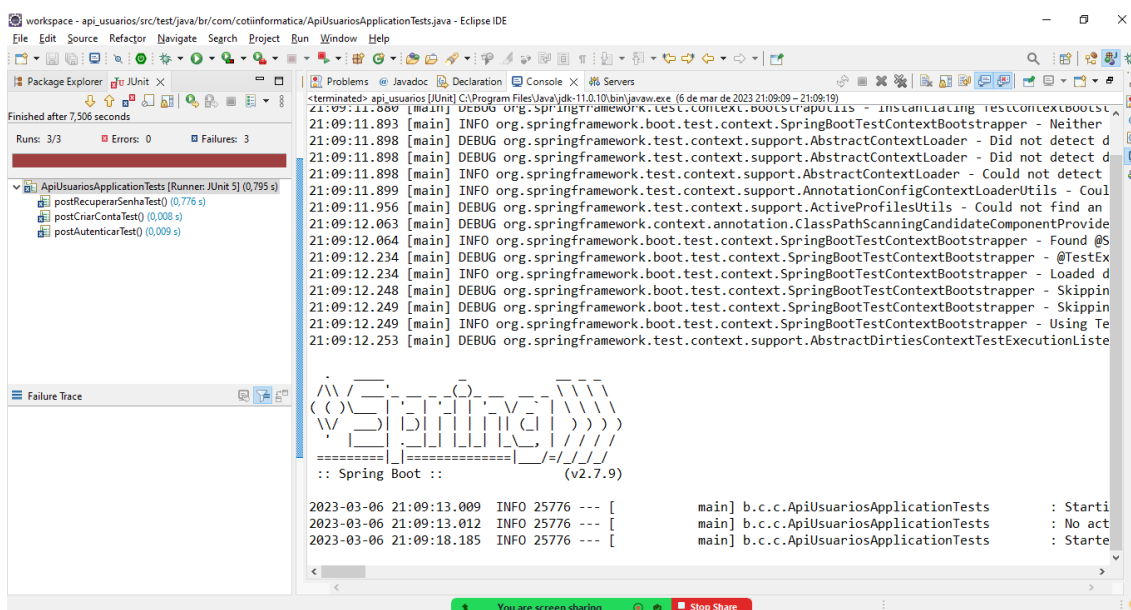
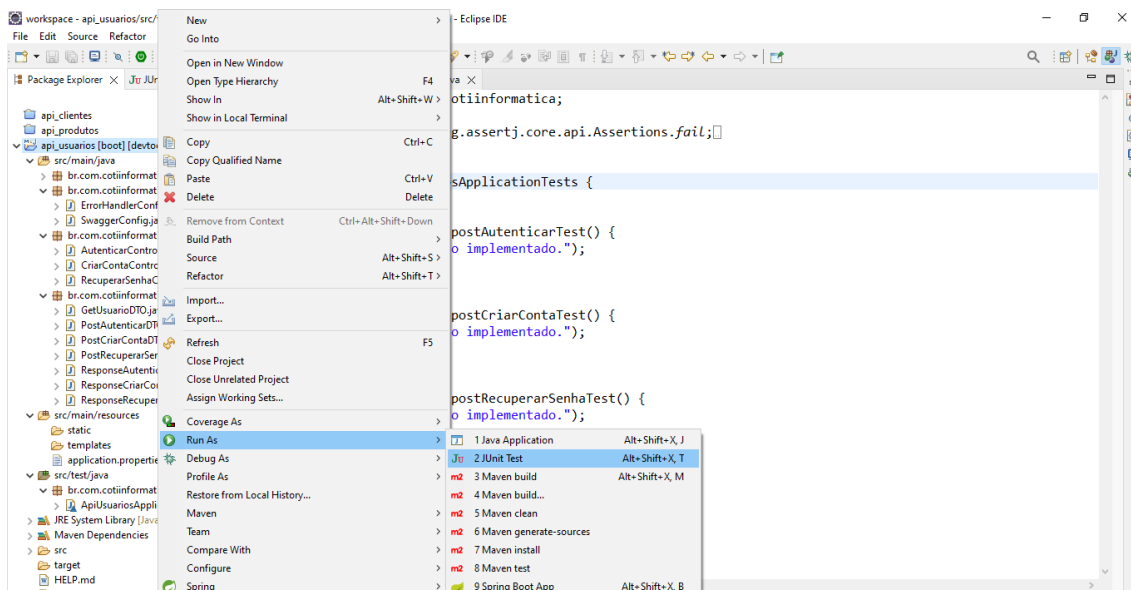
```
    public void postRecuperarSenhaTest() {  
        fail("Não implementado.");  
    }
```

```
}
```



** A biblioteca que o Java utiliza para executar os testes chama-se: **JUnit**

Executando os testes:



Para implementar os testes, precisamos gerar massa de dados. Ou seja, precisamos de alguma biblioteca capaz de gerar os dados utilizados no teste:

/pom.xml

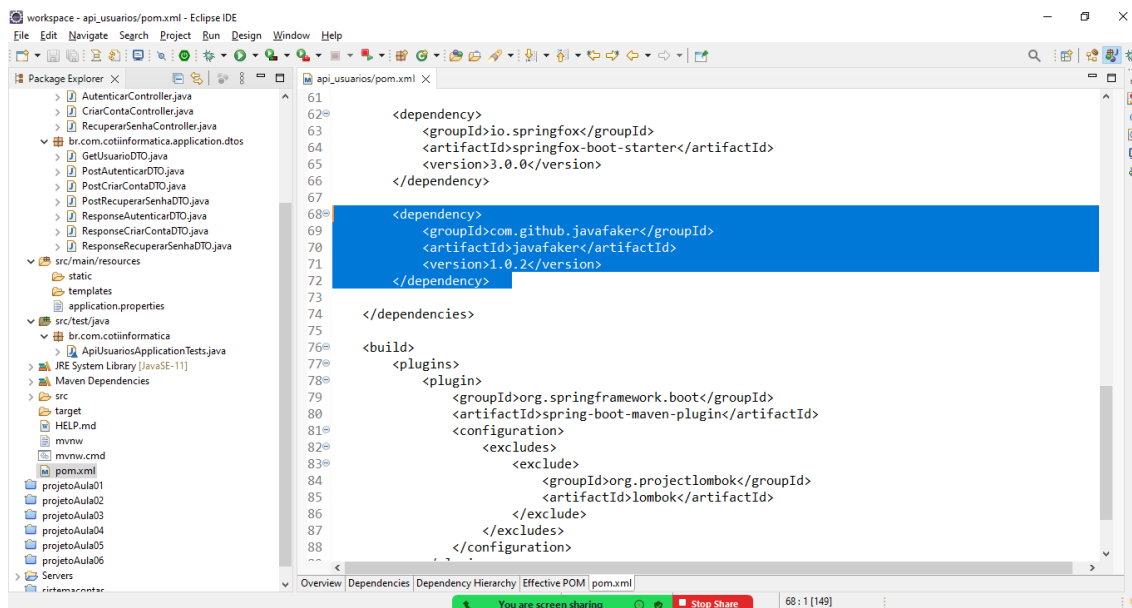
<dependency>

<groupId>com.github.javafaker</groupId>

<artifactId>javafaker</artifactId>

<version>1.0.2</version>

</dependency>



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

```
import static
```

```
org.springframework.test.web.servlet.request.MockMvcRequestBuilders.post;
```

```
import static
```

```
org.springframework.test.web.servlet.result.MockMvcResultMatchers.status;
```

```
import java.util.Locale;
```

```
import org.junit.jupiter.api.Test;
```

```
import org.springframework.beans.factory.annotation.Autowired;
```

```
import
```

```
org.springframework.boot.test.autoconfigure.web.servlet.AutoConfigureMockMvc;
```

```
import org.springframework.boot.test.context.SpringBootTest;
```

```
import org.springframework.test.web.servlet.MockMvc;
```

```
import com.fasterxml.jackson.databind.ObjectMapper;
```

```
import com.github.javafaker.Faker;
```

```
import br.com.cotiinformatica.application.dtos.PostAutenticarDTO;
```

```
import br.com.cotiinformatica.application.dtos.PostCriarContaDTO;
```

```
import br.com.cotiinformatica.application.dtos.PostRecuperarSenhaDTO;
```

```
@SpringBootTest
```

```
@AutoConfigureMockMvc
```

```
class ApiUsuariosApplicationTests {
```

```
    @Autowired
```

```
    private MockMvc mock;
```

```
@Autowired
private ObjectMapper objectMapper;

@Test
public void postAutenticarTest() throws Exception {

    PostAutenticarDTO dto = new PostAutenticarDTO();
    dto.setEmail("admin@cotiinformatica.com.br");
    dto.setSenha("@Admin123");

    mock.perform(
        post("/api/autenticar")
        .contentType("application/json")
        .content(
            objectMapper.writeValueAsString(dto))
        .andExpect(status().isOk());
    }

    @Test
    public void postCriarContaTest() throws Exception {

        PostCriarContaDTO dto = new PostCriarContaDTO();
        Faker faker = new Faker(new Locale("pt-BR"));

        dto.setNome(faker.name().fullName());
        dto.setEmail(faker.internet().emailAddress());
        dto.setSenha(faker.internet().password(8, 20) + "@");

        mock.perform(
            post("/api/criar-conta")
            .contentType("application/json")
            .content(
                objectMapper.writeValueAsString(dto))
            .andExpect(status().isCreated());
        }

        @Test
        public void postRecuperarSenhaTest() throws Exception {

            PostRecuperarSenhaDTO dto = new PostRecuperarSenhaDTO();
            dto.setEmail("teste@cotiinformatica.com.br");

            mock.perform(
                post("/api/recuperar-senha")
                .contentType("application/json")
                .content(
                    objectMapper.writeValueAsString(dto))
                .andExpect(status().isOk());
            }
        }
```

Nos controladores, estamos definindo por enquanto o retorno dos ENDPOINTS como status **501 – NOT IMPLEMENTED**

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

import javax.validation.Valid;

import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RestController;

import br.com.cotiinformatica.application.dtos.PostAutenticarDTO;
import br.com.cotiinformatica.application.dtos.ResponseAutenticarDTO;
import io.swagger.annotations.Api;
import io.swagger.annotations.ApiOperation;

@Api(tags = "Autenticação de usuários")
@RestController
public class AutenticarController {

    @ApiOperation("ENDPOINT para autenticação  
de usuários e obtenção de Token.")
    @PostMapping("/api/autenticar")
    public ResponseEntity<ResponseAutenticarDTO> post
        (@Valid @RequestBody PostAutenticarDTO dto) {
        return ResponseEntity.status
            (HttpStatus.NOT_IMPLEMENTED).body(null);
    }
}

package br.com.cotiinformatica.api.controllers;

import javax.validation.Valid;

import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RestController;

import br.com.cotiinformatica.application.dtos.PostCriarContaDTO;
import br.com.cotiinformatica.application.dtos.ResponseCriarContaDTO;
import io.swagger.annotations.Api;
import io.swagger.annotations.ApiOperation;
```

```
@Api(tags = "Criação de conta de usuários")
@RestController
public class CriarContaController {

    @ApiOperation("ENDPOINT para cadastro de usuários.")
    @PostMapping("/api/criar-conta")
    public ResponseEntity<ResponseCriarContaDTO> post
        (@Valid @RequestBody PostCriarContaDTO dto) {

        return ResponseEntity.status
            (HttpStatus.NOT_IMPLEMENTED).body(null);
    }
}

package br.com.cotiinformatica.api.controllers;

import javax.validation.Valid;

import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RestController;

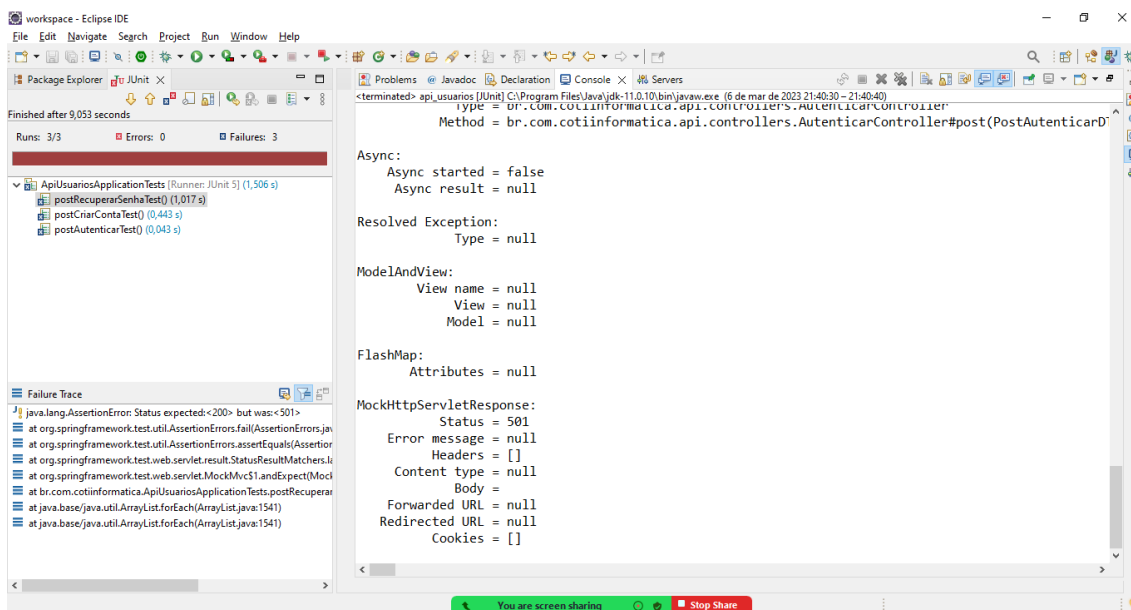
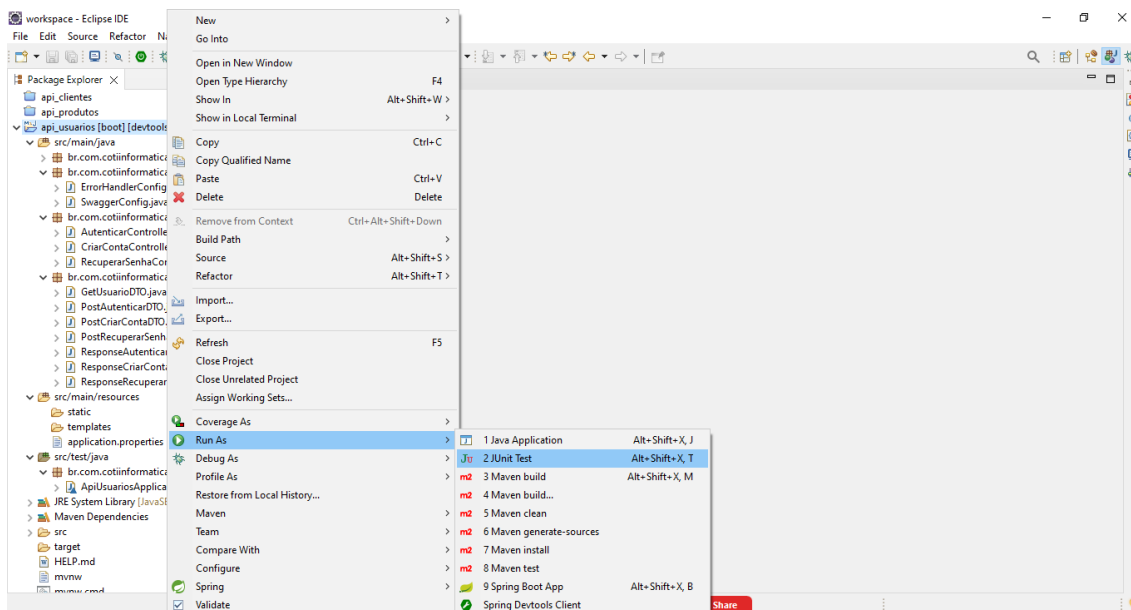
import br.com.cotiinformatica.application.dtos.PostRecuperarSenhaDTO;
import br.com.cotiinformatica.application.dtos.ResponseRecuperarSenhaDTO;
import io.swagger.annotations.Api;
import io.swagger.annotations.ApiOperation;

@Api(tags = "Recuperação de senha")
@RestController
public class RecuperarSenhaController {

    @ApiOperation("ENDPOINT para recuperação da
        senha de acesso do usuário.")
    @PostMapping("/api/recuperar-senha")
    public ResponseEntity<ResponseRecuperarSenhaDTO> post
        (@Valid @RequestBody PostRecuperarSenhaDTO dto) {

        return ResponseEntity.status
            (HttpStatus.NOT_IMPLEMENTED).body(null);
    }
}
```

Executando os testes:



Continua...