

Desenvolvimento de APIs REST

07 - Documentação da API com Swagger

- Configuração Swagger
- Documentação API
- Documentação Endpoints e Models
- Importação para Postman

Habilitar a documentação com Swagger



É uma aplicação open source que auxilia desenvolvedores nos processos de definir, criar, documentar e consumir APIs REST. O Swagger padroniza este tipo de integração, descrevendo os recursos que uma API deve possuir, como endpoints, dados recebidos, dados retornados, códigos HTTP e métodos de autenticação, entre outras opções disponíveis. O Swagger consegue ler a estrutura da sua API e gerar automaticamente uma documentação, essa documentação retorna todas as operações que sua API suporta, quais são os parâmetros de sua API, e se precisa de autorização.

OpenApi

O Swagger trabalha em conjunto com o OpenApi que é um padrão de especificação para descrever as APIs REST, que permite que humanos e computadores descubram e entendam os recursos de um serviço sem exigir acesso ao código fonte da aplicação. A especificação do Open API é aberta e está disponível no GitHub no link abaixo:

<https://github.com/OAI/OpenAPI-Specification>

<https://www.openapis.org>

Habilitar a documentação com Swagger



Utilitários

O Swagger possui algumas ferramentas que auxiliam o desenvolvedor de APIs REST, para geração da documentação, bibliotecas e módulos.

Swagger Inspector - Similar ao Postman e Insomnia. Podemos utilizar para testar nossas API's.

<https://inspector.swagger.io/>

Swagger UI - Serve para gerar nossas documentações.

Swagger Node - Módulo Swagger para node.

Swagger Editor - Editor para criação de definições baseadas em YAML ou JSON.

Outras ferramentas podem ser verificadas em:

<https://swagger.io/tools/>

Exemplos de API's:

<https://developer.ifood.com.br/reference>

<https://developer.itau.com.br/>

<https://petstore.swagger.io/>


Configurar Swagger Spring Boot



Para configurar o Swagger para nossa aplicação temos que baixar as dependências do maven no site

www.mvnrepository.com

Inserir as dependências abaixo no arquivo pom.xml

**SpringDoc OpenAPI Starter WebMVC UI » 2.3.0**
SpringDoc OpenAPI Starter WebMVC UI

License	Apache 2.0
Tags	spring openapi doc web ui api mvc starter
Date	Dec 03, 2023
Files	pom (1 KB) jar (22 KB) View All
Repositories	Central
Ranking	#3655 in MvnRepository (See Top Artifacts)
Used By	117 artifacts

Maven Gradle Gradle (Short) Gradle (Kotlin) SBT Ivy Grape Leiningen Buildr

```
<!-- https://mvnrepository.com/artifact/org.springdoc/springdoc-openapi-starter-webmvc-ui -->
<dependency>
  <groupId>org.springdoc</groupId>
  <artifactId>springdoc-openapi-starter-webmvc-ui</artifactId>
  <version>2.3.0</version>
</dependency>
```

☒ Include comment with link to declaration

Foi feito o download da última versão disponível. Esta versão é compatível com o **Spring Boot 3**

Habilitar a documentação com Swagger



Execute o Spring Boot e acesse a url
<http://localhost:8080/swagger-ui/index.html>

The screenshot displays the Swagger UI interface. At the top, the Swagger logo is on the left, and a search bar contains the text '/v3/api-docs' with an 'Explore' button on the right. Below this, the text 'OpenAPI definition' is shown with a 'v0' tag and an 'OAS 3.0' badge. Underneath, the 'Servers' section shows a dropdown menu with the selected value 'http://localhost:8080 - Generated server url'. The main content area is titled 'cliente-controller' and lists several API endpoints, each with a colored button indicating the HTTP method and a dropdown arrow on the right:

- GET /clientes/{id} (light blue)
- PUT /clientes/{id} (orange)
- DELETE /clientes/{id} (red)
- GET /clientes (light blue)
- POST /clientes (green)
- POST /clientes/all (green)

Below the endpoints, the 'Schemas' section is visible, showing a single schema named 'Cliente' with a right-pointing arrow.

Habilitar a documentação com Swagger



Acessando a url <http://localhost:8080/v3/api-docs> temos a documentação em JSON

```
localhost:8080/v3/api-docs

{
  "openapi": "3.0.1",
  "info": {
    "title": "OpenAPI definition",
    "version": "v0"
  },
  "servers": [
    {
      "url": "http://localhost:8080",
      "description": "Generated server url"
    }
  ],
  "paths": {
    "/clientes/{id}": {
      "get": {
        "tags": [
          "cliente-controller"
        ],
        "operationId": "buscarCliente",
        "parameters": [
          {
            "name": "id",
            "in": "path",
            "required": true,
            "schema": {
              "type": "integer",
              "format": "int64"
            }
          }
        ],
        "responses": {
          "200": {
            "description": "OK",
            "content": {
              "*/*": {
                "schema": {
                  "$ref": "#/components/schemas/Cliente"
                }
              }
            }
          }
        }
      }
    }
  }
}
```

Habilitar a documentação com Swagger



```
@Configuration
public class OpenAPIConfig {

    @Value("${dominio.openapi.dev-url}")
    private String devUrl;
    @Value("${dominio.openapi.prod-url}")
    private String prodUrl;

    @Bean
    public OpenAPI myOpenAPI() {
        Server devServer = new Server();
        devServer.setUrl(devUrl);
        devServer.setDescription("URL do servidor de desenvolvimento");
        Server prodServer = new Server();
        prodServer.setUrl(prodUrl);
        prodServer.setDescription("URL do servidor de produção");

        Contact contact = new Contact();
        contact.setEmail("contato@meudominio.com.br");
        contact.setName("Fulano");
        contact.setUrl("https://www.meudominio.com.br");

        License apacheLicense = new License().name("Apache
License").url("https://www.apache.org/licenses/LICENSE-2.0");

        Info info = new Info().title("API de Teste").version("1.0").contact(contact)
        .description("API para testes diversos").termsOfService("https://www.meudominio.com.br/termos")
        .license(apacheLicense);

        return new OpenAPI().info(info).servers(List.of(devServer, prodServer));
    }
}
```

Podemos melhorar as informações sobre a api inserindo o a classe para configuração **OpenAPIConfig** conforme abaixo.

A importações devem ser da classe **io.swagger.v3.oas.models** e **java.util.List**

Habilitar a documentação com Swagger



Adicionar no arquivo **application.properties** as variáveis definidas na classe **OpenAPIConfig**

```
dominio.openapi.dev-url=http://localhost:8080
dominio.openapi.prod-url=https://www.meudominio.com.br
```

API de Teste ^{1.0} ^{OAS 3.0}

[/v3/api-docs](#)

API para testes diversos

[Terms of service](#)

[Fulano - Website](#)

[Send email to Fulano](#)

[Apache License](#)

Servers

[http://localhost:8080 - URL do servidor de desenvolvimento](#) ▼

[http://localhost:8080 - URL do servidor de desenvolvimento](#)

[https://www.meudominio.com.br - URL do servidor de produção](#)

Habilitar a documentação com Swagger



Para deixar a documentação mais completa podem ser adicionada algumas anotações nos métodos do **controller** e campos do **entity** ou DTO.

```
@Tag(name="Cliente",description="Cadastro de Cliente")
@RestController @RequestMapping("/clientes") public class ClienteController
{

    @Autowired
    private ClienteRepository clienteRepository;

    @GetMapping

    @Operation(summary = "Lista todos os clientes", description = "A resposta lista os dados dos clientes id, nome, cpf e email.")
    @ApiResponses(value = {
        @ApiResponse(responseCode = "200",
            content = {@Content(schema = @Schema(implementation = Cliente.class), mediaType = "application/json")},
            description = "Retorna todos os clientes"),
        @ApiResponse(responseCode = "401", description = "Erro de autenticação"),
        @ApiResponse(responseCode = "403", description = "Não há permissão para acessar o recurso"),
        @ApiResponse(responseCode = "404", description = "Recurso não encontrado"),
        @ApiResponse(responseCode = "505", description = "Exceção interna da aplicação") })

    public List<Cliente> listar(){
        return clienteRepository.findAll();
    }
}
```

Para cada método vamos definir o swagger para fazer a documentação

@Tag – Adicionar título e descrição ao recurso.

@Operation - Serve para explicar a função do recurso.

@ApiResponses e @ApiResponse – Serve especificar os códigos e as mensagens de retorno diretamente no **controller** com as anotações.

Habilitar a documentação com Swagger



Visualizando no Swagger as alterações

GET /clientes Lista todos os clientes

A resposta lista os dados dos clientes id, nome, cpf, email e data de nascimento

Parameters [Try it out](#)

No parameters

Responses

Code	Description	Links
200	Retorna todos os clientes Media type: <input type="text" value="application/json"/> Controls: Accept header Example Value Schema <pre>{ "id": 0, "nome": "string", "cpf": "string", "email": "string", "dataNascimento": "2024-01-03" }</pre>	No links
401	Erro de autenticação Media type: <input type="text" value="*/"/> Example Value Schema <input type="button" value="▼ [Cliente > {...}]"/>	No links
403	Não há permissão para acessar o recurso Media type: <input type="text" value="*/"/> Example Value Schema <input type="button" value="▼ [Cliente > {...}]"/>	No links
404	Recurso não encontrado Media type: <input type="text" value="*/"/> Example Value Schema <input type="button" value="▼ [Cliente > {...}]"/>	No links
505	Exceção interna da aplicação Media type: <input type="text" value="*/"/> Example Value Schema <input type="button" value="▼ [Cliente > {...}]"/>	No links

Habilitar a documentação com Swagger



Adicionar a documentação nos outros recursos

```
@GetMapping("/{id}")
    @Operation(summary = "Retorna um cliente", description = "A resposta é um objeto com os dados do cliente id, nome, cpf e email.")
    @ApiResponses(value = {
        @ApiResponse(responseCode = "200", description = "Retorna um cliente cliente"),
        @ApiResponse(responseCode = "401", description = "Erro de autenticação"),
        @ApiResponse(responseCode = "403", description = "Não há permissão para acessar o recurso"),
        @ApiResponse(responseCode = "404", description = "Recurso não encontrado"),
        @ApiResponse(responseCode = "505", description = "Exceção interna da aplicação") })

    public ResponseEntity<Cliente> buscar(@PathVariable Long id) { Optional<Cliente> cliente = clienteRepository.findById(id);
    if (cliente.isPresent()) {
        return ResponseEntity.ok(cliente.get());
    }
    return ResponseEntity.notFound().build();
}

@PostMapping @ResponseStatus(HttpStatus.CREATED)
    @Operation(summary="Insere um cliente", description = "A resposta é um objeto com os dados cadastrado do cliente.")
    @ApiResponses(value= {
        @ApiResponse(responseCode=201, description ="Cliente adicionado"),
        @ApiResponse(responseCode=401, description ="Erro de autenticação"),
        @ApiResponse(responseCode=403, description ="Não há permissão para acessar o recurso"),
        @ApiResponse(responseCode =404, description ="Recurso não encontrado"), @ApiResponse(code=505, message="Exceção interna da aplicação")
    })

    public Cliente inserir(@Valid @RequestBody Cliente cliente) {
        return clienteRepository.save(cliente);
    }
```

Habilitar a documentação com Swagger



Adicionar a documentação nos outros recursos

```
@PutMapping("/{id}")
@Operation(summary="Atualizar Cliente", description="Atualiza dados de um cliente")
@ApiResponses(value= {
    @ApiResponse(responseCode=200, description="Cliente Atualizado"),
    @ApiResponse(responseCode=401, description="Erro de autenticação"),
    @ApiResponse(responseCode=403, description="Não há permissão para acessar o recurso"),
    @ApiResponse(responseCode=404, description="Recurso não encontrado"),
    @ApiResponse(responseCode=505, description="Exceção interna da aplicação")
})

public ResponseEntity<Cliente> atualizar(@PathVariable Long id, @Valid @RequestBody Cliente cliente) { if
    (!clienteRepository.existsById(id)) {
        return ResponseEntity.notFound().build();
    }
    cliente.setId(id); cliente=clienteRepository.save(cliente); return
    ResponseEntity.ok(cliente);
}

>DeleteMapping("/{id}")
@Operation(summary="Remover Cliente", description="Remove um cliente")
@ApiResponses(value= {
    @ApiResponse(responseCode=200, description="Cliente Removido"),
    @ApiResponse(responseCode=401, description="Erro de autenticação"),
    @ApiResponse(responseCode=403, description="Não há permissão para acessar o recurso"),
    @ApiResponse(responseCode=404, description="Recurso não encontrado"),
    @ApiResponse(responseCode=505, description="Exceção interna da aplicação")
})

public ResponseEntity<Void> remover(@PathVariable Long id) {
    if(!clienteRepository.existsById(id)){
        return ResponseEntity.notFound().build();
    }
    clienteRepository.deleteById(id);
    return ResponseEntity.noContent().build();
}
```

Habilitar a documentação com Swagger



Podemos adicionar documentação no nosso **model** usando a anotação **@Schema** importar de **io.swagger.v3.oas.annotations.media**

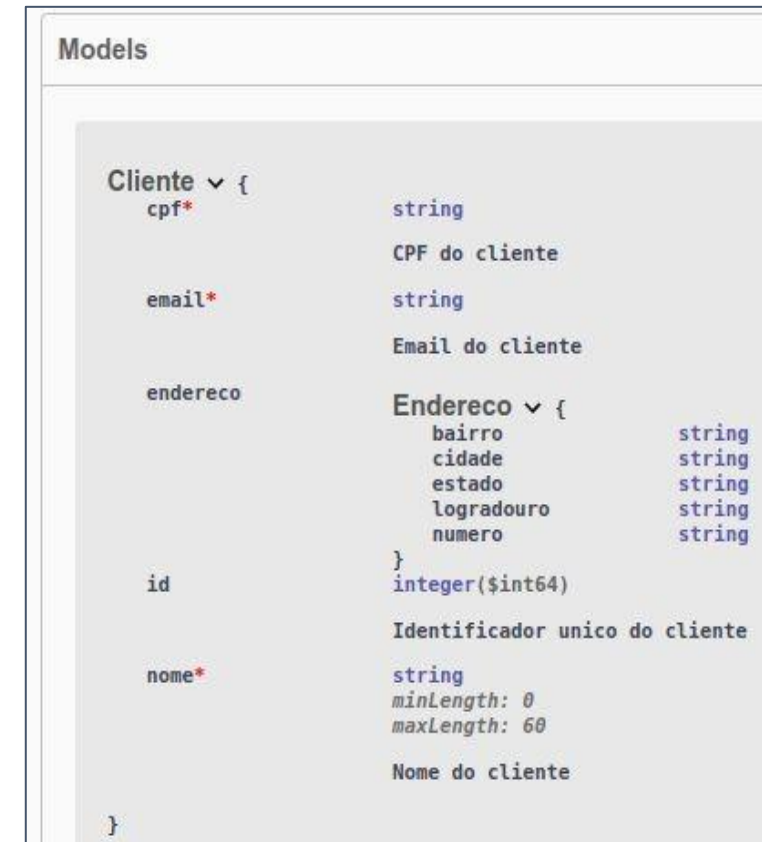
```
@Id
@GeneratedValue(strategy = GenerationType.IDENTITY)
@Column(name="id_cliente")
@Schema(description="Identificador unico do cliente")
private Long id;

@NotBlank(message="Preencha o nome")
@Size(max=60)
@Column
@Schema(description="Nome do cliente", required = true)
private String nome;

@CPF(message="CPF Inválido")
@Column
@Schema(description="CPF do cliente", required = true)
private String cpf;

@email(message="Email inválido")
@Column
@Schema(description="Email do cliente", required = true)
private String email;

@Embedded
@Schema(description="Endereco do cliente")
private Endereco endereco;
```



Habilitar a documentação com Swagger



Importando para o Postman

Clique em File – Import e copie a url destacada

API de Teste 1.0.0

[Base URL: localhost:8080/]
<http://localhost:8080/v2/api-docs>

Essa é uma API desenvolvida para tests

[Serratec - Website](#)
[Send email to Serratec](#)
[Apache License Version 2.0](#)

Import

File Folder **Link** Raw text Code repository **New** API Gateway **New**

Enter a URL

<http://localhost:8080/v2/api-docs>

Continue

Import

Select files to import · 1/1 selected

NAME	FORMAT	IMPORT AS
API de Teste	Swagger 2.0	API

☒ Generate collection from imported APIs

Link this collection as

Documentation

> Show advanced settings

Cancel

Import

API de Teste

draft

API de Teste

clientes

{Id}

> GET Lista todos os cliente

> POST Inserir os dados de um ...

pedidos

> GET listar

> POST inserir

> GET buscar

produtos

{Id}

> GET listar

> POST inserir

Habilitar a documentação com Swagger



Exercícios

1. Fazer a documentação do outros Controllers
2. Fazer a documentação das outras classes de modelo.