

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

Maven Gradle SBT Ivy Grape Leiningen Buildr

```
<!-- https://mvnrepository.com/artifact/io.springfox/springfox-boot-starter -->
<dependency>
  <groupId>io.springfox</groupId>
  <artifactId>springfox-boot-starter</artifactId>
  <version>3.0.0</version>
</dependency>
```

Maven Gradle SBT Ivy Grape Leiningen Buildr

```
<!-- https://mvnrepository.com/artifact/io.springfox/springfox-swagger-ui -->
<dependency>
  <groupId>io.springfox</groupId>
  <artifactId>springfox-swagger-ui</artifactId>
  <version>3.0.0</version>
</dependency>
```

Habilitar a documentação com Swagger



Vamos criar a classe para configuração com o nome de **SwaggerConfig** no pacote **config** em algum projeto do curso

```
@Configuration
public class SwaggerConfig {
    @Bean
    public Docket api() {
        return new Docket(DocumentationType.SWAGGER_2)
            .select()
            .apis(RequestHandlerSelectors.any())
            .paths(PathSelectors.any())
            .build();
    }
}
```

O método **api** tem como retorno um **Docket** que realiza tudo que é necessário para criação da documentação permitindo configurar aspectos dos endpoints expostos por ele.

Nos métodos **apis** e **paths** definimos que todas as apis e caminhos estarão disponíveis.

A partir da versão 2.6 do Spring Boot, é necessário incluir a linha abaixo no arquivo `application.properties`. Esse problema é causado porque o Spring Fox 3.0.0 não suporta a nova estratégia [PathPattern para Spring MVC](#), que é o novo padrão a partir do spring-boot 2.6.0. Caso não seja colocada, aparecerá um erro de `NullPointerException` no console.

```
spring.mvc.pathmatch.matching-strategy=ant_path_matcher
```

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 for an API. At the top, the Swagger logo is visible, along with the text "Supported by SMARTBEAR". A dropdown menu labeled "Select a definition" is set to "default". The main heading is "Api Documentation" with a version indicator "1.0". Below this, the base URL is specified as "[Base URL: localhost:8080/]" and the full URL as "http://localhost:8080/v2/api-docs". Links for "Api Documentation", "Terms of service", and "Apache 2.0" are provided. The API is organized into controllers: "basic-error-controller" (Basic Error Controller), "cliente-controller" (Cliente Controller), "pedido-controller" (Pedido Controller), and "produto-controller" (Produto Controller). The "cliente-controller" is expanded, showing five endpoints: a GET endpoint for "/clientes" (listar), a POST endpoint for "/clientes" (inserir), a GET endpoint for "/clientes/{id}" (buscar), a PUT endpoint for "/clientes/{id}" (atualizar), and a DELETE endpoint for "/clientes/{id}" (remover). Each endpoint is represented by a colored bar with the HTTP method, the path, and the action name.

Habilitar a documentação com Swagger

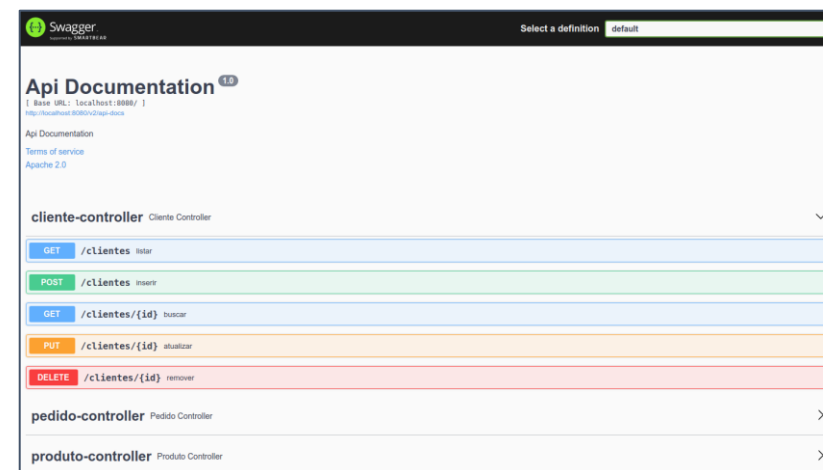
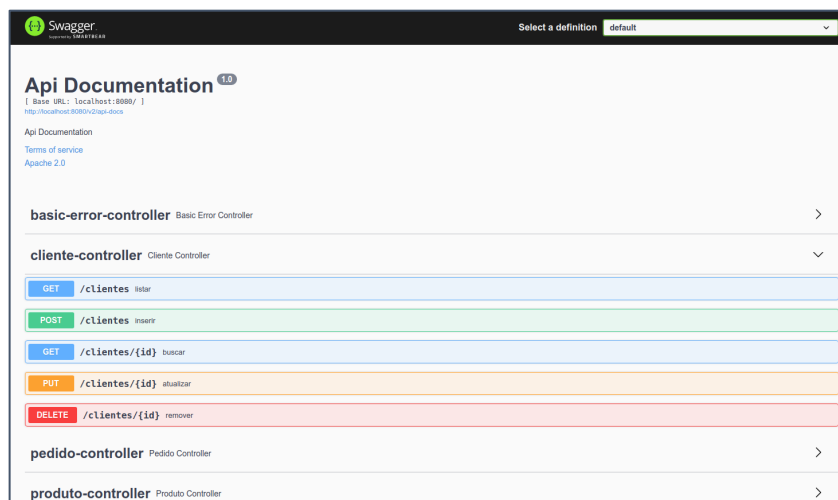


Podemos melhorar nossa documentação, exibindo somente os endpoints que precisamos apresentar personalizando assim as informações gerais da API.

```
@Configuration
public class SwaggerConfig {
    @Bean
    public Docket api() {
        return new Docket(DocumentationType.SWAGGER_2)
            .select()
            .apis(RequestHandlerSelectors.basePackage("org.serratec.backend"))
            .paths(PathSelectors.any())
            .build();
    }
}
```

Foi modificado o argumento do método **apis** onde foi inserido o pacote da nossa aplicação. É possível incluir apenas o pacote **controller** onde estão os controllers.

A documentação será exibida conforme a imagem abaixo (antes e depois)



Habilitar a documentação com Swagger



Podemos melhorar as informações sobre a api inserindo o método **apiInfo** destacado abaixo.

```
@Configuration
public class SwaggerConfig {
    @Bean
    public Docket api() {
        return new Docket(DocumentationType.SWAGGER_2)
            .select()
            .apis(RequestHandlerSelectors.basePackage("org.serratec.backend"))
            .paths(PathSelectors.any())
            .build()
            .apiInfo(apiInfo());
    }

    private ApiInfo apiInfo() {
        ApiInfo apiInfo = new ApiInfoBuilder()
            .title("API de Teste")
            .description("Essa é uma API desenvolvida para tests")
            .license("Apache License Version 2.0")
            .licenseUrl("https://www.apache.org/licenses/LICENSE-2.0")
            .version("1.0.0")
            .contact(new Contact("Serratec", "www.serrtatec.org.br", "teste@gmail.com"))
            .build();
        return apiInfo;
    }
}
```



A importação da classe **Contact** deve ser de `springfox.documentation.service.Contact`;

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.

```
@RestController
@RequestMapping("/clientes")
public class ClienteController {

    @Autowired
    private ClienteRepository clienteRepository;

    @GetMapping
    @ApiOperation(value="Lista todos os cliente", notes="Listagem de Clientes")
    @ApiResponses(value= {
        @ApiResponse(code=200, message="Retorna todos os clientes"),
        @ApiResponse(code=401, message="Erro de autenticação"),
        @ApiResponse(code=403, message="Não há permissão para acessar o recurso"),
        @ApiResponse(code=404, message="Recurso não encontrado"),
        @ApiResponse(code=505, message="Exceção interna da aplicação"),
    })
    public List<Cliente> listar() {
        return clienteRepository.findAll();
    }
}
```

Para cada método vamos definir o swagger para fazer a documentação

@ApiOperation - 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 cliente

Listagem de Clientes

Parameters

Try it out

No parameters

Responses

Response content type

/

Code	Description
200	Retorna todos os clientes <div>Example Value Model</div> <pre>[{ "cpf": "string", "email": "string", "endereco": { "bairro": "string", "cidade": "string", "estado": "string", "logradouro": "string", "numero": "string" }, "id": 0, "nome": "string" }]</pre>
401	Erro de autenticação
403	Não há permissão para acessar o recurso
404	Recurso não encontrado
505	Exceção interna da aplicação

Habilitar a documentação com Swagger



Adicionar a documentação nos outros recursos

```
@GetMapping("/{id}")
@ApiOperation(value="Retorna um cliente", notes="Cliente")
@ApiResponses(value= {
    @ApiResponse(code=200, message="Retorna um cliente"),
    @ApiResponse(code=401, message="Erro de autenticação"),
    @ApiResponse(code=403, message="Não há permissão para acessar o recurso"),
    @ApiResponse(code=404, message="Recurso não encontrado"),
    @ApiResponse(code=505, message="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)
@ApiOperation(value="Insere os dados de um Cliente", notes="Inserir Cliente")
@ApiResponses(value= {
    @ApiResponse(code=201, message="Cliente adicionado"),
    @ApiResponse(code=401, message="Erro de autenticação"),
    @ApiResponse(code=403, message="Não há permissão para acessar o recurso"),
    @ApiResponse(code=404, message="Recurso não encontrado"),
    @ApiResponse(code=505, message="Exceção interna da aplicação"),
})
public Cliente inserir(@Valid @RequestBody Cliente cliente) {
    return clienteRepository.save(cliente);
}
```

```
@PutMapping("/{id}")
@ApiOperation(value="Atualiza dados de um cliente", notes="Atualizar Cliente")
@ApiResponses(value= {
    @ApiResponse(code=200, message="Cliente Atualizado"),
    @ApiResponse(code=401, message="Erro de autenticação"),
    @ApiResponse(code=403, message="Não há permissão para acessar o recurso"),
    @ApiResponse(code=404, message="Recurso não encontrado"),
    @ApiResponse(code=505, message="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}")
@ApiOperation(value="Remove um cliente", notes="Remover Cliente")
@ApiResponses(value= {
    @ApiResponse(code=200, message="Cliente Removido"),
    @ApiResponse(code=401, message="Erro de autenticação"),
    @ApiResponse(code=403, message="Não há permissão para acessar o recurso"),
    @ApiResponse(code=404, message="Recurso não encontrado"),
    @ApiResponse(code=505, message="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**

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

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

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

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

@Embedded
@ApiModelProperty(value="Endereco do cliente")
private Endereco endereco;
```

Models

```
Cliente ▾ {
  cpf*      string
            CPF do cliente

  email*    string
            Email do cliente

  endereco  Endereco ▾ {
    bairro  string
    cidade  string
    estado  string
    logradouro string
    numero  string
  }

  id        integer($int64)
            Identificador unico do cliente

  nome*     string
            minLength: 0
            maxLength: 60
            Nome do cliente
}
```

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.