

Documentação da API com o Swagger 3.0

Introdução

O Swagger trata-se de uma aplicação open source que auxilia as pessoas desenvolvedoras nos processos de definir, criar, documentar e consumir APIs REST. O Swagger tem por objetivo padronizar este tipo de integração, descrevendo os recursos que uma API deve possuir, como endpoints, dados recebidos, dados retornados, códigos HTTP, métodos de autenticação, entre outros.

APIs REST são frequentemente usadas para a integração de aplicações, seja para consumir serviços de terceiros, seja para prover novos. Para estas APIs, o Swagger facilita a modelagem, a documentação e a geração de código.

Configuração

Dependências

Configurando o pom.xml

Insira a seguinte dependência no projeto:

```
<dependency>
  <groupId>io.springfox</groupId>
  <artifactId>springfox-boot-starter</artifactId>
  <version>3.0.0</version>
</dependency>
```

SwaggerConfig

Crie uma nova package no seu projeto chamada **configuration**, dentro dela crie uma classe chamada SwaggerConfig e configure segundo o modelo abaixo:

```
package br.org.generation.blogpessoal.configuration;

import java.util.ArrayList;
import java.util.List;

import org.springframework.context.annotation.Bean;
```

```

import org.springframework.context.annotation.Configuration;
import org.springframework.http.HttpMethod;

import springfox.documentation.builders.ApiInfoBuilder;
import springfox.documentation.builders.PathSelectors;
import springfox.documentation.builders.RequestHandlerSelectors;
import springfox.documentation.builders.ResponseBuilder;
import springfox.documentation.service.ApiInfo;
import springfox.documentation.service.Contact;
import springfox.documentation.service.Response;
import springfox.documentation.spi.DocumentationType;
import springfox.documentation.spring.web.plugins.Docket;

@Configuration
public class SwaggerConfig {
    @Bean
    public Docket api() {
        return new Docket(DocumentationType.SWAGGER_2)
            .select()
            .apis(RequestHandlerSelectors
                .basePackage("br.org.generation.blogpessoal.controller"))
            .paths(PathSelectors.any())
            .build()
            .apiInfo(metadata())
            .useDefaultResponseMessages(false)
            .globalResponses(HttpMethod.GET, responseMessage())
            .globalResponses(HttpMethod.POST, responseMessage())
            .globalResponses(HttpMethod.PUT, responseMessage())
            .globalResponses(HttpMethod.DELETE, responseMessage());
    }

    public static ApiInfo metadata() {

        return new ApiInfoBuilder()
            .title("API - Blog Pessoal")
            .description("Projeto API Spring - Blog Pessoal")
            .version("1.0.0")
            .license("Apache License Version 2.0")
            .licenseUrl("https://github.com/rafaelq80")
            .contact(contact())
            .build();
    }

    private static Contact contact() {

        return new Contact("Rafael Queiróz",
            "https://github.com/rafaelq80",
            "rafaelproinfo@gmail.com");
    }
}

```

```

private static List<Response> responseMessage() {

    return new ArrayList<Response>() {

        private static final long serialVersionUID = 1L;

        {
            add(new ResponseBuilder().code("200")
                .description("Sucesso!").build());
            add(new ResponseBuilder().code("201")
                .description("Criado!").build());
            add(new ResponseBuilder().code("400")
                .description("Erro na requisição!").build());
            add(new ResponseBuilder().code("401")
                .description("Não Autorizado!").build());
            add(new ResponseBuilder().code("403")
                .description("Proibido!").build());
            add(new ResponseBuilder().code("404")
                .description("Não Encontrado!").build());
            add(new ResponseBuilder().code("500")
                .description("Erro!").build());
        }
    };
}

```

Principais Configurações

Método public Docket api()

Define a package onde estão as classes do tipo @RestController, para que o Swagger mapeie todas as classes e seus respectivos endpoints para montar a documentação do projeto.

```

.apis(RequestHandlerSelectors.basePackage("br.org.generation.blogpessoal.controller")
)

```

Método public static ApiInfo metadata()

1) Define o título da sua aplicação que será exibida na documentação.

```

.title("Blog Pessoal")

```

2) Cria uma descrição para a sua aplicação.

```
.description("API do Projeto de blog pessoal")
```

3) Define a versão da sua aplicação.

```
.version("1.0.0")
```

4) Define o tipo de licença da sua aplicação.

```
.license("Apache License Version 2.0")
```

5) Informa o link de acesso da licença da sua aplicação (geralmente se aplica a licença no Github).

```
.licenseUrl(https://github.com/rafaelq80)
```

**Em nosso exemplo inserimos o endereço do Github somente para termos uma url válida.*

6) Define os dados para contato com o desenvolvedor inseridos no método contact().

```
.contact(contact()).build()
```

Método private static Contact contact()

Define os dados do Desenvolvedor (Nome, Website e o E-mail).

```
return new Contact("Rafael Queiróz", "https://github.com/rafaelq80",  
"rafaelproinfo@gmail.com");
```

Método private static List responseMessage()

Define as mensagens personalizadas para os códigos de Resposta do protocolo http (http Response) para todos os verbos (GET, POST, PUT e DELETE). Cada linha é referente a um Status Code.

```
add(new ResponseBuilder().code("200").description("Sucesso!").build());
```

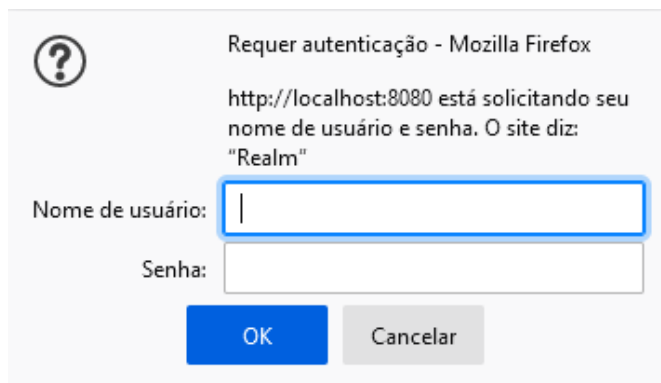
Salve todos arquivos e inicie a sua aplicação.

Executando o Swagger

Abra o seu navegador na Internet e digite o seguinte endereço abaixo para abrir a sua documentação.

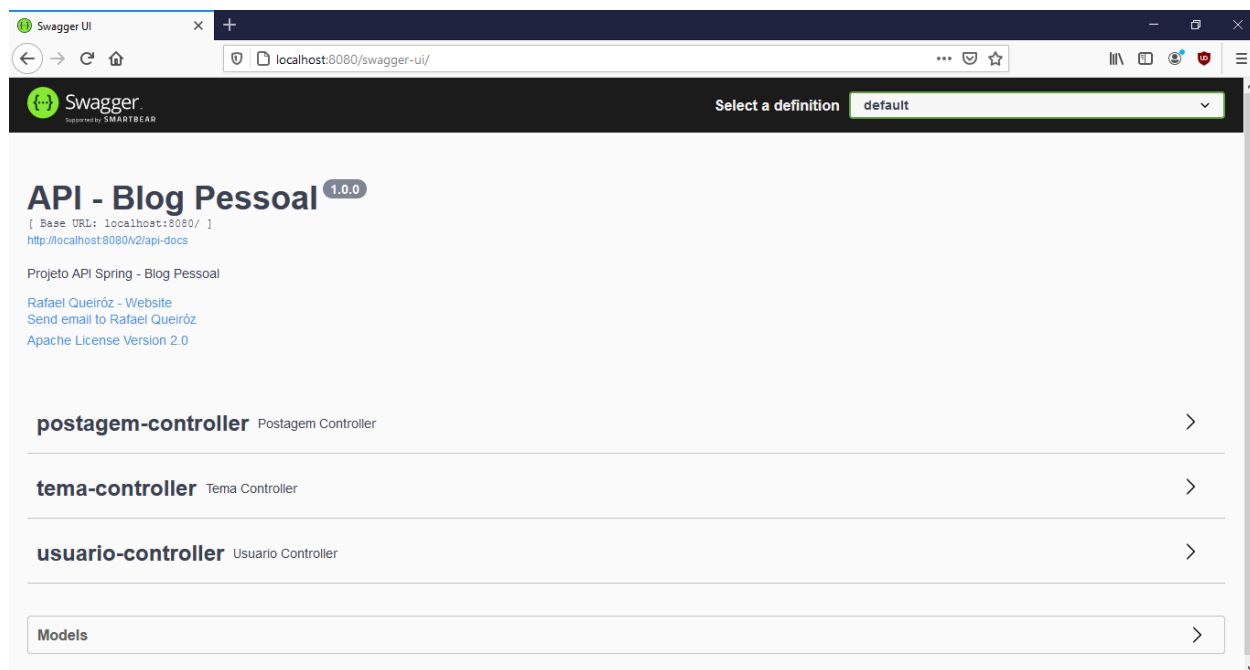
<http://localhost:8080/swagger-ui/>

Em aplicações com camadas de segurança, você precisará logar com uma conta de usuário antes de exibir a sua documentação no Swagger. Insira um usuário cadastrado no Banco de Dados local via Postman para fazer o login.



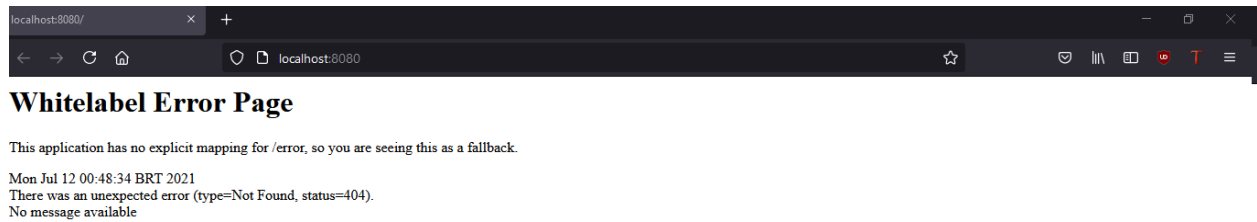
A screenshot of a Firefox authentication dialog box. It has a title bar with a question mark icon and the text 'Requer autenticação - Mozilla Firefox'. The main text says 'http://localhost:8080 está solicitando seu nome de usuário e senha. O site diz: "Realm"'. Below this are two input fields: 'Nome de usuário:' and 'Senha:'. At the bottom are two buttons: 'OK' (blue) and 'Cancelar' (grey).

Pronto! A sua documentação no Swagger está funcionando.



Definindo o Swagger como página principal da API

Vamos configurar o **Swagger** como página principal da nossa API, ou seja, ao digitarmos o endereço: <http://localhost:8080>, ao invés de abrir a página abaixo:



Abriremos a página do Swagger.

Na camada principal do nosso projeto Blog Pessoal (**br.org.generation.blogpessoal**) vamos abrir o arquivo **BlogpessoalApplication**

Vamos alterar o arquivo de:

```
package br.org.generation.blogpessoal;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class BlogpessoalApplication {

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

}
```

Para:

```
package br.org.generation.blogpessoal;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
import org.springframework.web.servlet.ModelAndView;

@SpringBootApplication
@RestController
@RequestMapping("/")
public class BlogpessoalApplication {

    @GetMapping
    public ModelAndView swaggerUi() {

        return new ModelAndView("redirect:/swagger-ui/");

    }

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

}
```

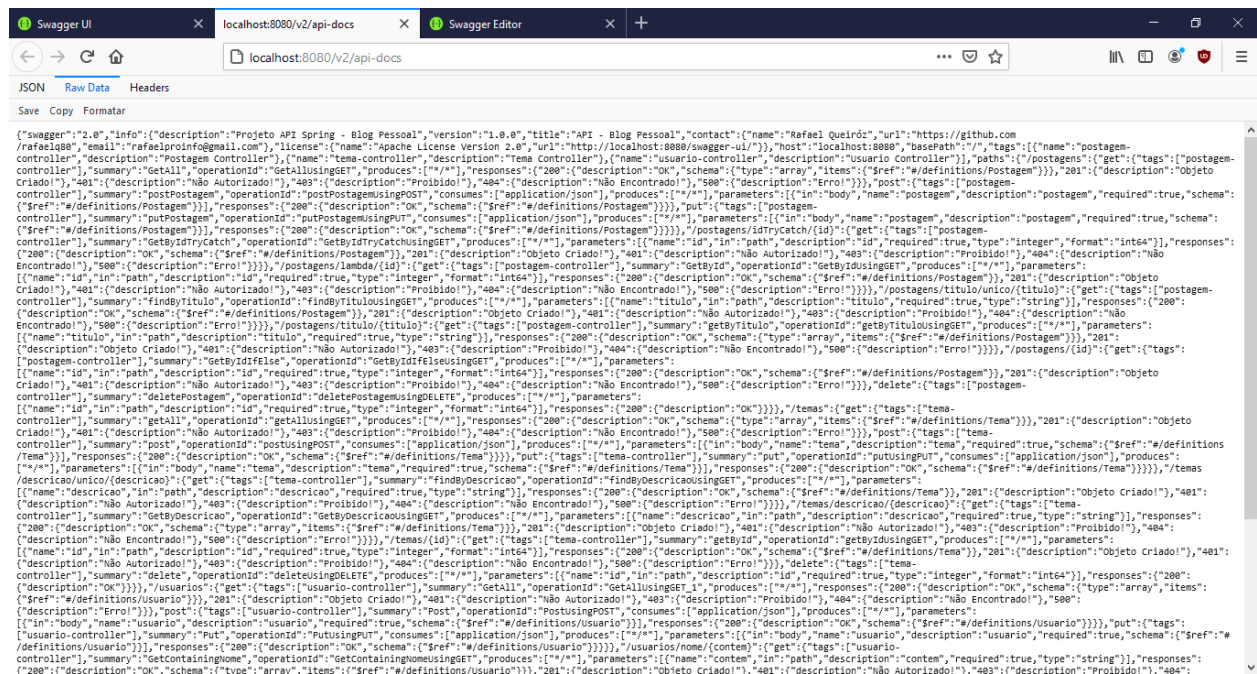
As alterações acima transformam a classe principal da nossa API (**BlogpessoalApplication**) em uma classe do tipo **RestController**, que responderá à todas as requisições do tipo **GET** para o **endpoint "/"** (raiz do nosso projeto) e fará o redirecionamento para o Swagger, ou seja, o Swagger será a página home do nosso projeto.

Gerando o PDF da Documentação

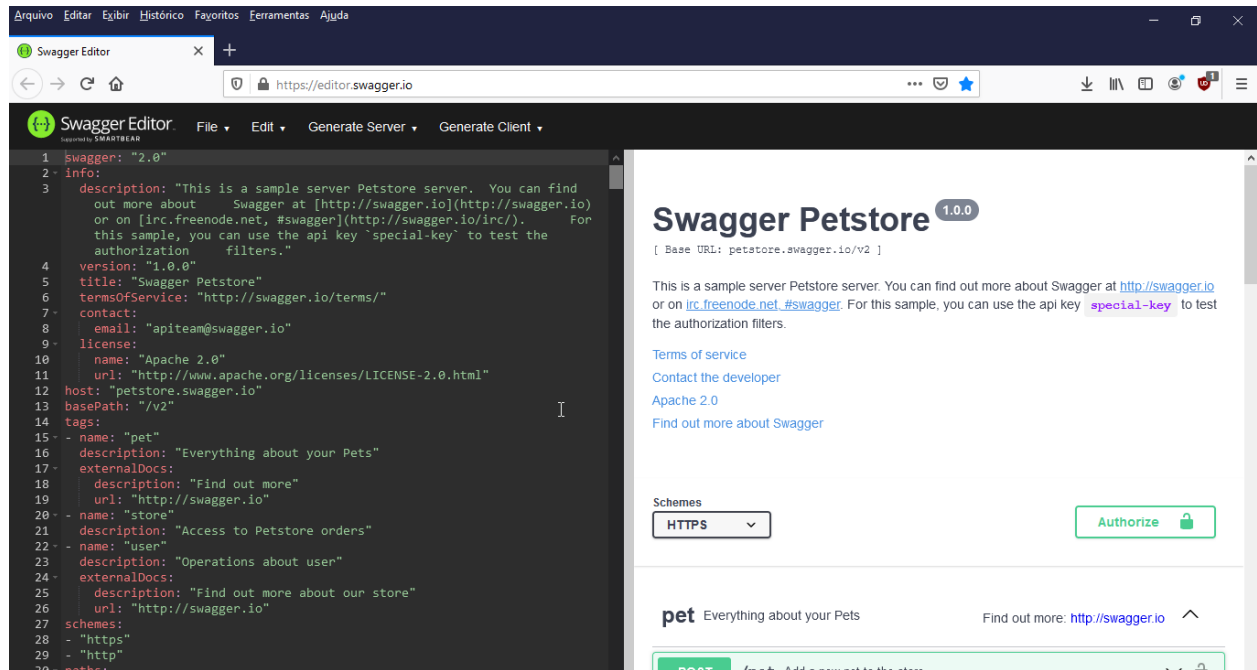
1. No Swagger, clique no link: <http://localhost:8080/v2/api-docs> para visualizar a documentação no formato JSON.



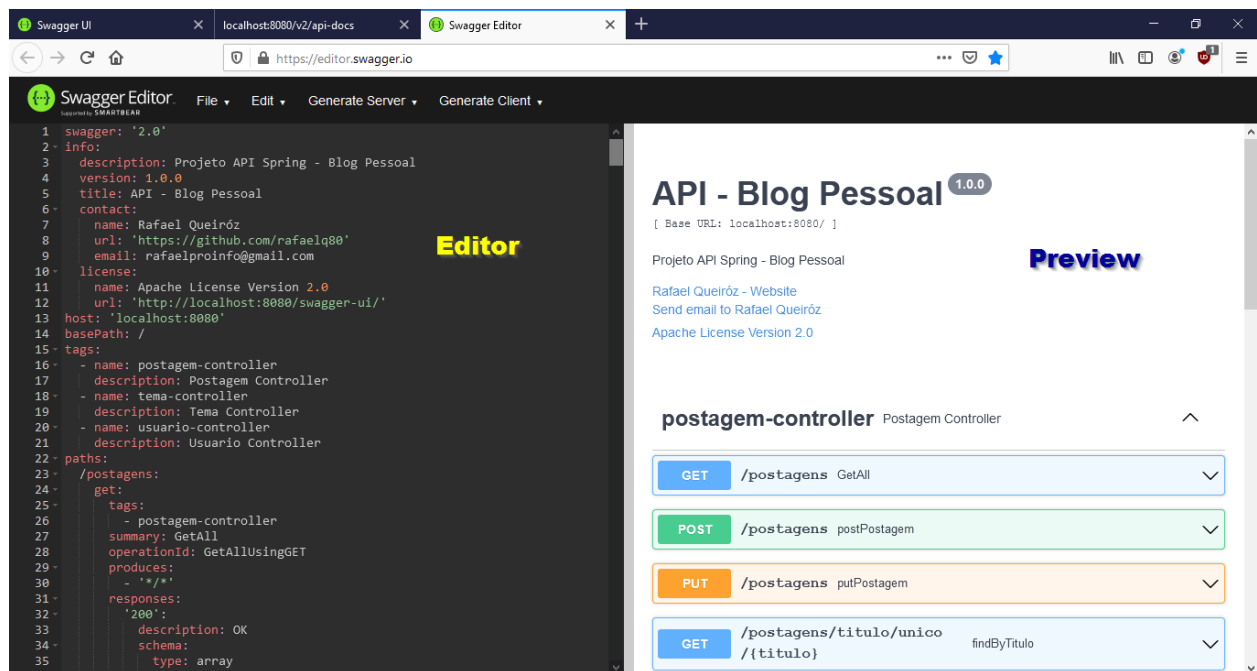
2. Visualize o código no formato Raw Data (No Chrome e no Edge é o formato padrão),
Selecione todo o código (Ctrl + A) e Copie (Ctrl + C).



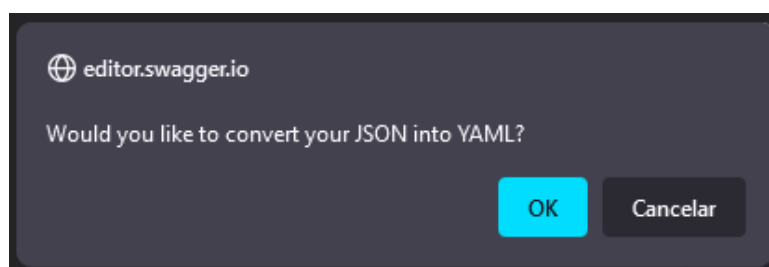
3) Acesse o site ****Swagger Editor**** (<https://editor.swagger.io/>).



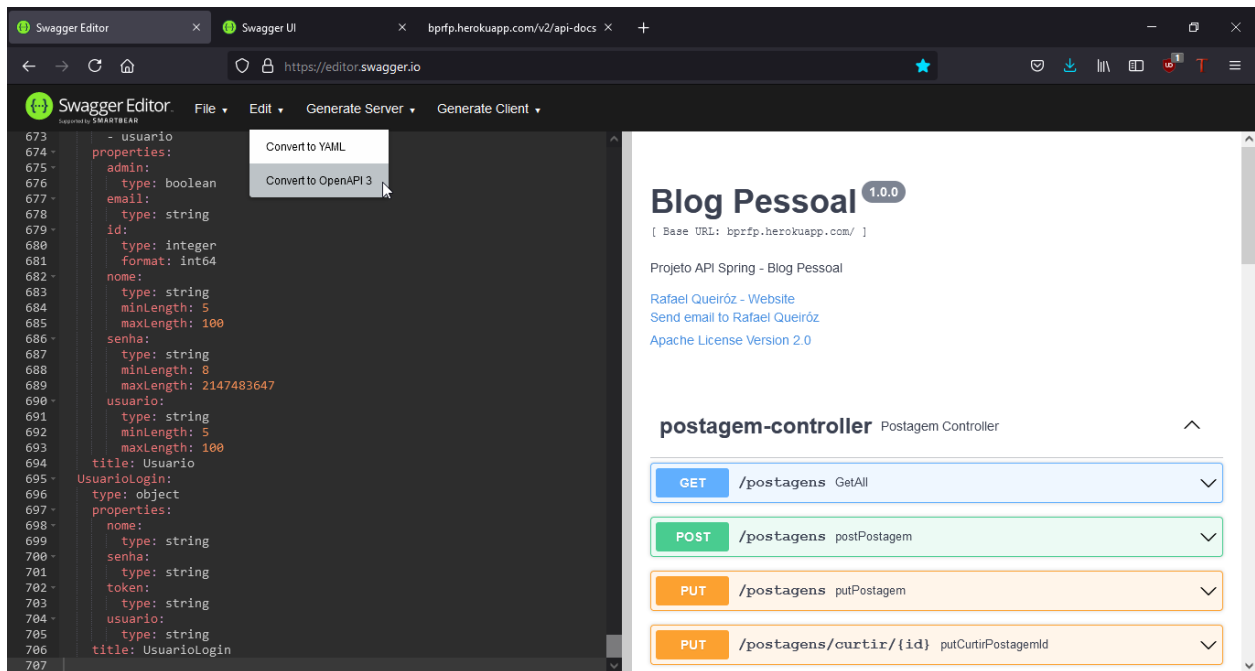
4. No **Swagger Editor**, apague o código exemplo e cole o código copiado da Documentação dentro do Editor (Ctrl + V).



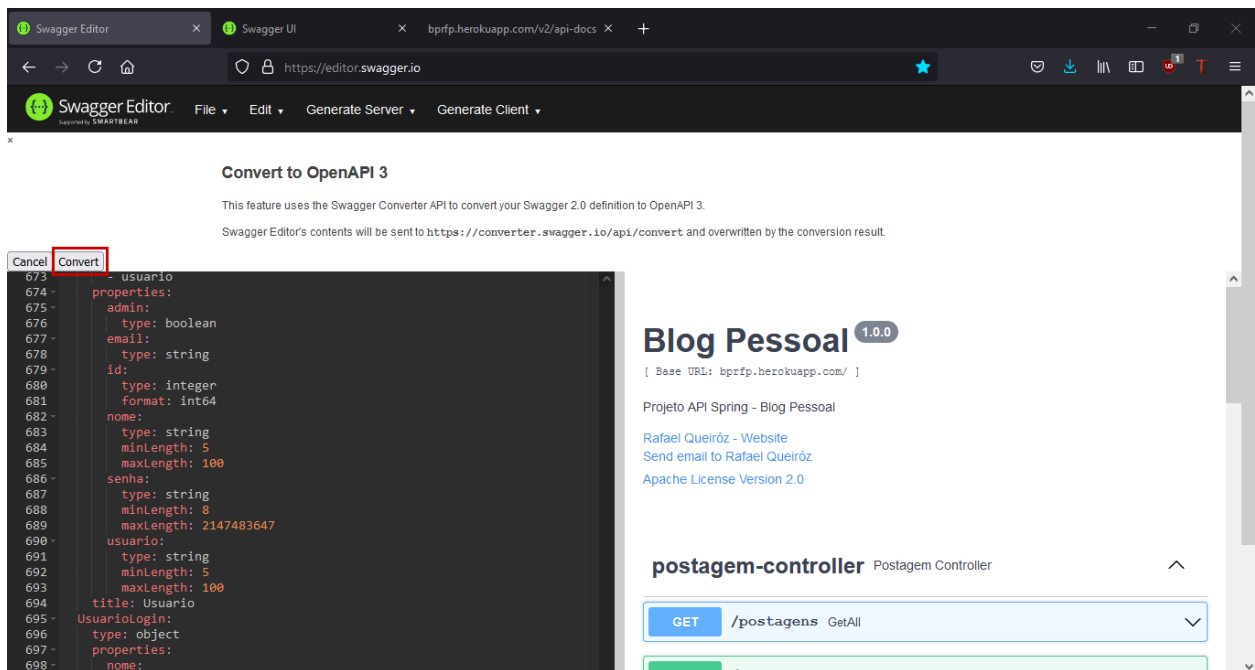
5. O Swagger Editor perguntará se você deseja converter o código JSON em YAML. Clique em OK para converter.



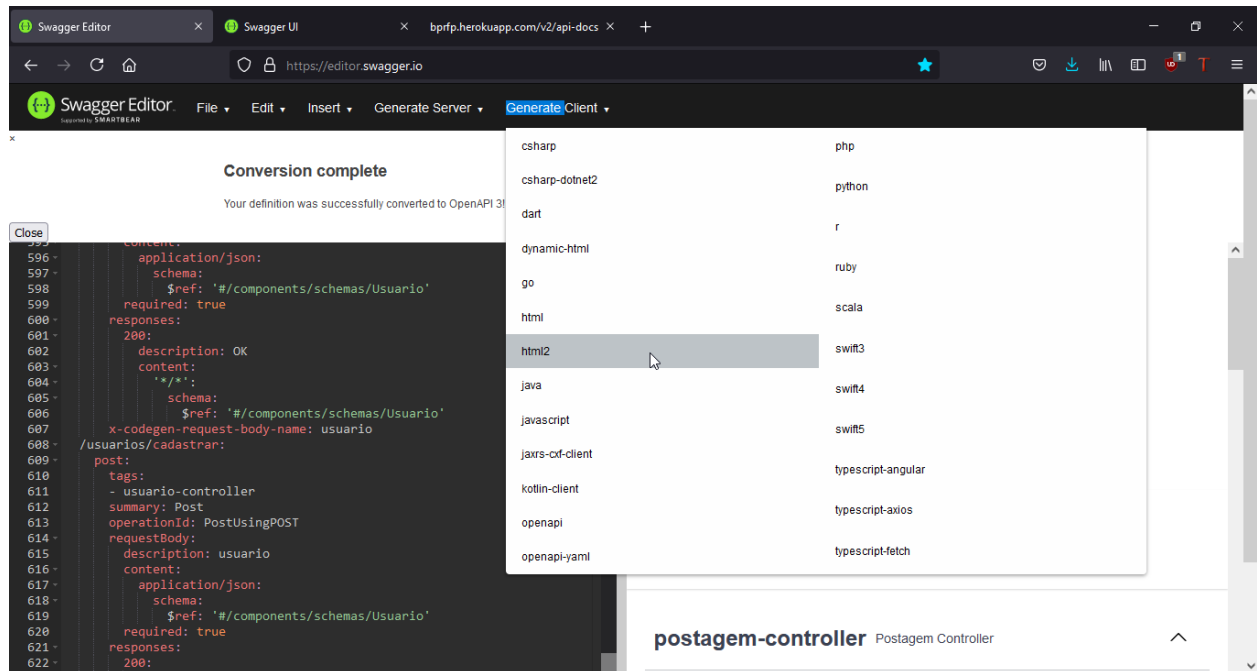
6. No Menu **Edit**, selecione a opção **Convert to OpenAPI3**



7. Em seguida, clique na guia **Convert**



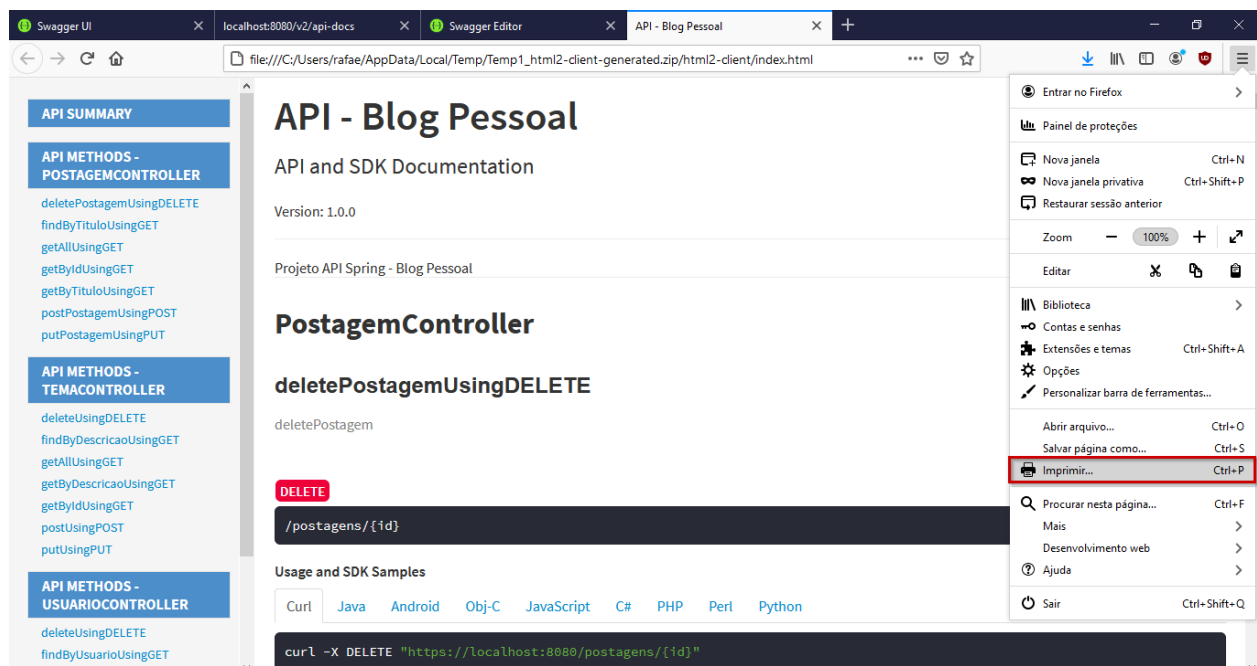
8) No menu **Generate Client**, selecione a opção **html2**.



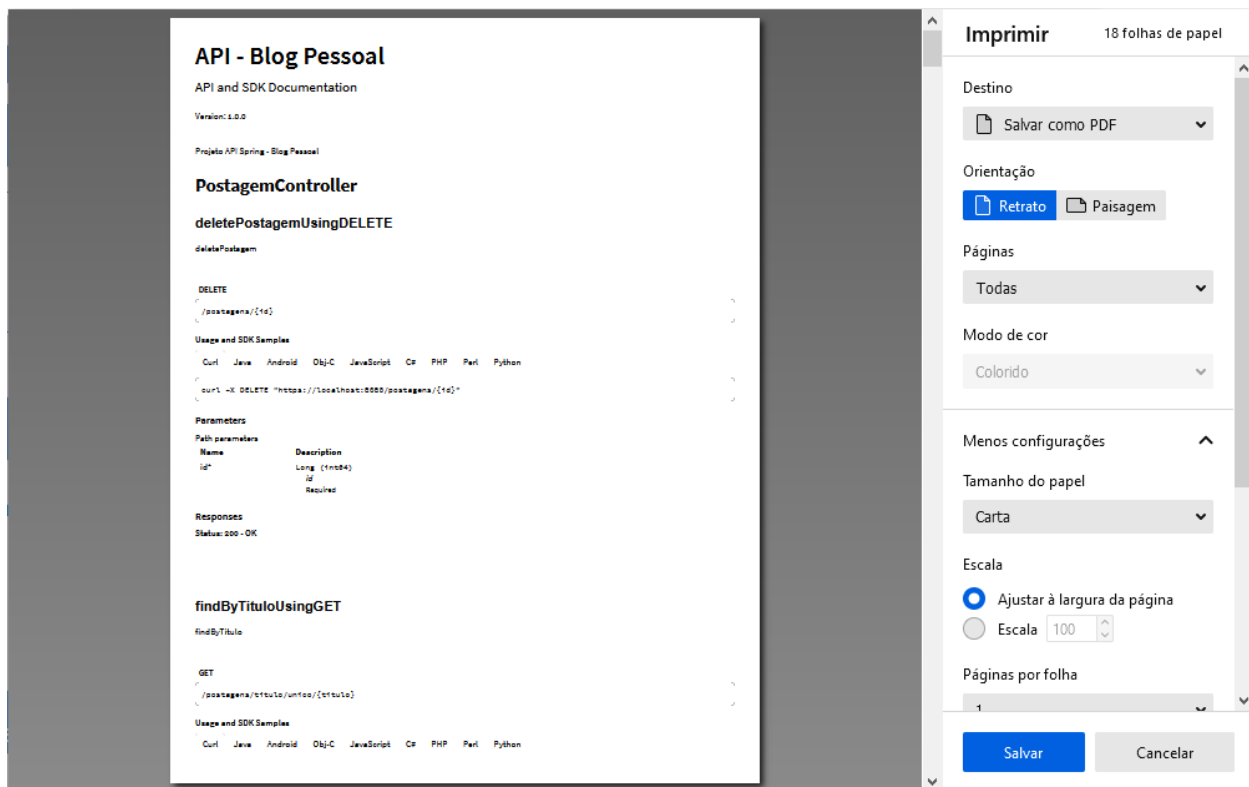
9. O **Swagger Editor** solicitará o download do arquivo **html2-client-generated.zip**. Faça o download do arquivo, descompacte no seu computador e abra o arquivo **index.html** no seu navegador.

Nome	Tipo	Tamanho Compact...
.swagger-codegen	Pasta de arquivos	
.swagger-codegen-ignore	Arquivo SWAGGER-CODE...	1 KB
index.html	Firefox Document	162 KB

10 No seu navegador, no menu principal, clique em **Imprimir**.



11. Na janela de impressão, no item **Destino**, selecione a opção **Salvar em PDF** e clique no Botão **Salvar**.



Documentação em PDF gerada!

