Deploy do Back end da API no Heroku

O que é Deploy?

O verbo **deploy**, em inglês, significa **implantar**.

Em programação, seu sentido está intimamente relacionado à sua tradução literal: fazer um deploy, em termos práticos, significa colocar no ar alguma aplicação que teve seu desenvolvimento concluído.
Quando um site é finalizado por um desenvolvedor e, após seus testes, é finalmente hospedado na nuvem e colocado no ar, ele passa pelo processo de deploy.
De mesmo modo, quando um sistema sofre alguma melhoria ou alteração em seu código-fonte, implementar essa alteração ao sistema que está no ar também é um tipo de deploy.

O que veremos por aqui?

Esse documento é um passo a passo para você subir (deploy) a sua API criada no SPRING gratuitamente para a nuvem e isso irá gerar um link de acesso a sua página que poderá ser acessado em qualquer lugar a partir de qualquer dispositivo com acesso a Internet.
Para realizar esse deploy vamos precisar fazer algumas modificações em nosso projeto, que serão detalhadas nas próximas páginas.

#Passo 01 - Criar a Documentação da API no Swagger

Para criar a Documentação da API no Swagger, utilize o ebook do Swagger.

#Passo 02 - Criação do usuário em memória

Vamos criar um usuário padrão em memória para simplificar o acesso a nossa API. O usuário em memória é um usuário para testes, que dispensa o cadastro no Banco de Dados. Quando a API estiver em produção este usuário deve ser desabilitado.

Na camada Security, abra o arquivo **BasicSecurityConfig** e altere o método **protected void configure(AuthenticationManagerBuilder auth) throws Exception** de:

```
@Override
protected void configure(AuthenticationManagerBuilder auth)
throws Exception {
    auth.userDetailsService(userDetailsService);
}
```

Para:

```
@Override
protected void configure(AuthenticationManagerBuilder auth)
throws Exception {
    auth.userDetailsService(userDetailsService);

    auth.inMemoryAuthentication()
    .withUser("root")
    .password(passwordEncoder().encode("root"))
    .authorities("ROLE_USER");
}
```

#Passo 03 - Atualização do método configure(HttpSecurity http)

Vamos fazer uma atualização no método **configure(HttpSecurity http)**, na Classe **BasicSecurityConfig**, na camada Security, para evitar erros do tipo 401 (Unauthorized) no envio de requisições via frontend no Heroku.

Na camada Security, abra o arquivo **BasicSecurityConfig** e altere o método **protected void configure(HttpSecurity http) throws Exception** de:

```
@Override
protected  void  configure(HttpSecurity  http) throws  Exception {

http.authorizeRequests()
    .antMatchers("/usuarios/cadastrar").permitAll()
    .antMatchers("/usuarios/logar").permitAll()
    .anyRequest().authenticated()
    .and().httpBasic()
    .and().sessionManagement()
    .sessionCreationPolicy(SessionCreationPolicy.STATELESS)
    .and().cors()
    .and().csrf().disable();
}
```

Para:

```
@Override
protected  void  configure(HttpSecurity  http) throws  Exception {

http.authorizeRequests()
    .antMatchers("/usuarios/cadastrar").permitAll()
    .antMatchers("/usuarios/logar").permitAll()
    .antMatchers(HttpMethod.OPTIONS).permitAll()
    .anyRequest().authenticated()
    .and().httpBasic()
    .and().sessionManagement()
    .sessionCreationPolicy(SessionCreationPolicy.STATELESS)
    .and().cors()
    .and().csrf().disable();
}
```

O parâmetro **HttpMethod.OPTIONS** permite que o cliente (frontend), possa descobrir quais são as opções de requisição permitidas para um determinado recurso em um servidor. Nesta implementação, está sendo liberada todas as opções das requisições através do método **permitAll()**.

#Passo 04 - Testar a API no seu computador

1.Execute a sua aplicação localmente pelo Eclipse ou pelo STS
2. Abra o endereço: http://localhost:8080/ no seu navegador
3. Verifique se o Swagger abre automaticamente
4. Caso a API solicite Usuário e senha, experimente o **Usuário: root** e a **Senha: root**, que foram criados em memória.
5. Aproveite para testar todos os Endpoints da aplicação no Swagger ou no Postman (/postagens, /temas e /usuarios).
6. Antes de continuar pare a execução do Projeto.

IMPORTANTE: Antes de fazer o Deploy a API deve estar funcionando sem erros.

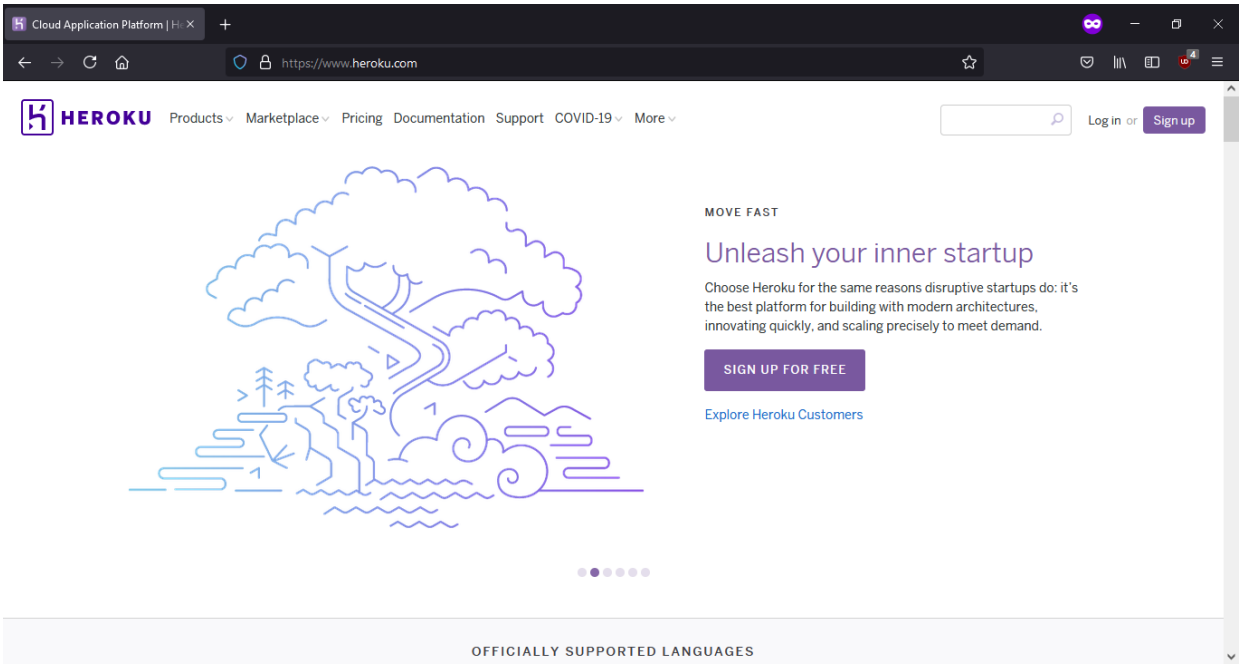
#Passo 05 - Criar uma conta grátis no Heroku

O **Heroku** é uma plataforma na nuvem que permite efetuar o Deploy de aplicações Back End ou Front End seja para hospedagem, testes em produção ou escalar as suas aplicações.

O grande diferencial do Heroku são as contas gratuitas, que permitem implantar até 5 Aplicações na mesma conta com Banco de Dados PostgreSQL incluso.

Para criar a sua conta grátis:

1- Acesse o endereço: https://www.heroku.com



2- Crie a sua conta grátis no Heroku clicando no botão **SIGN UP FOR FREE** e siga as instruções.

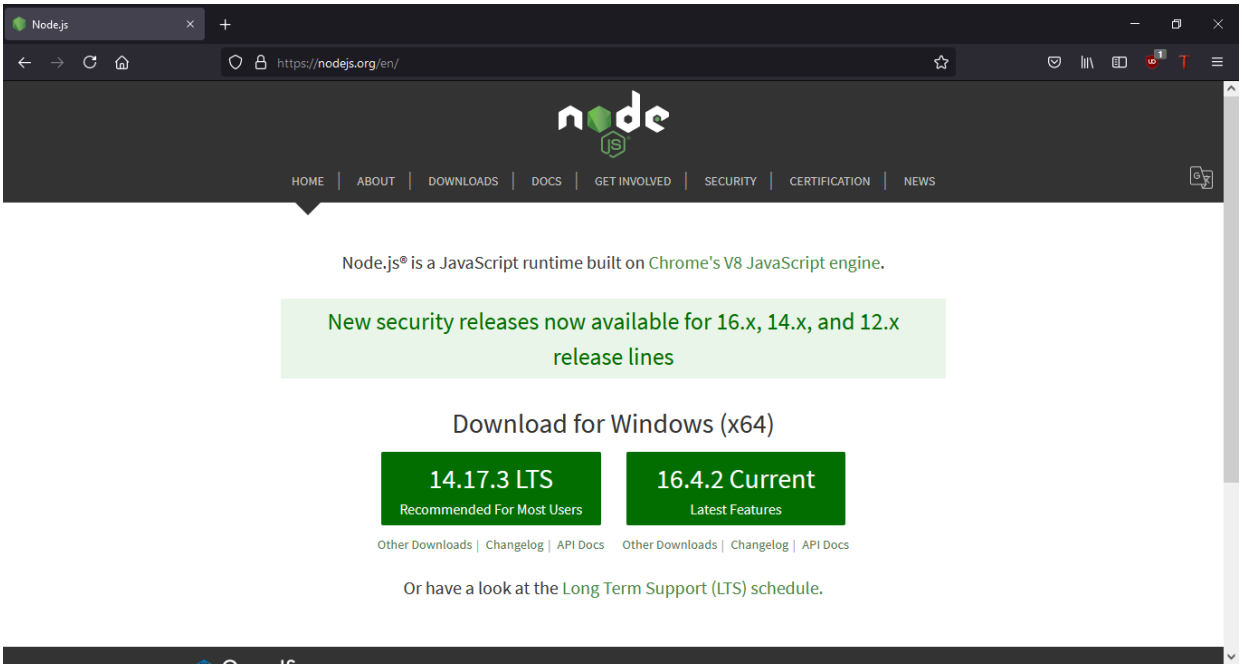
#Passo 06 - Instalação do Node.js

O Node.js pode ser definido como um **ambiente de execução Javascript**. Isso significa que com o Node.js é possível criar aplicações Javascript e executar como uma aplicação local em uma máquina, não dependendo de um navegador, como estamos acostumados.

O **NPM** é um gerenciador de pacotes para a linguagem de programação Javascript, semelhante ao Maven do Java, que permite instalar pacotes Javascript no nosso computador.

Para realizarmos o nosso Deploy utilizaremos o NPM para instalar o **Heroku Client**.

1- Acesse o endereço: https://nodejs.org/en/



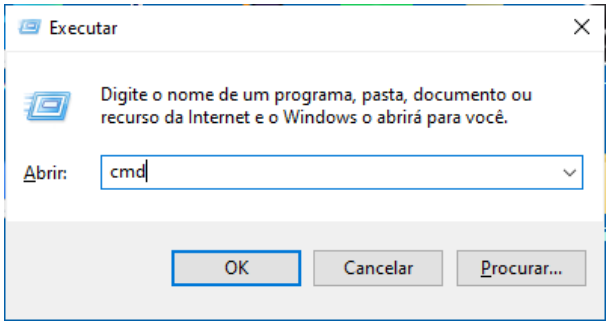
2- Faça o download da versão LTS do Node.js e instale no seu computador.

Em caso de dúvidas, acesse o Guia de instalação do Node.js

#Passo 07 - Instalação do Heroku Client

O **Heroku Client** é uma Interface de linha de comando (CLI) do Heroku, que facilita a criação e o gerenciamento de seus aplicativos Heroku diretamente do terminal.
Para instalar e executar os comandos do Heroku Client usaremos o Prompt de comando do Windows.

1- Para instalar, execute o atalho  para abrir a janela Executar



2- Digite o comando **cmd** para abrir o **Prompt de comando do Windows**

3- Antes de instalar o **Heroku Client**, verifique se o Node já está instalado através do comando:

```
npm -version
```

```
7.19.1
```

**A versão pode ser diferente da imagem*

4- Para instalar o **Heroku Client** digite o comando:

```
npm i -g heroku
```

```
npm WARN deprecated strip-eof@2.0.0: Renamed to `strip-final-newline` to better represent its functionality.
npm WARN deprecated debug@4.1.1: Debug versions >=3.2.0 <3.2.7 || >=4 <4.3.1 have a low-severity ReDos regression when used in a Node.js
environment. It is recommended you upgrade to 3.2.7 or 4.3.1. (https://github.com/visionmedia/debug/issues/797)
npm WARN deprecated uuid@3.2.1: Please upgrade  to version 7 or higher.  Older versions may use Math.random() in certain circumstances,
which is known to be problematic.  See https://v8.dev/blog/math-random for details.
npm WARN deprecated uuid@3.3.2: Please upgrade  to version 7 or higher.  Older versions may use Math.random() in certain circumstances,
which is known to be problematic.  See https://v8.dev/blog/math-random for details.

added 13 packages, changed 620 packages, and audited 691 packages in 2m

22 packages are looking for funding
  run `npm fund` for details

7 vulnerabilities (2 moderate, 5 high)

To address all issues, run:
  npm audit fix

Run `npm audit` for details.
npm notice
npm notice New minor version of npm available! 7.17.0 -> 7.19.1
npm notice Changelog: https://github.com/npm/cli/releases/tag/v7.19.1
npm notice Run npm install -g npm@7.19.1 to update!
npm notice
```

5- Confirme a instalação do Heroku Client através do comando:

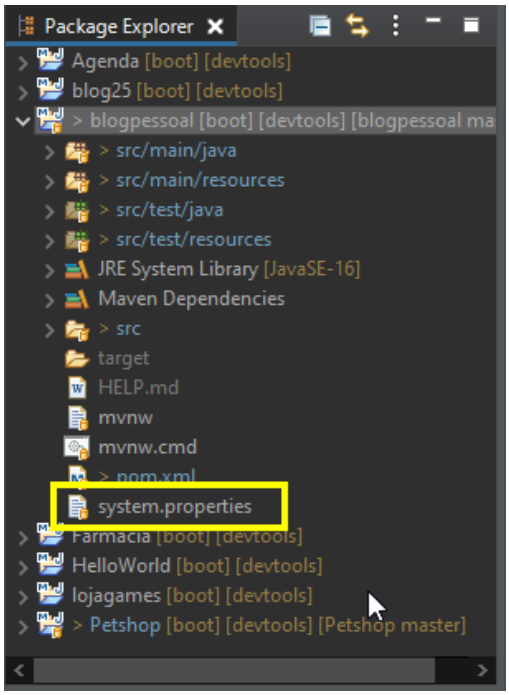
```
heroku version
```

```
heroku/7.56.0 win32-x64 node-v14.17.3
```

**A versão pode ser diferente da imagem*

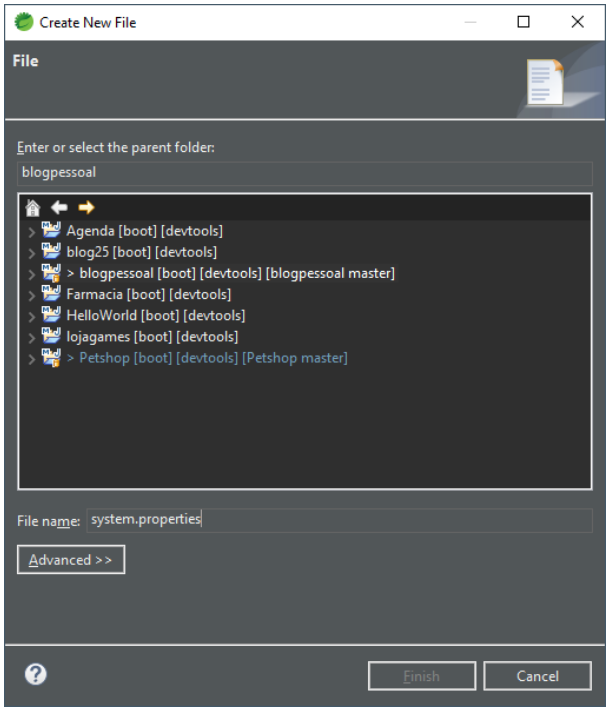
#Passo 08 - Criação do arquivo system.properties

1- Na raiz do seu projeto (na pasta blogpessoal), crie o arquivo **system.properties**.



2- Na Guia **Package explorer**, clique com o botão direito do mouse e clique na opção **New->File**.

3- Em **File name**, digite: **system.properties** e clique no botão **Finish**.



4- No arquivo **system.properties** indique a versão do Java que será utilizada no Heroku:

```
java.runtime.version=16
```

#Passo 09 - Configuração do PostgreSQL no arquivo pom.xml

No arquivo, **pom.xml**, vamos alterar as linhas:

```
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <scope>runtime</scope>
</dependency>
```

Para:

```
 <dependency>
    <groupId>org.postgresql</groupId>
    <artifactId>postgresql</artifactId>
</dependency>
```

#Passo 10 - Configuração do Banco de Dados no arquivo application.properties

No arquivo, **application.properties**, vamos alterar as linhas:

```
spring.jpa.hibernate.ddl-auto=update
spring.jpa.database=mysql
spring.datasource.url=jdbc:mysql://localhost/db_blogpessoal?
createDatabaseIfNotExist=true&serverTimezone=America/Sao_Paulo&useSSl=false
spring.datasource.username=root
spring.datasource.password=root
spring.jpa.properties.hibernate.dialect =
org.hibernate.dialect.MySQL8Dialect

spring.jpa.show-sql=true

spring.jackson.date-format=yyyy-MM-dd HH:mm:ss
spring.jackson.time-zone=Brazil/East
```

Observe que a linha 3 do arquivo application.properties (nome do banco de dados) está dividida em 2 linhas no pdf. No STS/Eclipse mantenha em uma única linha.

Para:

```
spring.jpa.generate-ddl=true
spring.datasource.url=${JDBC_DATASOURCE_URL}
spring.jpa.show-sql=true

spring.jackson.date-format=yyyy-MM-dd HH:mm:ss
spring.jackson.time-zone=Brazil/East
```

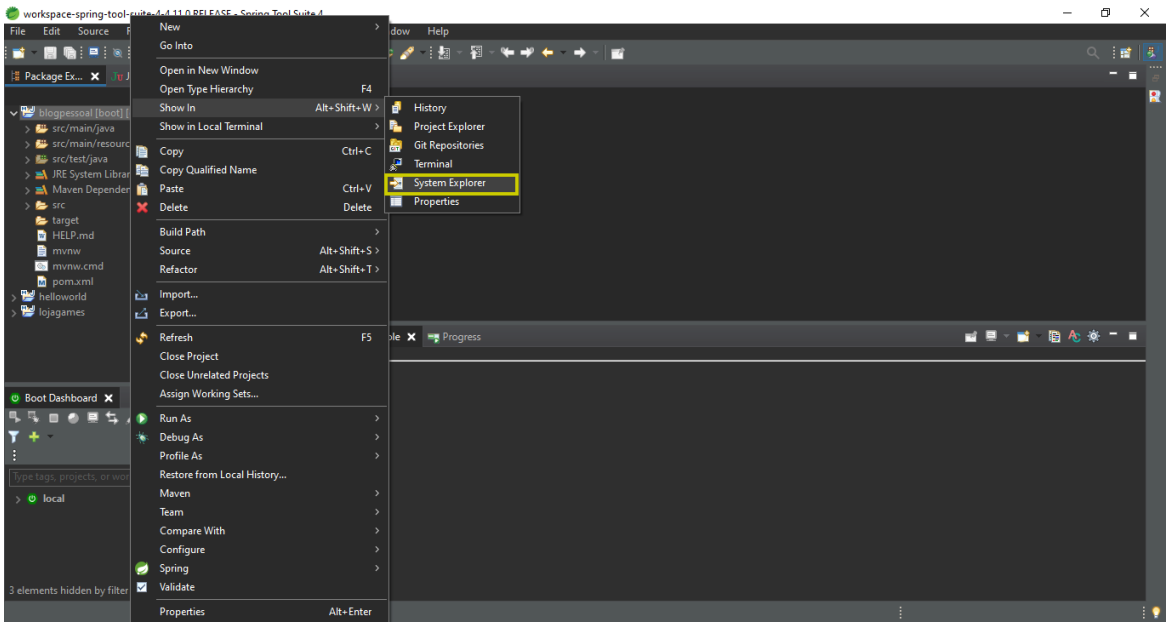
*** Importante ***

A partir deste ponto, **o seu projeto não executará mais localmente (http://localhost:8080/)**. Para voltar a executar localmente, será necessário **desfazer as configurações efetuadas nos arquivos pom.xml e application.properties**.

#Passo 11 - Deploy com o Git

Vamos preparar o nosso repositório local para subir a aplicação para o Heroku utilizando o Git.

1- Na pasta do projeto, clique com o botão direito do mouse e na sequência clique na opção: **Show in => System Explorer**



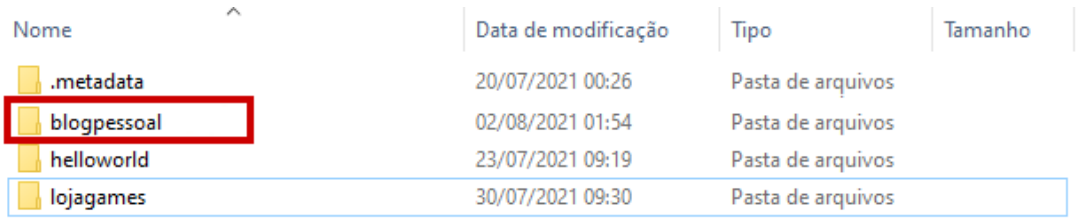
2- Será aberta a pasta Workspace onde o Eclipse/STS grava os seus projetos:

- Se você estiver usando o STS geralmente a pasta fica em: **c:\Usuarios\seu usuario\Documents\workspace-spring-tool-suite-4-4.11.0.RELEASE** (a versão pode ser diferente).

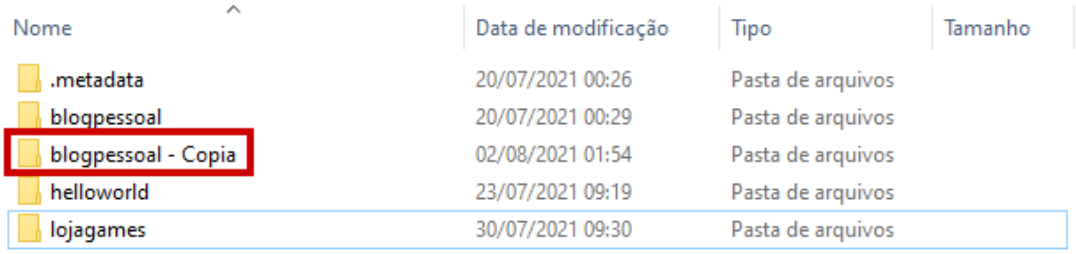
- Se você estiver utilizando o Eclipse, geralmente a pasta fica em: **c:\Usuarios\seu usuario\eclipse-workspace**.

  **seu usuario = Usuário do seu computador*

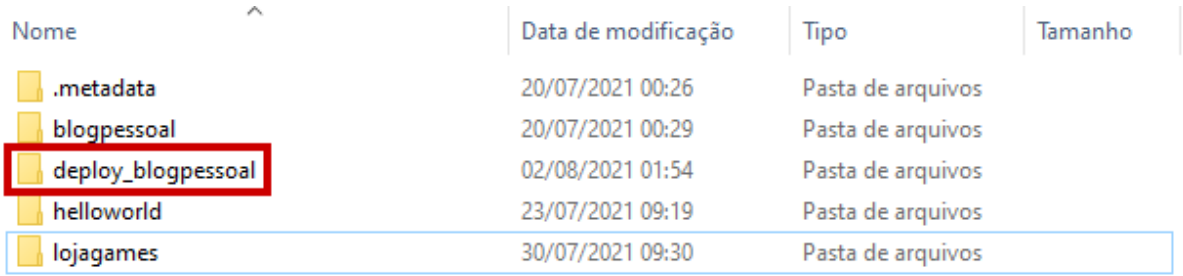
3- Copie a pasta da API: ***blogpessoal*** (o nome da sua pasta pode ser diferente)



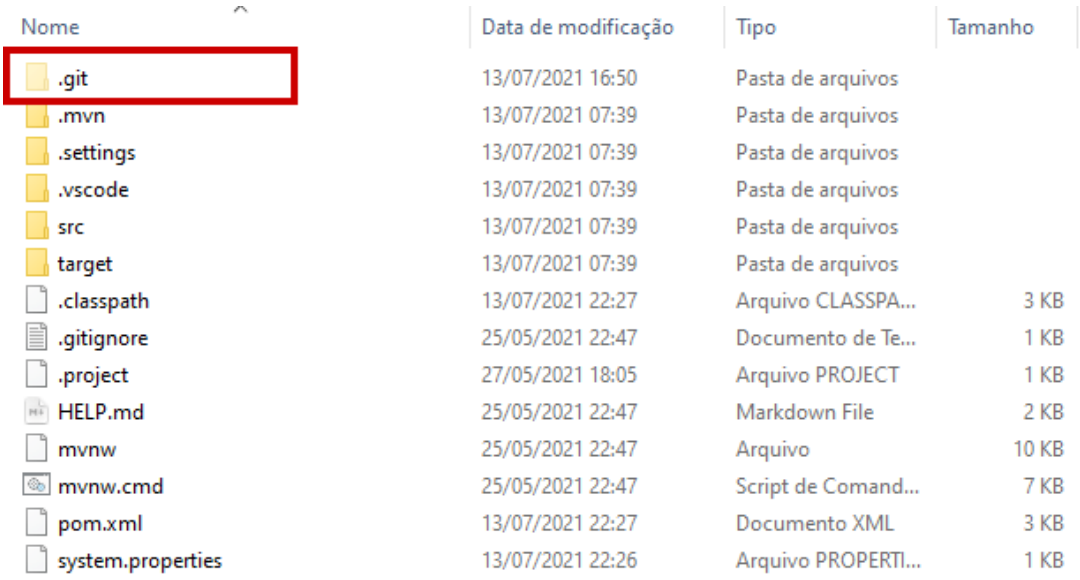
4- Cole a pasta no mesmo diretório



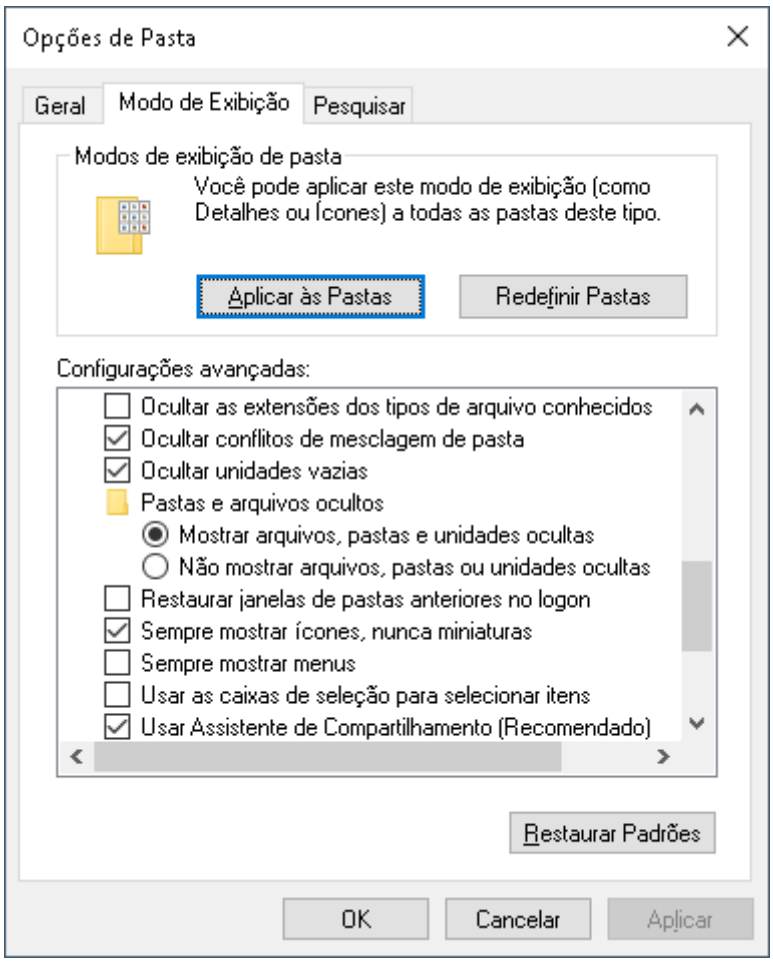
5- Renomeie a pasta para deploy_blogpessoal



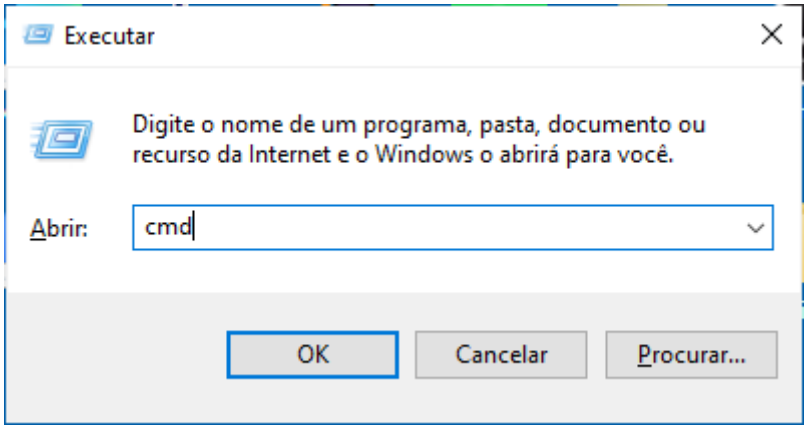
6- Abra esta pasta e verifique se existe uma pasta chamada .git. Caso exista, apague esta pasta. **Esta pasta estará presente APENAS se você inicializou o git dentro dela.**



Caso esta pasta não esteja sendo exibida, na janela do Windows Explorer, clique na ***Guia Exibir*** *e na sequência no botão* ***Opções****. Na janela* ***Opções de Pasta****, na* ***Guia Modo de Exibição****, no item* ***Configurações avançadas****, localize a opção:* ***Pastas e arquivos ocultos*** *e marque a opção* ***Mostrar arquivos, pastas e unidades ocultas*** *(como mostra a figura abaixo). Em seguida clique em* ***OK*** *para concluir.*



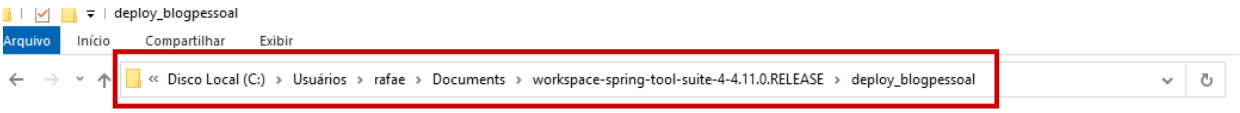
7- Execute o atalho  para abrir a janela Executar



8- Digite o comando abaixo para abrir o **Prompt de Comando do Windows**:

```
cmd
```

9- Na pasta do seu projeto, no Windows explorer, copie o caminho da pasta coforme a figura abaixo:



10- No Prompt de comando do Windows digite os comando cd e cole na frente do comando o caminho copiado:

```
cd C:\Users\seu usuario\Documents\
workspace-spring-tool-suite-4-4.11.0.RELEASE\deploy_blogpessoal
```

**o nome da pasta pode ser diferente*

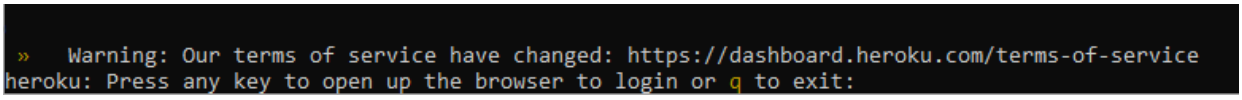
11- Digite a sequência de comandos abaixo para inicializar o seu repositório local para efetuar o Deploy no Heroku:

```
git init
git add .
git commit -m “Deploy inicial - Blog Pessoal”
```

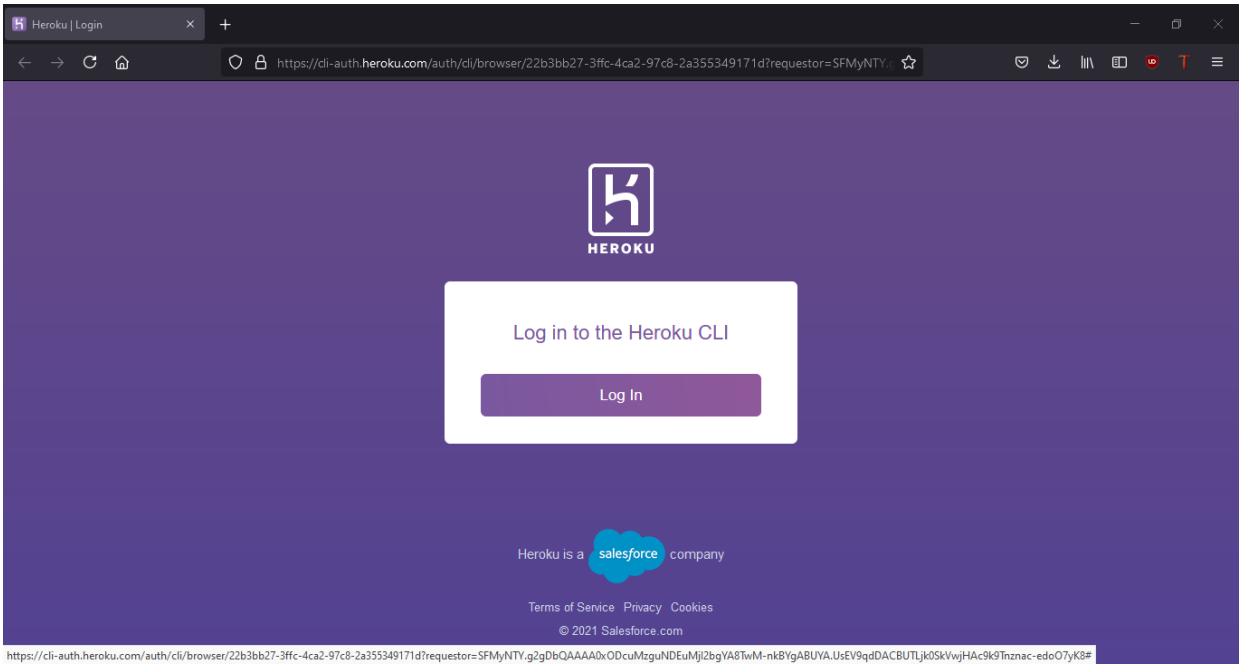
#Passo 12 - Login no Heroku

1- Digite o comando:

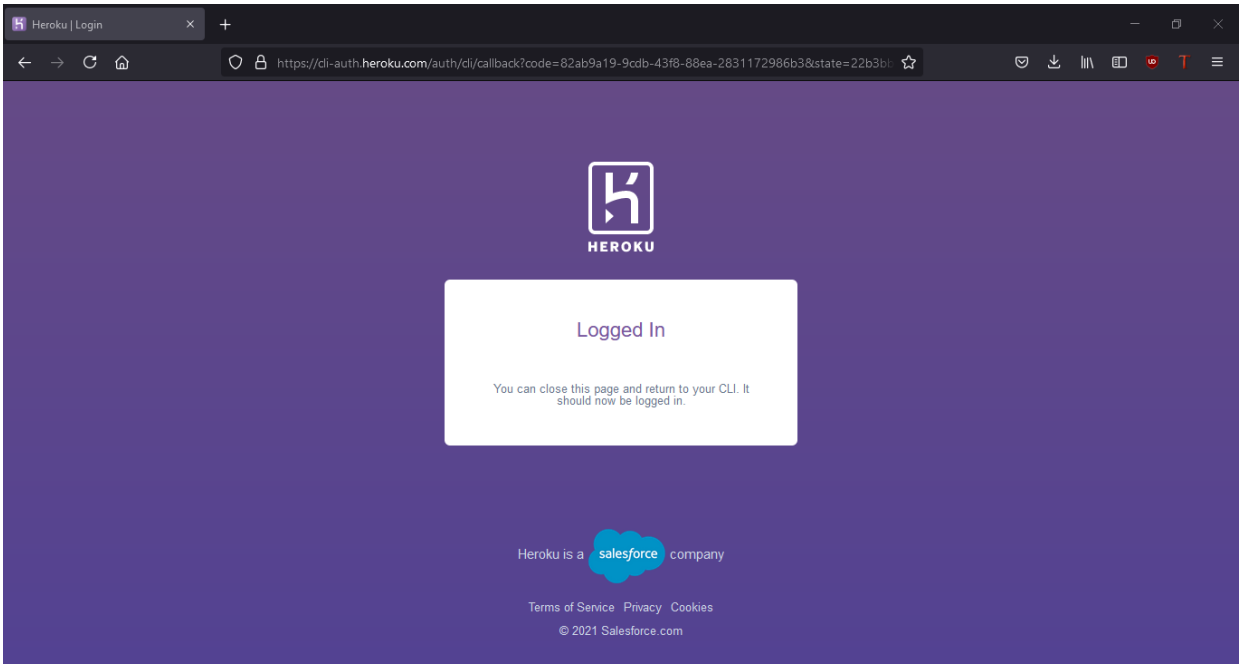
```
heroku login
```



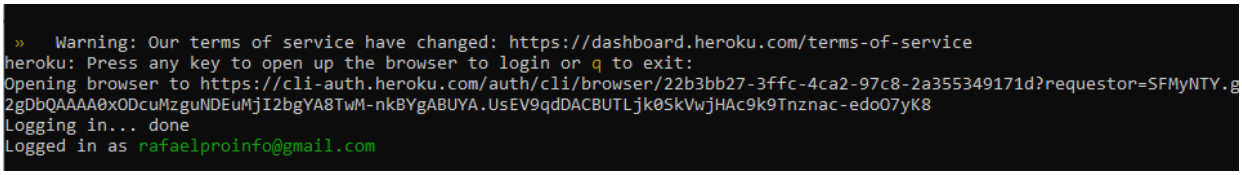
2- Será aberta a janela abaixo. Clique no botão **Log in**



3- Após efetuar o login na sua conta, será exibida a janela abaixo.



4- Volte para o Prompt de comando para continuar o Deploy.



#Passo 13 - Criar um novo projeto no Heroku

Para criar um novo projeto na sua conta do Heroku, digite o comando:

```
heroku create nomedoprojeto
```

*** Importante ***

O NOME DO PROJETO NÃO PODE TER LETRAS MAIUSCULAS, NUMEROS OU CARACTERES ESPECIAIS. ALÉM DISSO ELE PRECISA SER UNICO DENTRO DA PLATAFORMA HEROKU.

Se o nome escolhido já estiver em uso no Heroku, será exibida a mensagem abaixo:

Se o nome escolhido estiver disponível, será exibida a mensagem abaixo:

#Passo 14 - Adicionar o Banco de dados (PostgreSQL) no Heroku

Para adicionar um Banco de Dados PostgreSQL no seu projeto, digite o comando:

```
heroku addons:create heroku-postgresql:hobby-dev -a nomedoprojeto
```

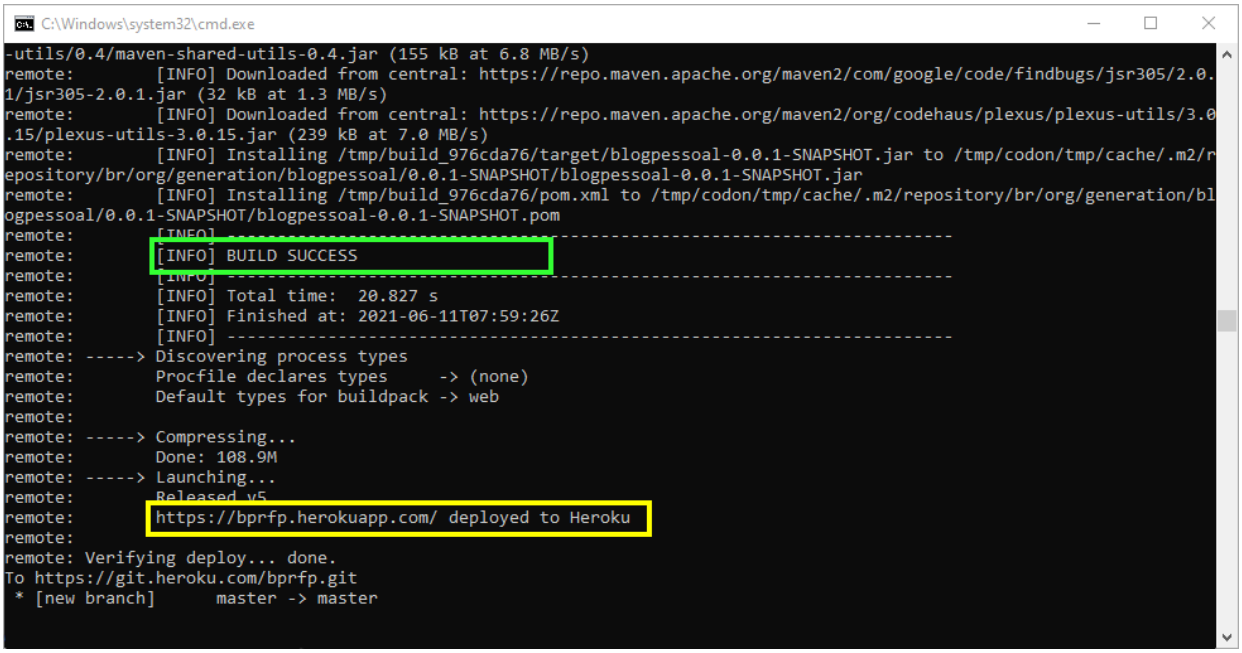
```
Creating heroku-postgresql:hobby-dev on ⬢ bprfp... free
Database has been created and is available
 ! This database is empty. If upgrading, you can transfer
 ! data from another database with pg:copy
Created postgresql-flexible-10004 as DATABASE_URL
Use heroku addons:docs heroku-postgresql to view documentation
```

#Passo 15 - Efetuar o Deploy

Para concluir o Deploy, digite o comando:

```
git push heroku master
```

Se tudo deu certo, será exibida a mensagem **BUILD SUCESS** (destacado em verde na imagem) e será exibido o endereço (https://nomedoprojeto.herokuapp.com) para acessar a API na Internet (destacado em amarelo na imagem)

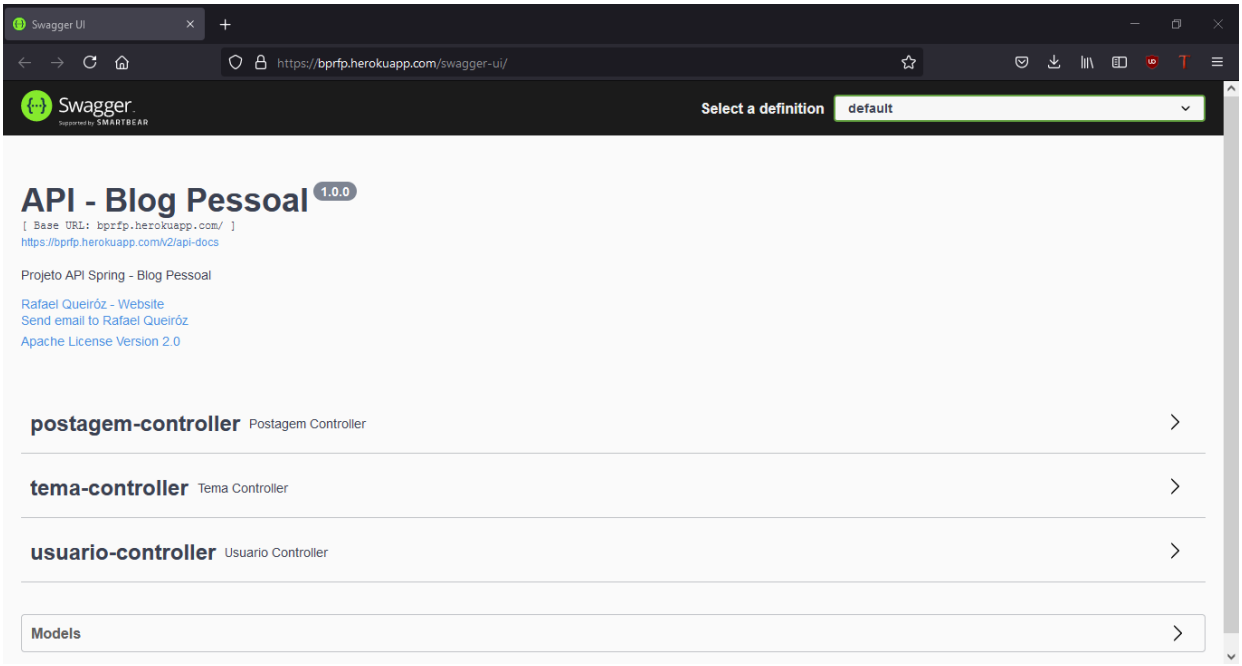


#Passo 16 - Testar o link e a API

1- Abra o navegador, copie op endereço do Deploy e cole na Barra de endereço do navegador.

2- Será solicitado o usuário e a senha. Digite **root** para ambos.

3- Sua API deverá abrir o Swagger.



4- Faça alguns testes via Swagger para certificar-se de que tudo está funcionando

Atualizar o Deploy no Heroku

Uma vez que o Deploy foi feito no Heroku, assim como no Github, basta atualizar os arquivos na pasta **deploy_blogpessoal** e efetuar a sequência de comandos abaixo para atualizar o Deploy.

```
git add .
git commit -m “Atualização do Deploy - Blog Pessoal”
git push heroku master
```

Caso ocorra algum erro de vinculação (link), verifique se a pasta está vinculada ao Heroku utilizando o comando abaixo:

```
git remote
```

Caso não apareça o resultado heroku, utilize o comando abaixo para vincular a pasta com o heroku.

```
heroku git:remote -a project
```

Caso o comando acima falhe, inicialize o repositório git e refaça a vinculação.

```
git init
heroku git:remote -a project
```

Para atualizar o Deploy, utilize os comandos baixo:

```
git add .
git commit -m “Atualização do Deploy - Blog Pessoal”
git push heroku master
```

Caso o ultimo comando falhe, acrescente a opção -f para forçar o Deploy.

```
git push -f heroku master
```

Se todas as opções acima falharem, verifique se o erro não está na aplicação.