

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

Para criar a Documentação da API no Swagger, utilize o Guia de Configuração do Swagger.

Passo 02 - Checar se o usuário root foi criado em memória

Verifique se o usuário em memória (root) foi criado. O usuário em memória é um usuário para testes, gerado na memória do servidor, que dispensa o cadastro no Banco de Dados. Em produção é altamente recomendado que este usuário seja desabilitado.

Na camada Security, abra o arquivo **BasicSecurityConfig** e verifique se o método **protected void configure(AuthenticationManagerBuilder auth) throws Exception** está igual ao código abaixo:

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

    auth.userDetailsService(userDetailsService);

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

}
```

Observe que está sendo criado o usuário **root**, com a senha criptografada **root** e com os direitos de acesso de um usuário comum (**ROLE_USER**).

⚠ **IMPORTANTE:** Não altere o nome e a senha do usuário em memória. Os instrutores da sua turma utilizarão este usuário para abrir, testar e corrigir a sua API.

Passo 03 - Verifique o método **configure(HttpSecurity http)**

Verifique se foi adicionada a linha **.antMatchers(HttpMethod.OPTIONS).permitAll()**, no método **configure(HttpSecurity http)**, da Classe **BasicSecurityConfig**, para evitar Respostas do tipo **401 (Unauthorized)** nas Requisições enviadas pelo Frontend para a nossa API que será hospedada no Heroku.

Na camada Security, abra o arquivo **BasicSecurityConfig** e verifique se o método **protected void configure(HttpSecurity http) throws Exception** está igual ao código abaixo:

```
@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.S
        .and().cors()
        .and().csrf().disable();
}
```

A linha adicionada com o parâmetro **HttpMethod.OPTIONS** permite que o Frontend descubra quais são as opções das requisições para um determinado recurso em um servidor. Nesta implementação, está sendo liberada todas as opções das requisições através do método **permitAll()**.

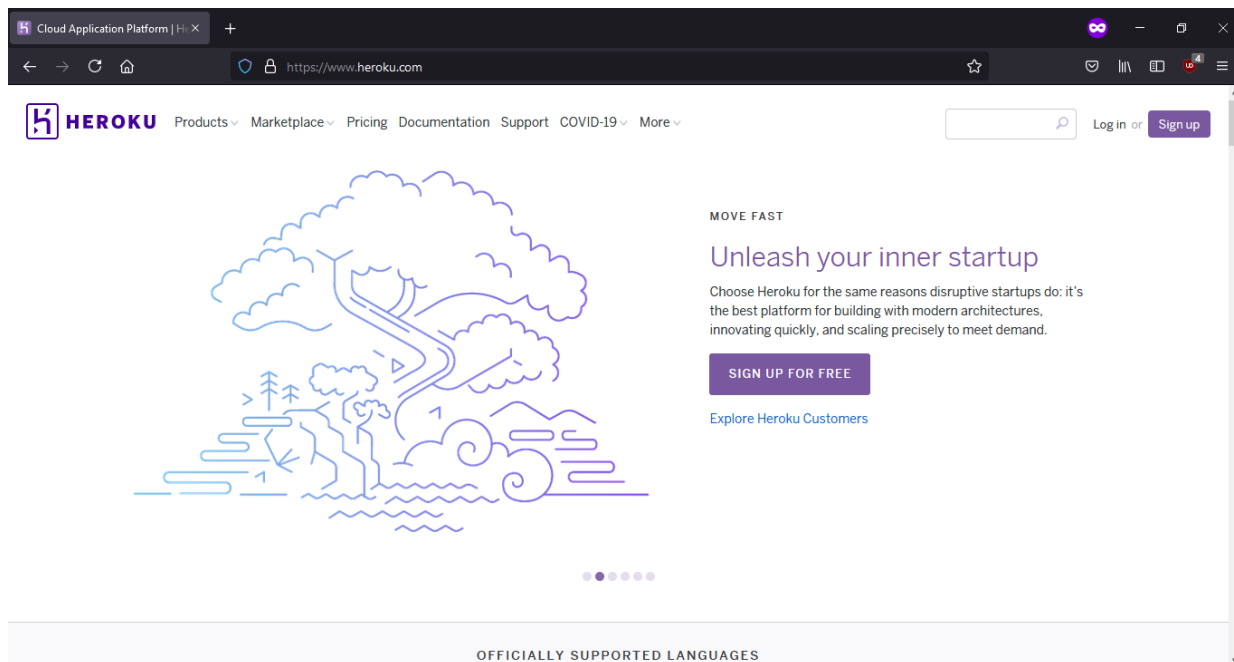
Passo 04 - Testar a API no seu computador

1. Execute a sua aplicação localmente pelo Eclipse/STS
2. Abra o endereço: <http://localhost:8080/> no seu navegador
3. Verifique se o **Swagger** está inicializando automaticamente
4. Caso a API solicite **Usuário** e **Senha**, experimente o **Usuário: root** e a **Senha: root**, que foram criados em memória no passo 02.
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: Lembre-se que antes de fazer o Deploy é fundamental que a API esteja rodando e sem erros.

Passo 05 - Criar uma conta grátis no Heroku

1. Acesse o endereço: <https://www.heroku.com>

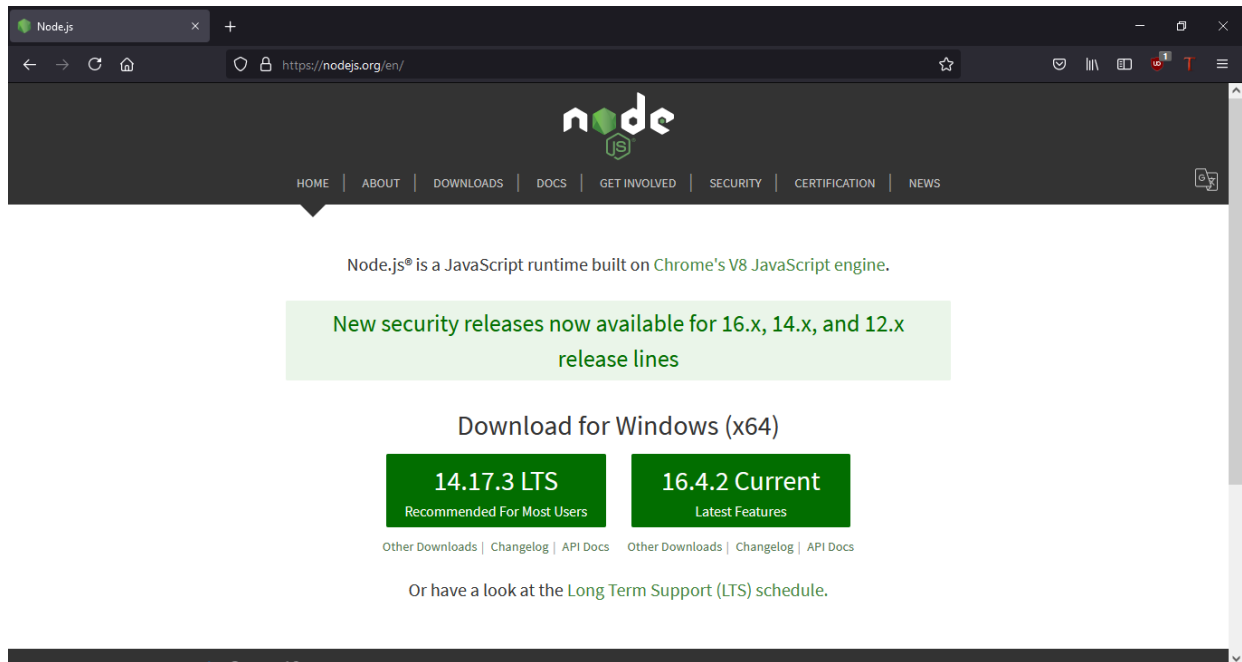


2. Crie a sua conta grátis no Heroku clicando no botão **SIGN UP FOR FREE** e siga as instruções.

⚠ IMPORTANTE: Não habilite em sua conta no Heroku a opção MFA (Multi-Factor Authentication), ou seja, o login em 2 etapas. Em alguns servidores do Heroku não é possível efetuar login via Heroku Client caso esta opção esteja habilitada.

Passo 06 - Instalação do Node.js

1. Acesse o endereço: <https://nodejs.org/en/>

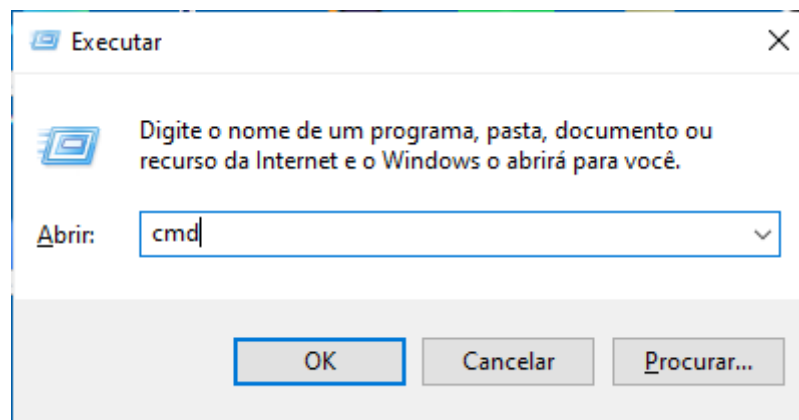


2. Faça o download da versão 14 do Node.js e instale no seu computador. Em caso de dúvidas, consulte o Guia de instalação do Node.js.

Passo 07 - Instalação do Heroku Client

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
```

```
C:\Users\rafae\Documents\workspace-spring-tool-suite-4-4.11.0.RELEASE\blogpessoal>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
```

```
C:\Users\rafae\Documents\workspace-spring-tool-suite-4-4.11.0.RELEASE\blogpessoal>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:

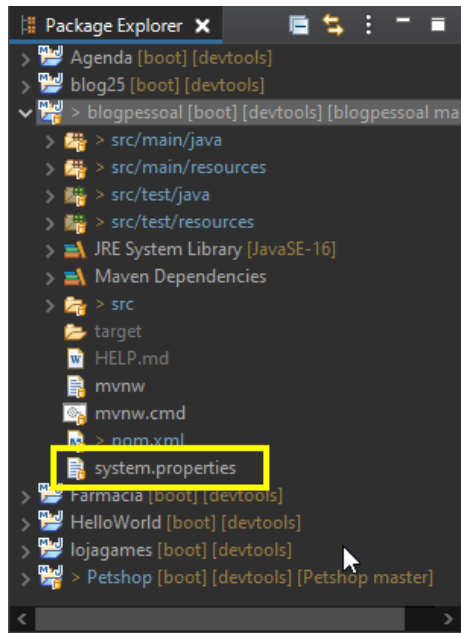
```
heroku version
```

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

***A versão pode ser diferente da imagem**

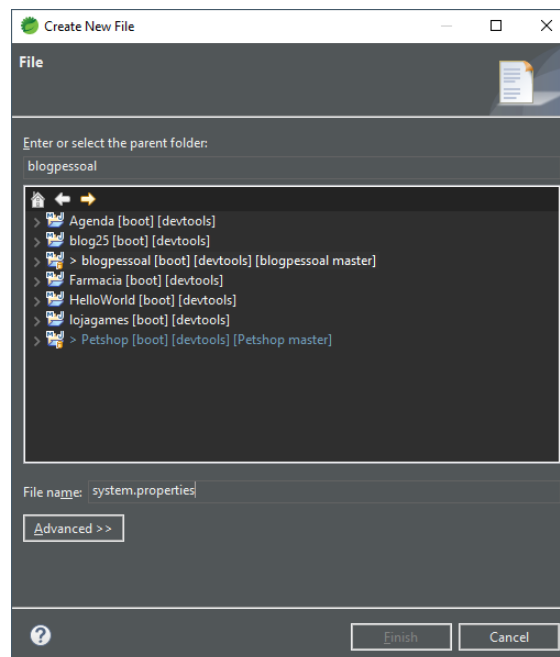
Passo 08 - Criação do arquivo system.properties

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 pela API no Heroku através da linha abaixo:

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

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

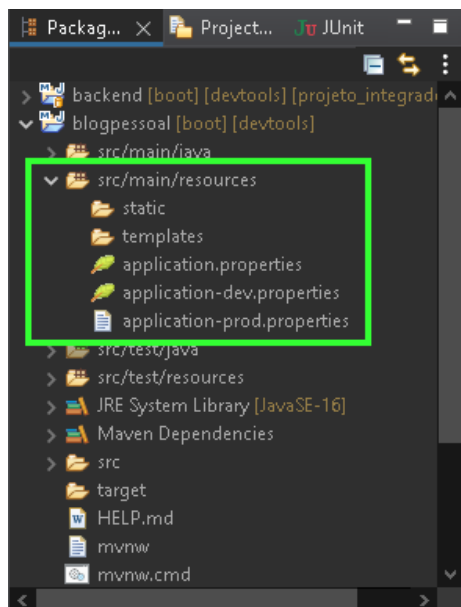
No arquivo, **pom.xml**, vamos adicionar as linhas abaixo, com a dependência do **PostgreSQL**:

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

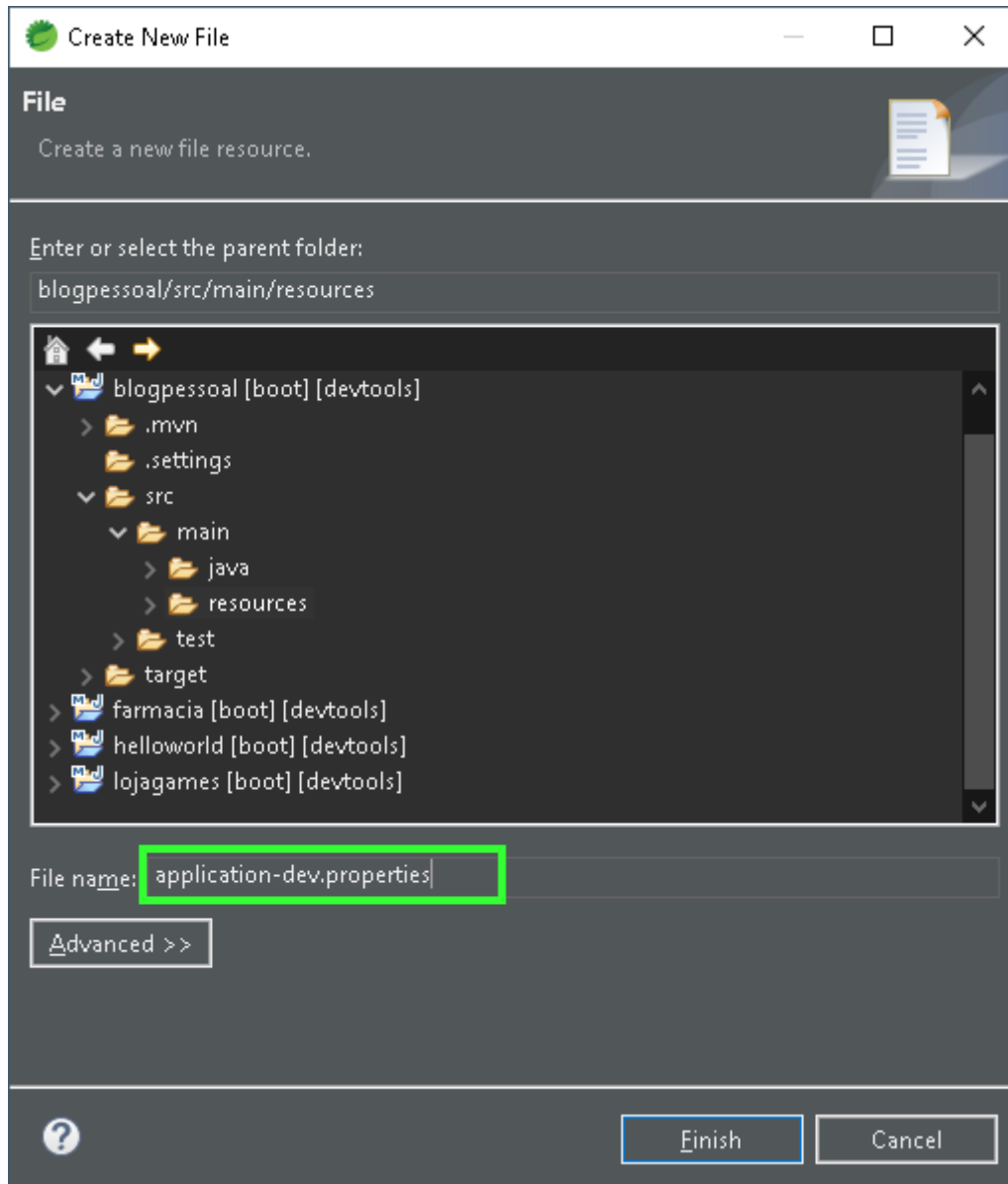
👉 Passo 10 - Configuração do Banco de Dados no arquivo application.properties

A Configuração do Banco de dados Local é diferente da configuração que será utilizada no Heroku. Para simplificar o processo, vamos utilizar o conceito de profiles (perfis), desta forma será possível alternar entre as configurações Local (MySQL) e Remota (PostgreSQL) de forma simples e rápida.

1. Na Source Folder **src/main/resources**, crie os arquivos **application-dev.properties** (Configuração do Banco de dados local) e **application-prod.properties** (Configuração do Banco de dados na nuvem).



2. No lado esquerdo superior, na Guia **Package explorer**, na Source Folder **src/main/resources**, clique com o botão direito do mouse e clique na opção **New** → **File**.
3. Em **File name**, digite o nome do primeiro arquivo (**application-dev.properties**) e clique no botão **Finish**.



4. Repita os itens 2 e 3 para criar o segundo arquivo **application-prod.properties**.

Agora vamos configurar os 3 arquivos:

1. Abra o arquivo **application.properties**, selecione todo o seu conteúdo e Recorte (**Ctrl + X**).
2. Abra o arquivo **application-dev.properties**, Cole (**Ctrl + V**) o conteúdo recortado do arquivo **application.properties** e salve o arquivo.

3. O arquivo **application-dev.properties** ficará com a configuração semelhante a esta:

```
spring.jpa.hibernate.ddl-auto=update
spring.jpa.database=mysql
spring.datasource.url=jdbc:mysql://localhost/db_blogpessoal?createDatabaseIfNotExist=true
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
```

4. No arquivo, **application.properties**, vamos inserir o conteúdo abaixo:

```
spring.profiles.active=prod
```

5. Para alternar entre as configurações, utilizaremos as 2 opções abaixo:

spring.profiles.active=dev → O Spring executará a aplicação com a configuração do Banco de dados local (MySQL)

spring.profiles.active=prod → O Spring executará a aplicação com a configuração do Banco de dados na nuvem (Heroku)

Para o Deploy, devemos deixar a linha **spring.profiles.active** configurada com a opção **prod**.

6. No arquivo, **application-prod.properties**, vamos inserir o conteúdo **exatamente** como está abaixo:

```
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
```

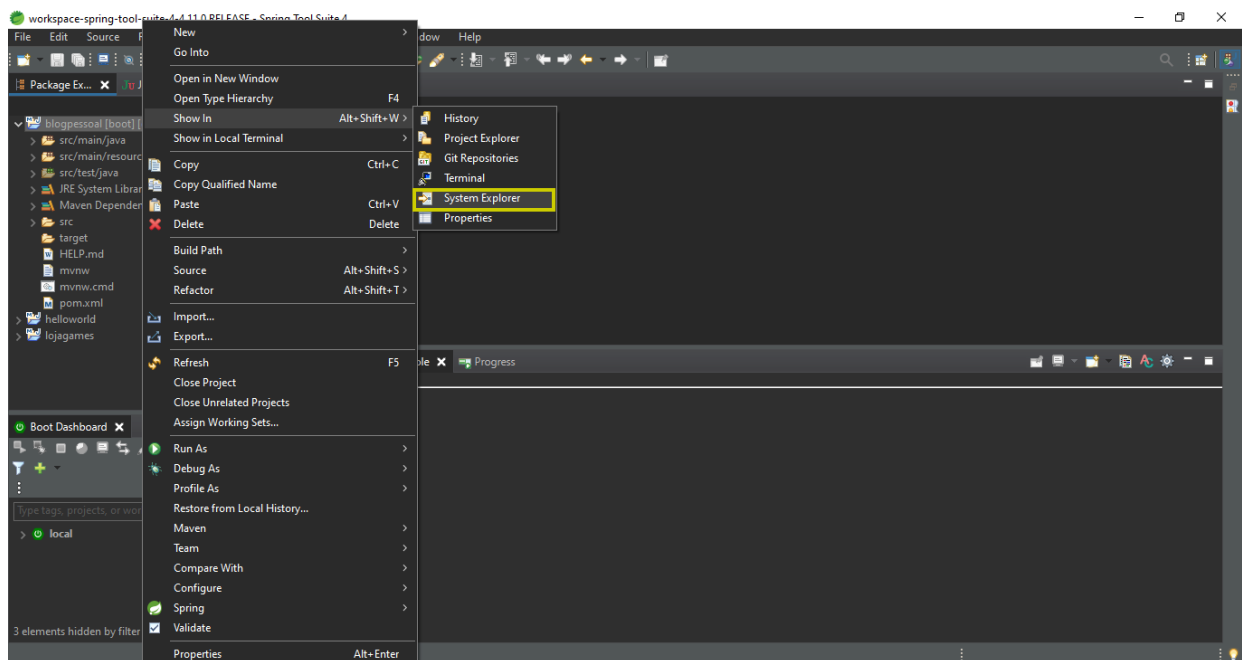
*** Importante ***

A partir deste ponto, **o seu projeto não executará mais localmente** (<http://localhost:8080/>). Para voltar a executar localmente, **abra o arquivo `application.properties` e altere a 1ª linha de `prod` para `dev`.**

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**



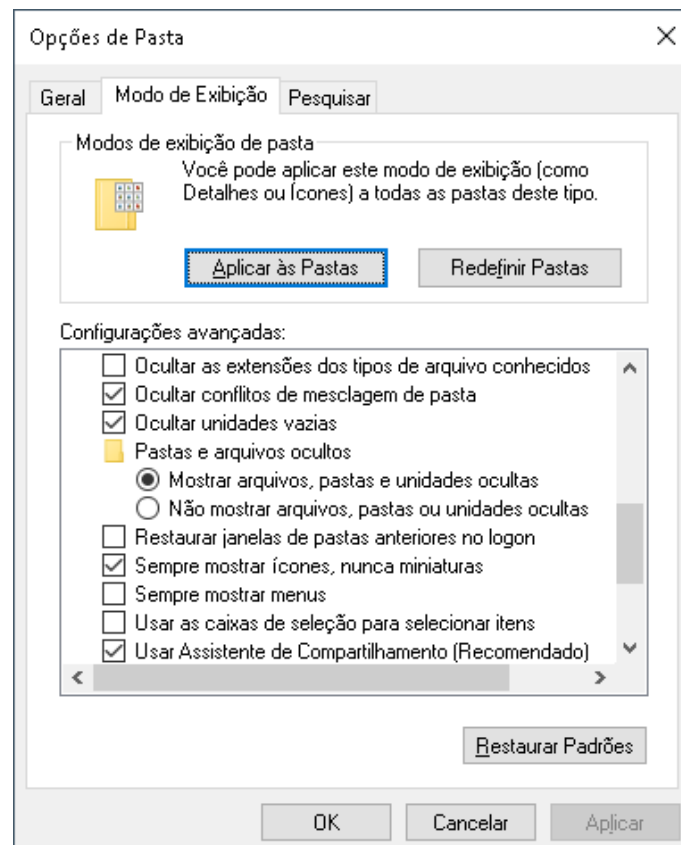
2. Será aberta a pasta Workspace onde o Eclipse/STS grava os seus projetos:
- Se você estiver usando o STS geralmente a pasta fica em: **c:\Usuarios\seu usuario\Documents\workspace-spring-tool-suite-4-4.11.0.RELEASE** (a versão pode ser diferente).

**seu usuario = Usuário do seu computador*

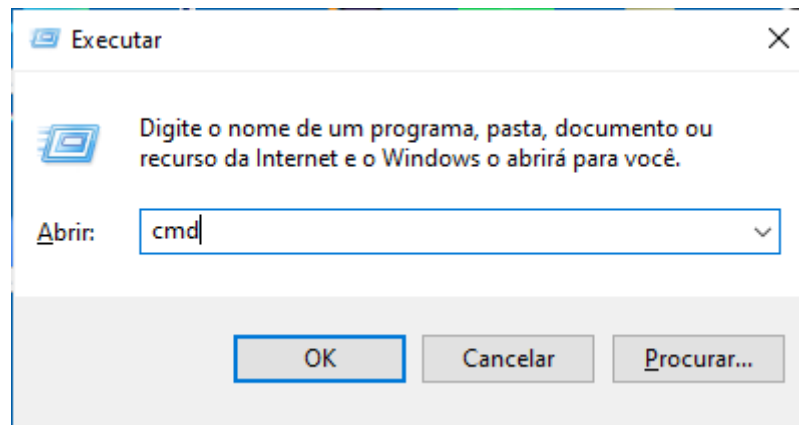
3. Abra esta pasta do projeto e verifique se existe uma pasta chamada **.git**. Caso ela exista, apague esta pasta. **Esta pasta estará presente APENAS se você inicializou o git dentro da pasta do projeto.**

Nome	Data de modificação	Tipo	Tamanho
.git	13/07/2021 16:50	Pasta de arquivos	
.mvn	13/07/2021 07:39	Pasta de arquivos	
.settings	13/07/2021 07:39	Pasta de arquivos	
.vscode	13/07/2021 07:39	Pasta de arquivos	
src	13/07/2021 07:39	Pasta de arquivos	
target	13/07/2021 07:39	Pasta de arquivos	
.classpath	13/07/2021 22:27	Arquivo CLASSPA...	3 KB
.gitignore	25/05/2021 22:47	Documento de Te...	1 KB
.project	27/05/2021 18:05	Arquivo PROJECT	1 KB
HELP.md	25/05/2021 22:47	Markdown File	2 KB
mvnw	25/05/2021 22:47	Arquivo	10 KB
mvnw.cmd	25/05/2021 22:47	Script de Comand...	7 KB
pom.xml	13/07/2021 22:27	Documento XML	3 KB
system.properties	13/07/2021 22:26	Arquivo PROPERTI...	1 KB

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.



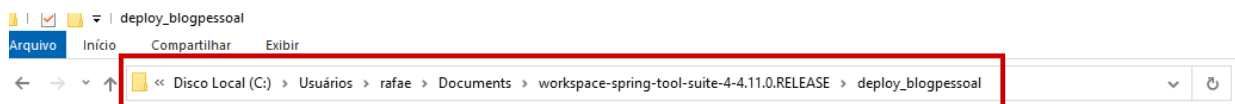
4. Execute o atalho  +  para abrir a janela Executar



5. Digite o comando abaixo para abrir o **Prompt de Comando do Windows**:

`cmd`

6. Na pasta do seu projeto, no **Windows Explorer**, copie o caminho da pasta conforme a figura abaixo:



7. No Prompt de comando do Windows digite o 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*

8. 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"
```

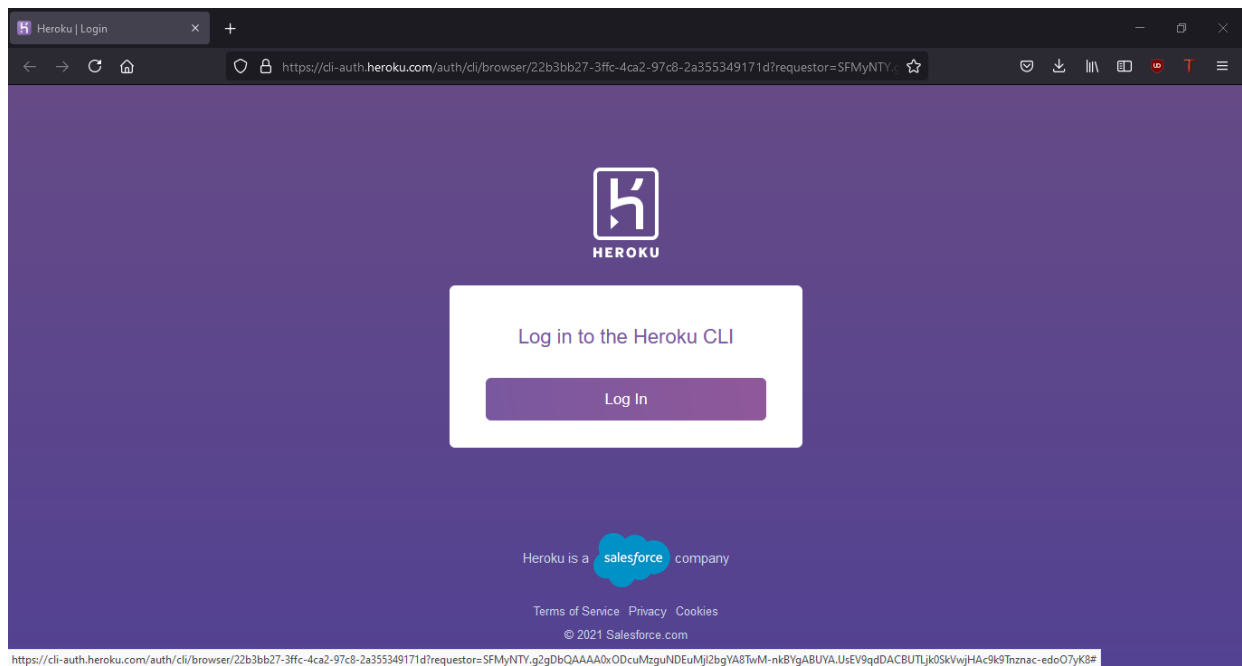
Passo 12 - Login no Heroku

1. Digite o comando:

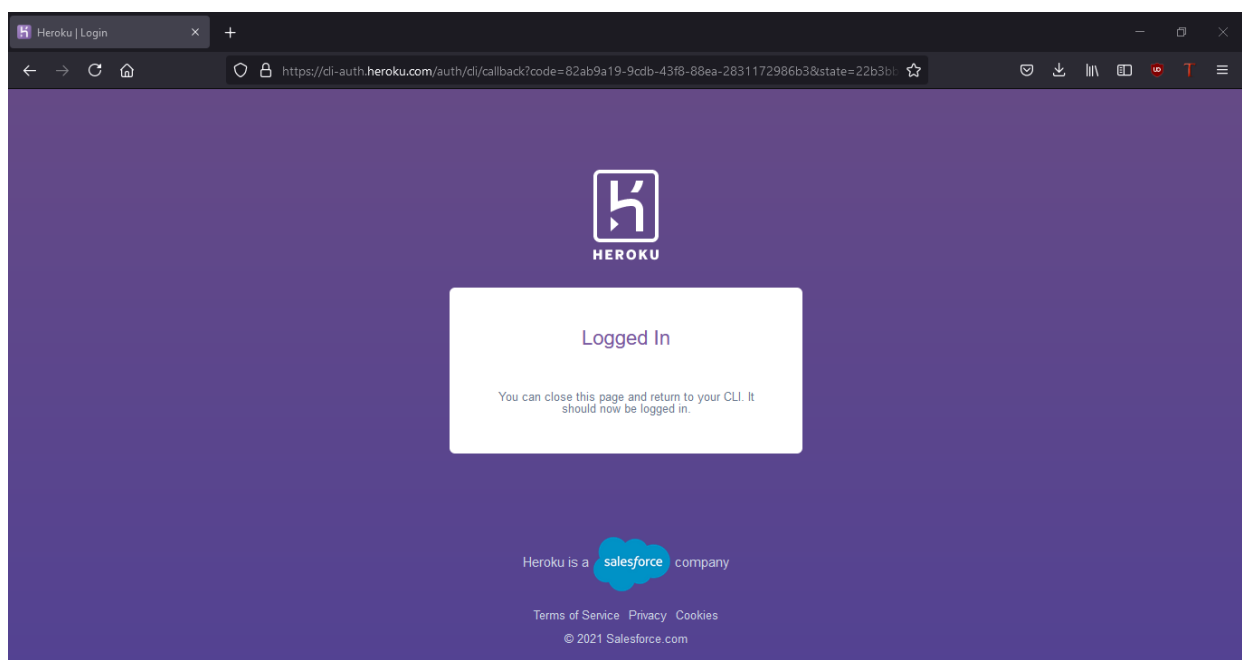
```
heroku login
```

```
>> Warning: Our terms of service have changed: https://dashboard.heroku.com/terms-of-service
heroku: Press any key to open up the browser to login or q to exit:
```

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.

```
>> Warning: Our terms of service have changed: https://dashboard.heroku.com/terms-of-service
heroku: Press any key to open up the browser to login or q to exit:
Opening browser to https://cli-auth.heroku.com/auth/cli/browser/22b3bb27-3ffc-4ca2-97c8-2a355349171d?requestor=SFMyNTY.g
2gDbQAAAA0xODcuMzguNDEuMjI2bgYA8TwM-nkBYgABUYA.USeV9qdDACBUTLjk0SkVwJHAc9k9Tnznac-edo07yK8
Logging in... done
Logged in as rafaelpinfo@gmail.com
```



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 existir, será exibida a mensagem abaixo:

```
Creating ☐ bloggen... !
! Name bloggen is already taken
```


Se o nome escolhido for aceito, será exibida a mensagem abaixo:

```
Creating ☐ bprfp... done
https://bprfp.herokuapp.com/ | https://git.heroku.com/bprfp.git
```

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:

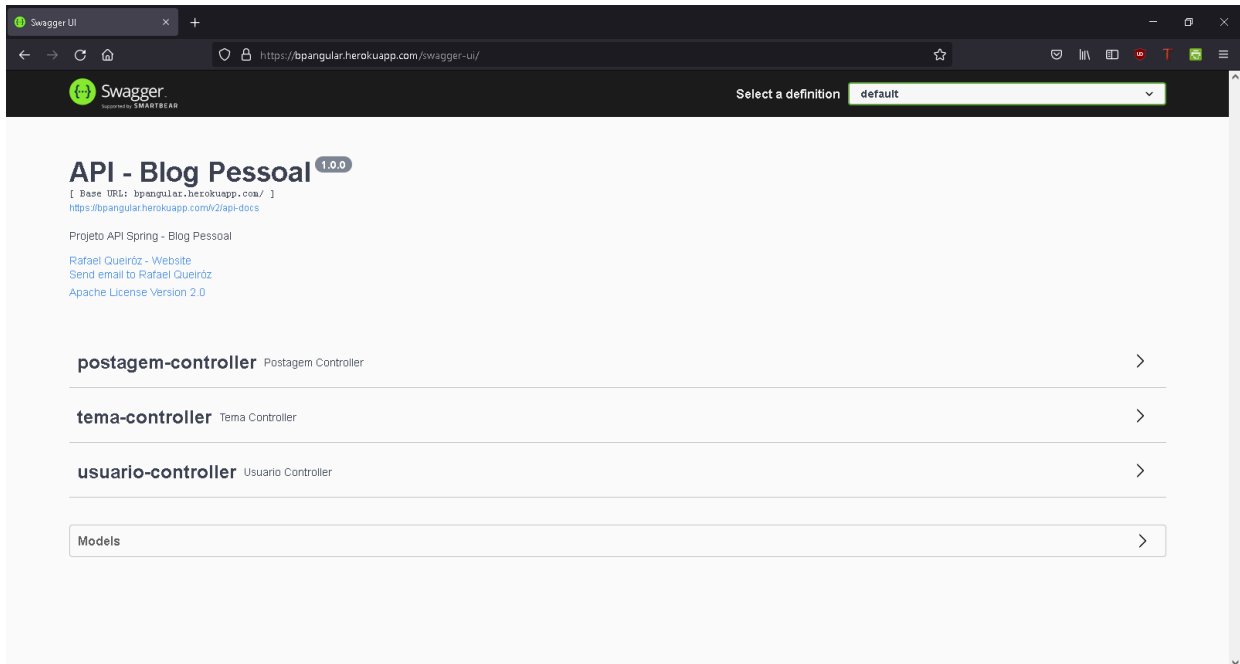
```
git push heroku master
```

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)

```
C:\Windows\system32\cmd.exe
-Utils\0.4/maven-shared-utils-0.4.jar (155 kB at 6.8 MB/s)
remote: [INFO] Downloaded from central: https://repo.maven.apache.org/maven2/com/google/code/findbugs/jsr305/2.0.1/jsr305-2.0.1.jar (32 kB at 1.3 MB/s)
remote: [INFO] Downloaded from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-utils/3.0.15/plexus-utils-3.0.15.jar (239 kB at 7.0 MB/s)
remote: [INFO] Installing /tmp/build_976cda76/target/blogpessoal-0.0.1-SNAPSHOT.jar to /tmp/codon/tmp/cache/.m2/repository/br/org/generation/blogpessoal/0.0.1-SNAPSHOT/blogpessoal-0.0.1-SNAPSHOT.jar
remote: [INFO] Installing /tmp/build_976cda76/pom.xml to /tmp/codon/tmp/cache/.m2/repository/br/org/generation/blogpessoal/0.0.1-SNAPSHOT/blogpessoal-0.0.1-SNAPSHOT.pom
remote: [INFO] -----
remote: [INFO] BUILD SUCCESS
remote: [INFO] -----
remote: [INFO] Total time: 20.827 s
remote: [INFO] Finished at: 2021-06-11T07:59:26Z
remote: [INFO] -----
remote: -----> Discovering process types
remote: Procfile declares types -> (none)
remote: Default types for buildpack -> web
remote: -----> Compressing...
remote: Done: 108.9M
remote: -----> Launching...
remote: Released v5
remote: https://bprfp.herokuapp.com/ deployed to Heroku
remote: Verifying deploy... done.
To https://git.heroku.com/bprfp.git
* [new branch] master -> master
```


Passo 16 - Testar o link e a API

1. Abra o navegador e digite o endereço a sua API
2. Será solicitado o usuário e a senha. Digite **root** para ambos.
3. Sua API abrirá o Swagger.



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

Atualizar o Deploy no Heroku

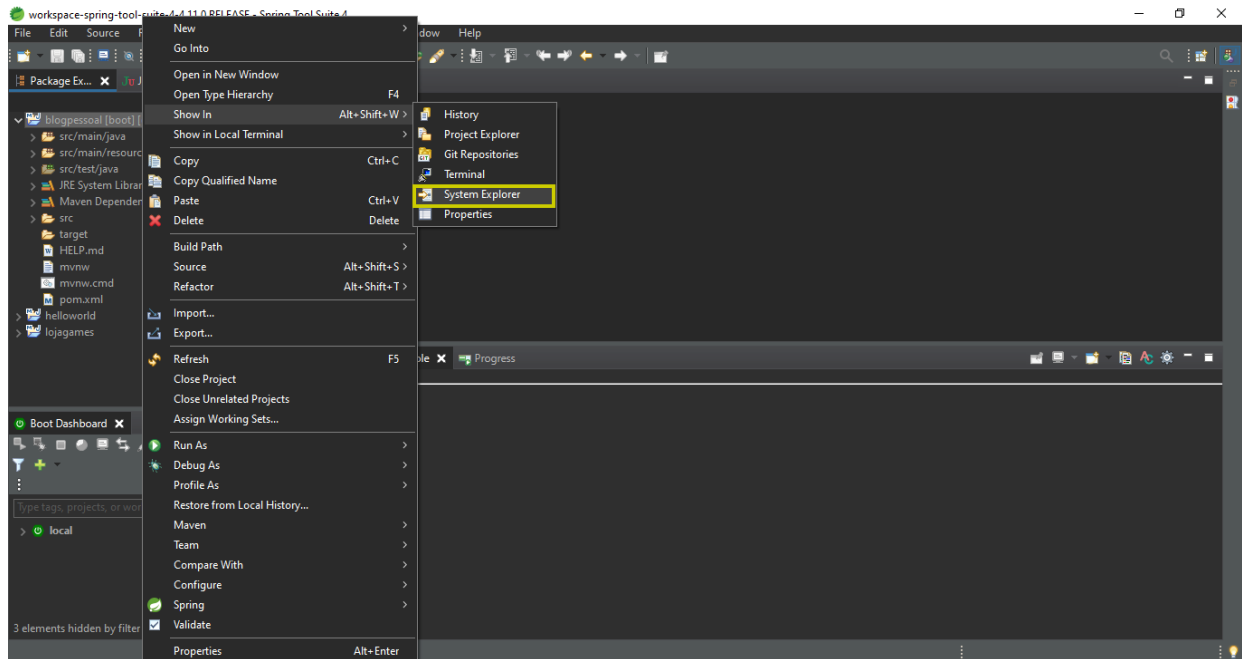
1. Para fazer alterações no código do projeto e executar localmente, altere o arquivo, **application.properties** conforme o código abaixo:

```
spring.profiles.active=dev
```

2. Faça as alterações necessárias no seu projeto, execute localmente e verifique se está tudo funcionando.
3. Antes de refazer o Deploy, altere o arquivo, **application.properties** conforme o código abaixo:

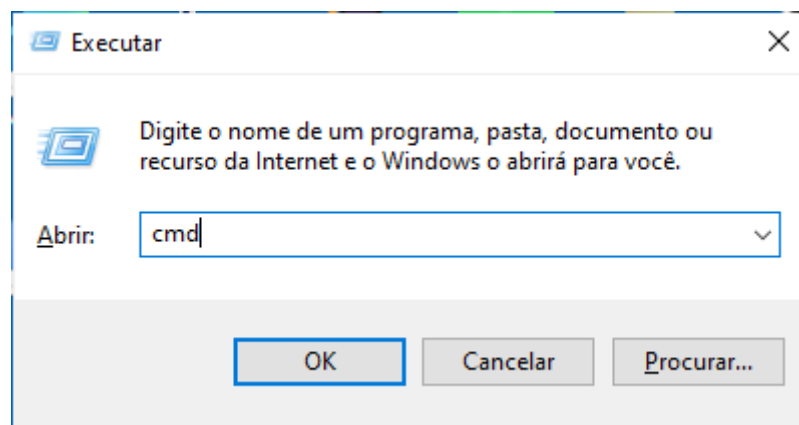
```
spring.profiles.active=prod
```

4. Na pasta do projeto, clique com o botão direito do mouse e na sequência clique na opção: **Show in → System Explorer**



5. Será aberta a pasta Workspace onde o Eclipse/STS grava os seus projetos. Abra a pasta do projeto (**blogpessoal**)

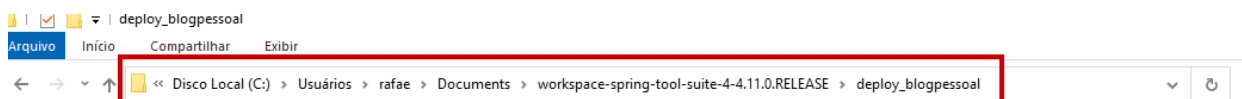
6. Execute o atalho  +  para abrir a janela Executar



7. Digite o comando abaixo para abrir o **Prompt de Comando do Windows**:

cmd

8. Na pasta do seu projeto, no Windows explorer, copie o caminho da pasta conforme a figura abaixo:



9. No Prompt de comando do Windows digite o 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*

10. Atualize o Deploy utilizando a sequência de comandos abaixo:

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

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

```
git remote
```

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

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

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

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

14. Para atualizar o Deploy, utilize os comandos baixo:

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

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

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

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