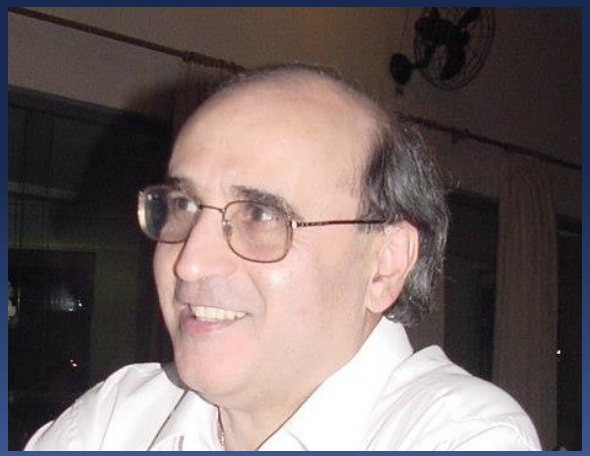
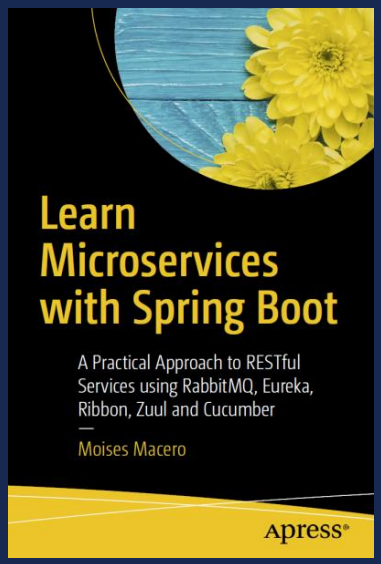
Spring Boot

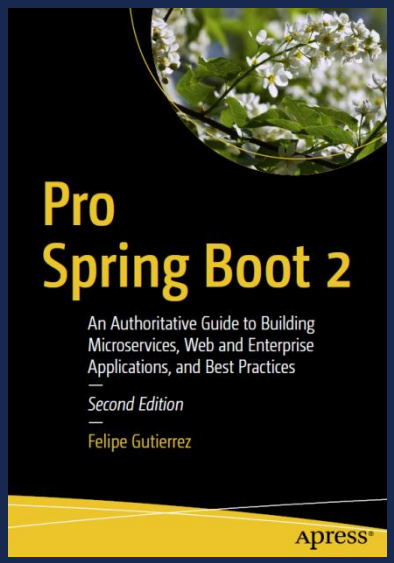
Unidade 1 - Introdução

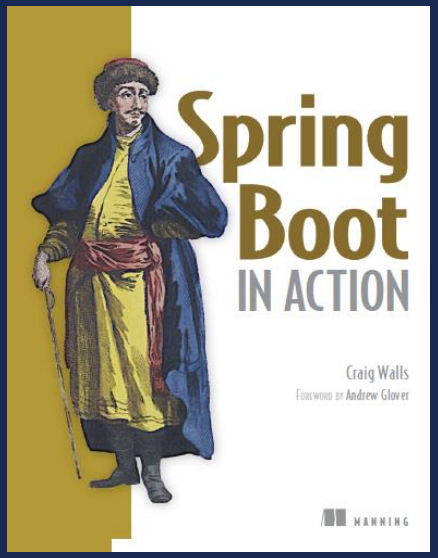


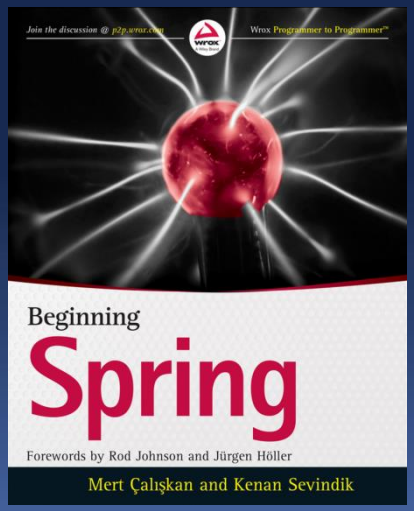
Prof. Aparecido V. de Freitas
Doutor em Engenharia
da Computação pela EPUSP
aparecidovfreitas@gmail.com

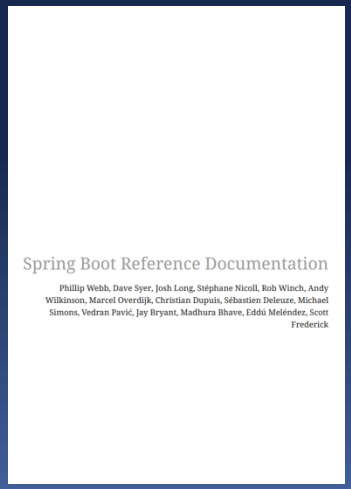
Bibliografia











Introdução

- O framework **Spring** foi desenvolvido por volta dos anos 2000 e continua sendo largamente utilizado até os dias de hoje em aplicações Java;
- O grande objetivo do **Spring**, desde sua concepção, é simplificar o desenvolvimento de aplicações **JEE**;
- Desenvolvido por Rod Johnson, a partir do código escrito inicialmente no livro "J2EE Design and Development", tendo recebido -- após a liberação -- a contribuição de desenvolvedores da comunidade;
- Spring foca na simplicidade de código com o uso de conceitos tais como inversão de controle e injeção de dependências;
- Spring corresponde à uma forma de complementar a arquitetura **JEE**.

Introdução

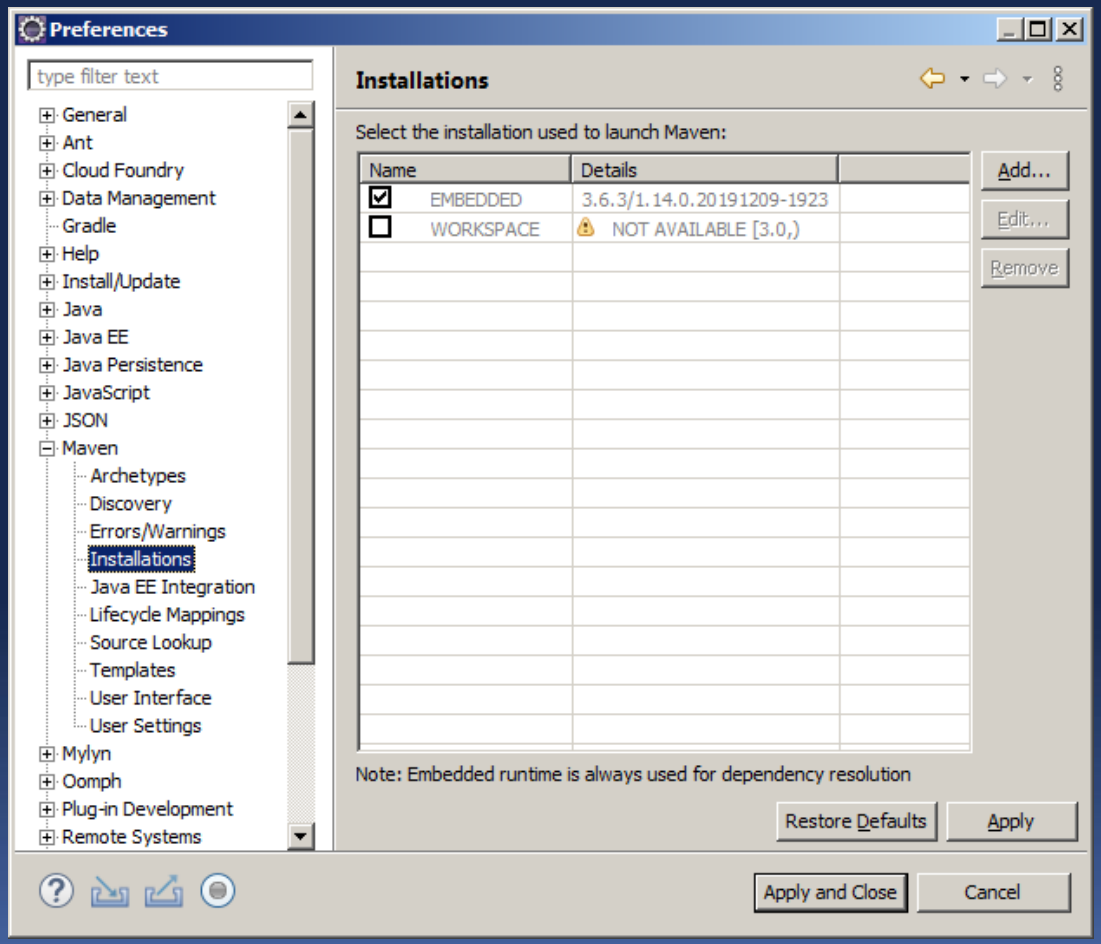
- Com **Spring Boot** pode-se criar **APIs REST**;
- O framework **Spring** disponibiliza uma forma simples e prática de se criar um projeto Rest;
- No endereço **start.spring.io** pode-se configurar o projeto a partir de algumas informações que são solicitadas;
- Spring Boot gera um Projeto Maven e, assim, configuraremos o Eclipse para que o Maven baixe as dependências num **Repositório Maven Local**.

Integração do Maven com Eclipse

- Para a integração do **Maven** com o **Eclipse** recomenda-se utilizar o plugin **M2E**;
- **M2E** é um plugin do **Eclipse** que faz a integração com o **Maven**;
- Existe também o **Maven Eclipse Plugin** que gera um projeto para o Eclipse. Este é um plugin do **Maven** que simplesmente gera os arquivos de configuração necessários para a IDE; Portanto, são plugins diferentes;
- A distribuição Eclipse for JEE Developers já vem com o plugin M2E e uma instalação interna do Maven;
- Caso você esteja com uma versão diferente do Eclipse, use o menu Help > Eclipse Marketplace ... , pesquise por M2E e instale o plugin;

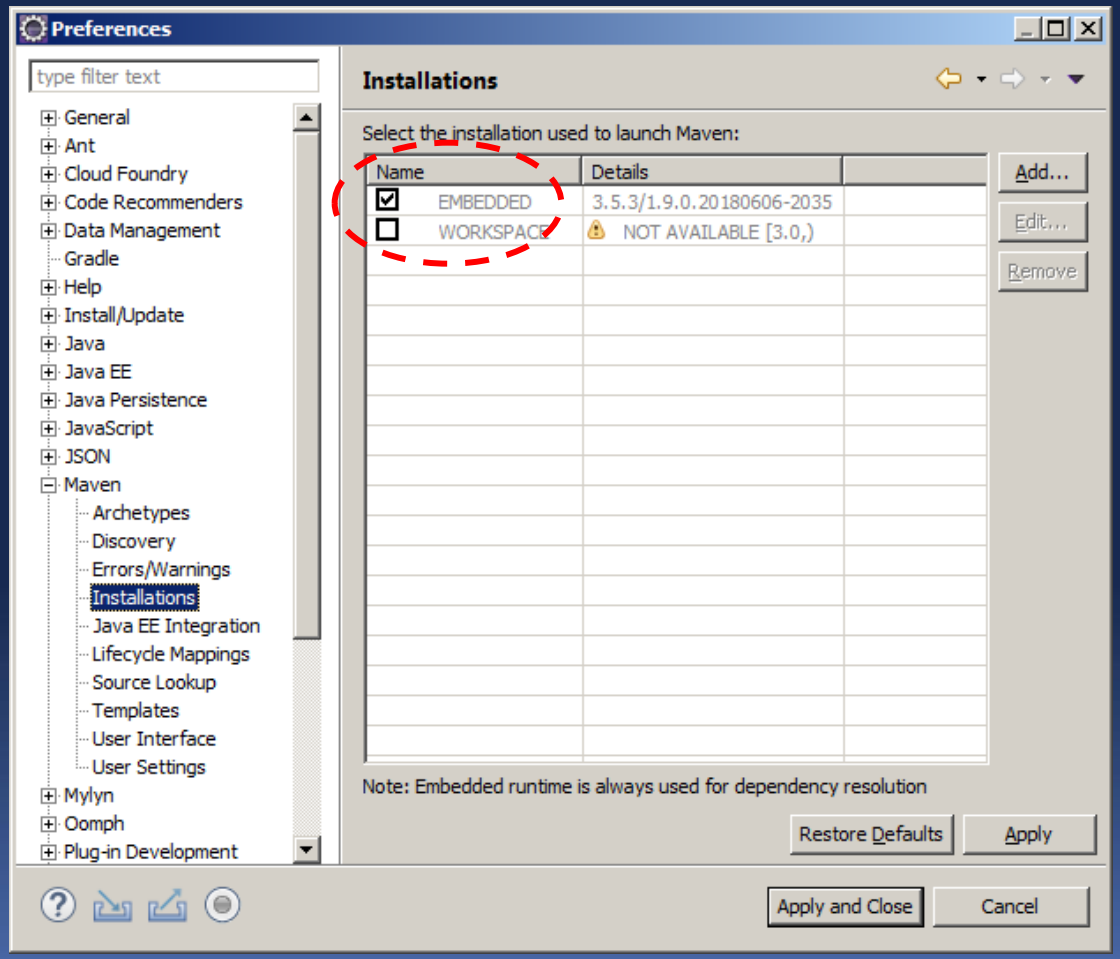
Integração do Maven com Eclipse

- ✓ Com o plugin **M2E** instalado e o Eclipse aberto, acesse o menu Window > Preferences;
- ✓ Vá para a opção **Windows> Preferences > Maven > Installations**.



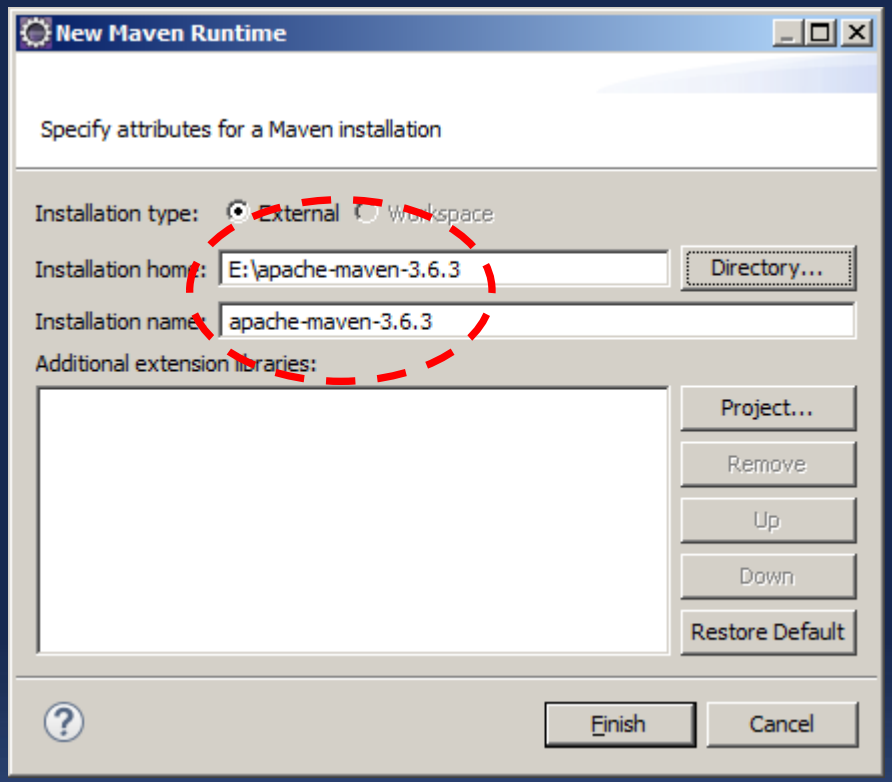
Integração do Maven com Eclipse

- ✓ Observe que já existe uma instalação "embarcada", mas com uma versão desatualizada;
- ✓ Vamos adicionar o nosso Maven recentemente instalado;
- ✓ Clique em **Add**...



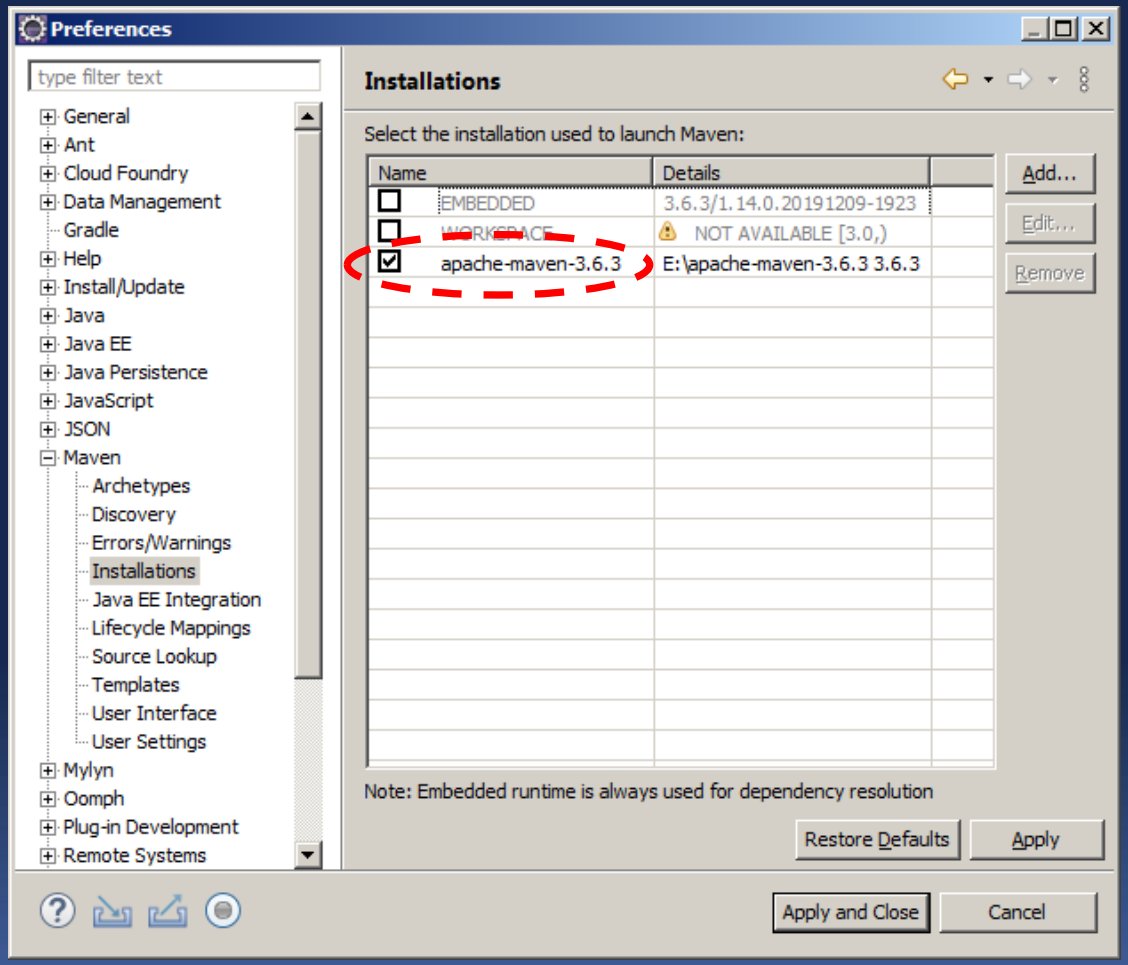
Integração do Maven com Eclipse

- ✓ Selecione a pasta com a nossa instalação do Maven e tecle **Finish**.



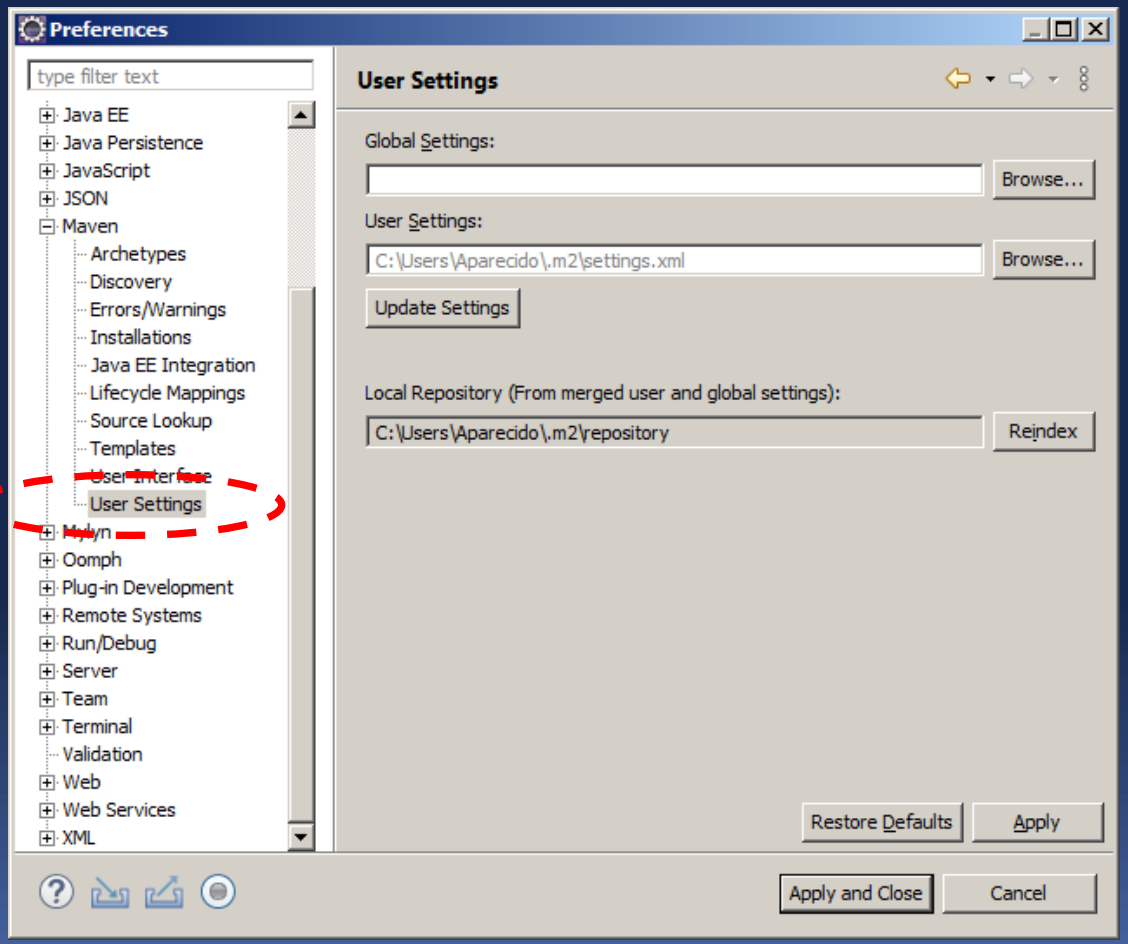
Integração do Maven com Eclipse

- ✓ Marque a nossa instalação e tecle **Apply and Close**.



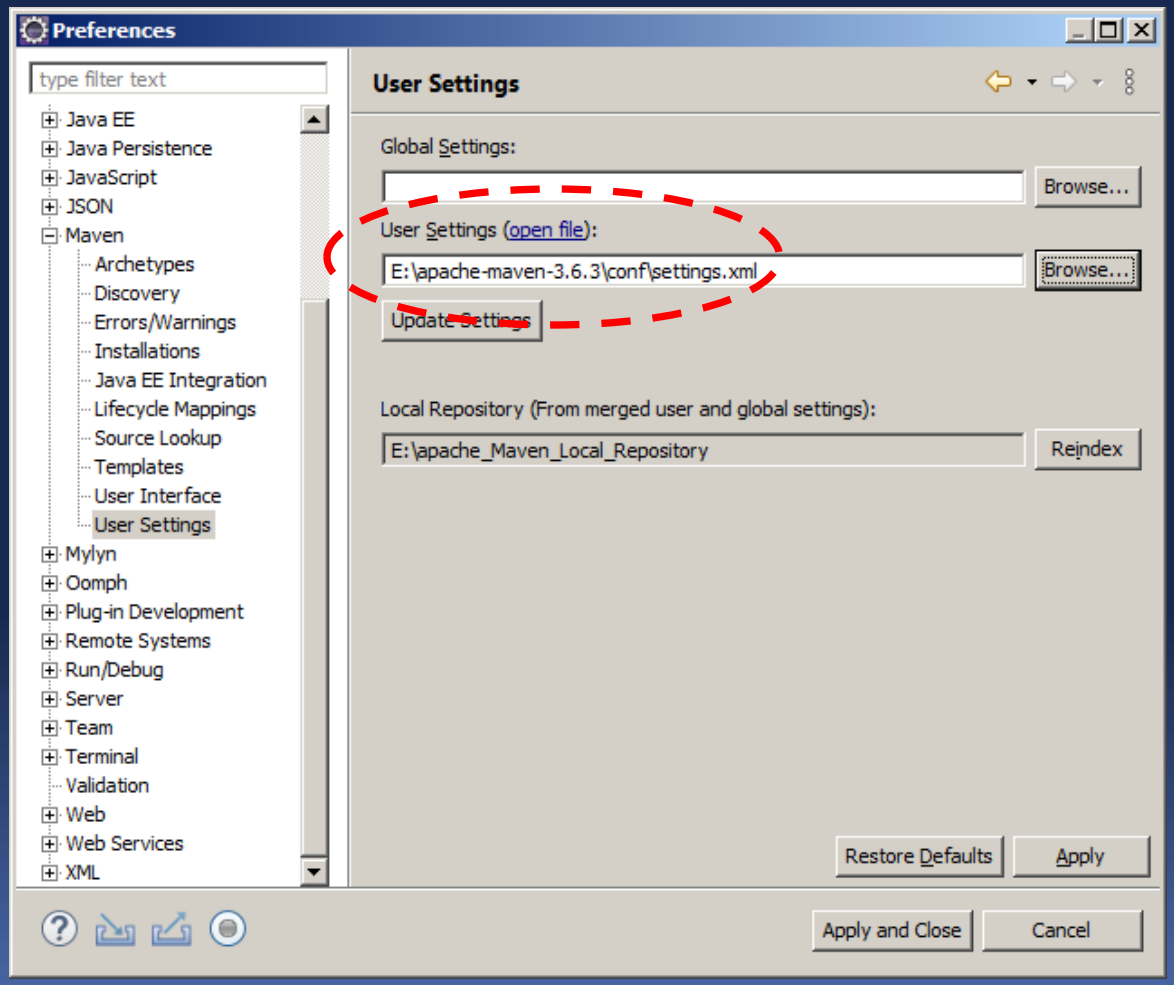
Integração do Maven com Eclipse

- ✓ Agora vamos atualizar o arquivo de configuração do **Maven**;
- ✓ Vá em Windows >Preferences > Maven > **User Settings** e defina o arquivo de configuração existente no diretório onde o **Maven** foi instalado.



Integração do Maven com Eclipse

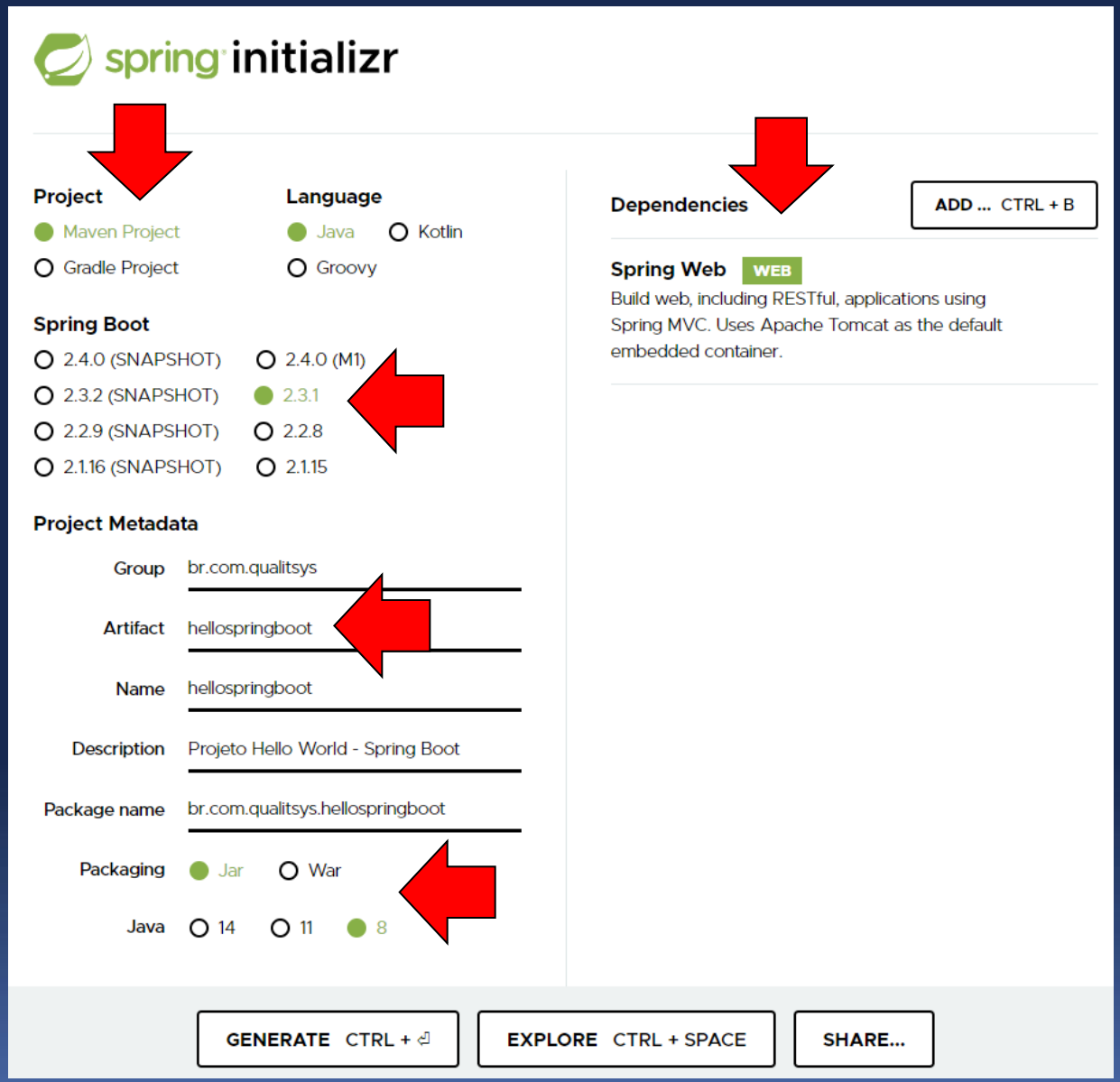
- ✓ Após a definição do arquivo de configuração da nossa instalação do Maven, tecle em **Update Settings**;
- ✓ Em seguida, tecle em Apply and Close.



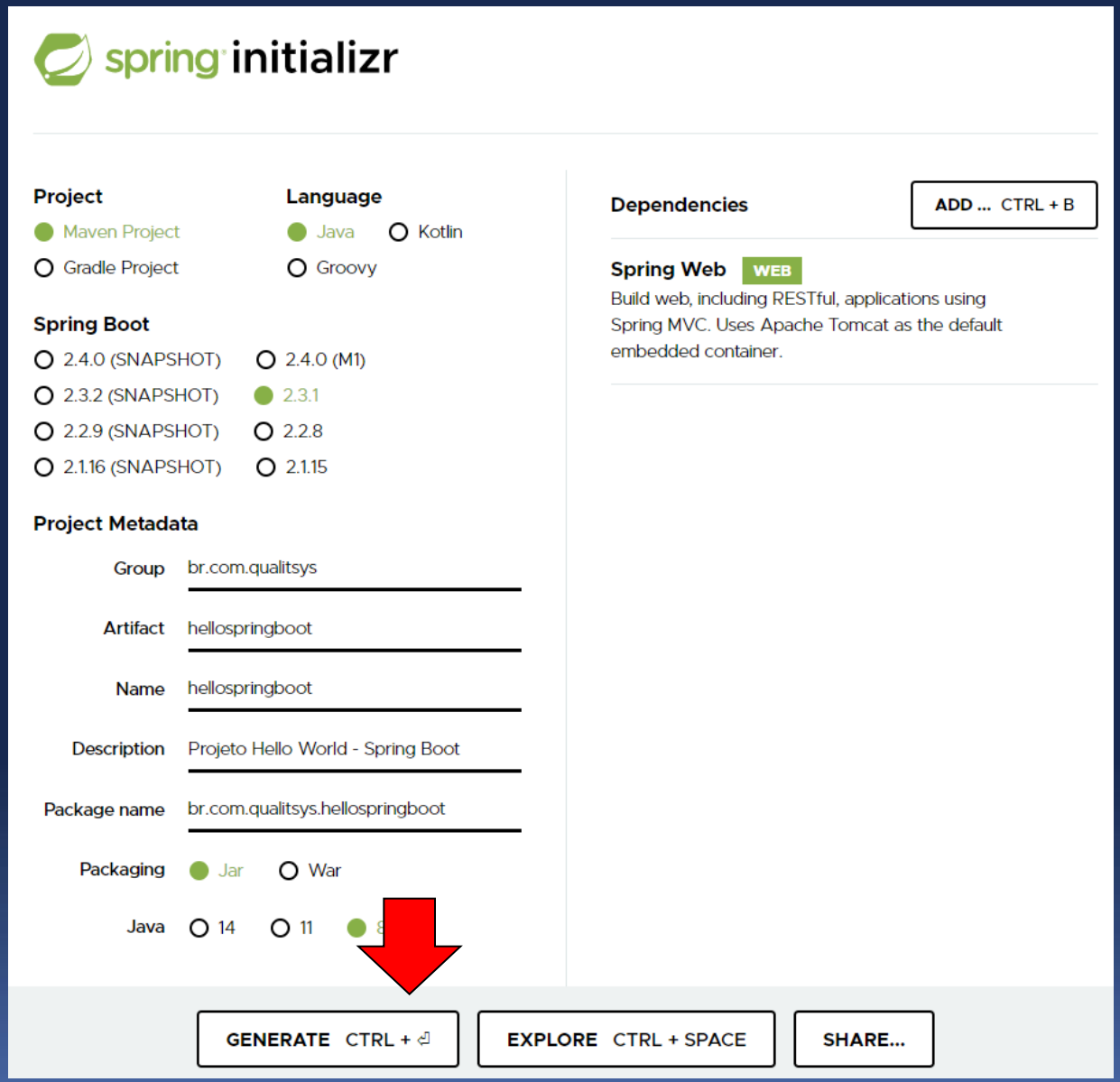
Pronto! Agora nosso Eclipse agora está configurado para que o Maven baixe os pacotes no nosso repositório local !

Agora vamos criar o projeto Maven com o Spring Boot !

http://start.spring.io



Generate Project

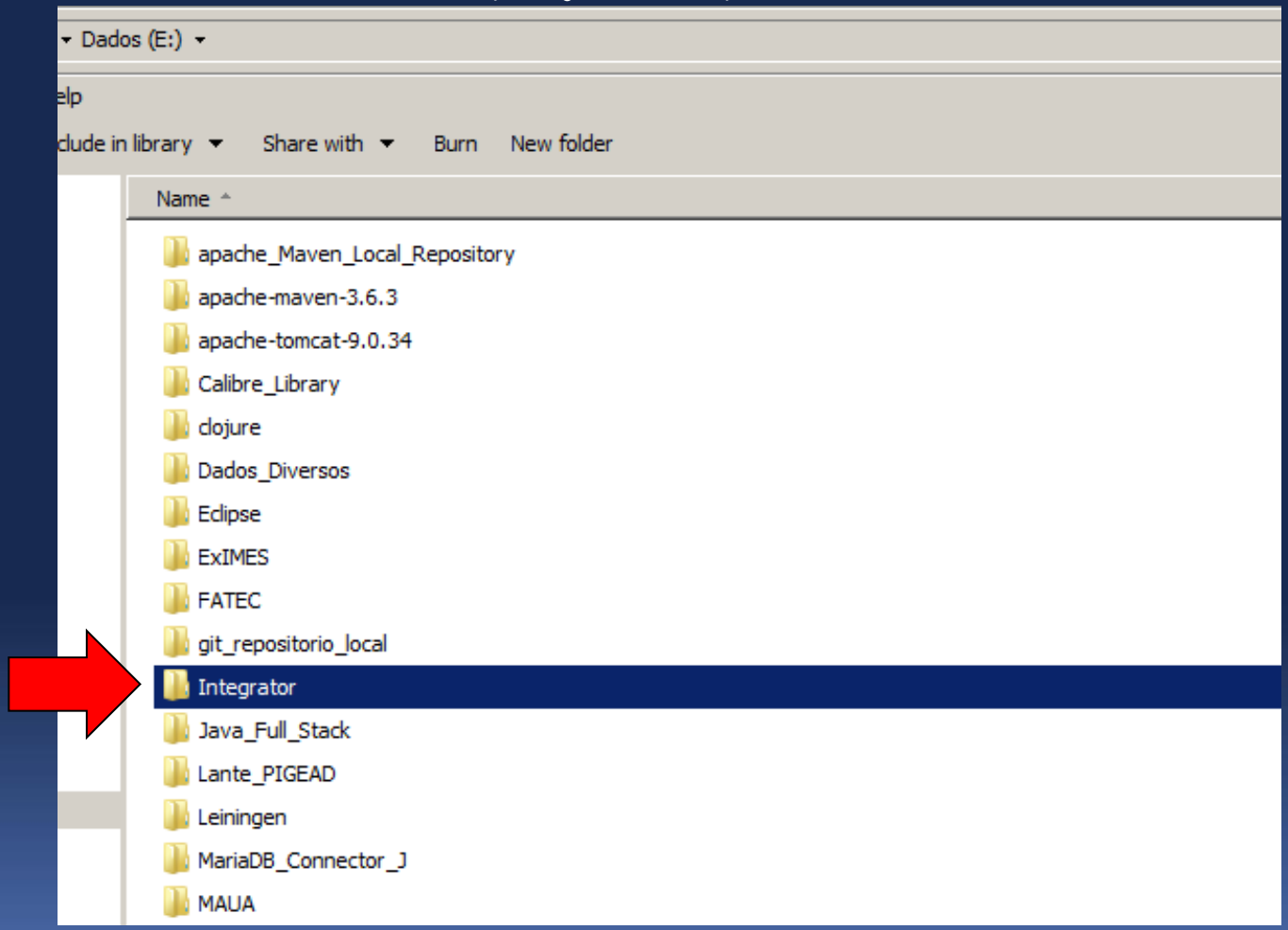


Generate Project

- Spring Boot gerou o projeto Maven no arquivo **hellospringboot.zip**

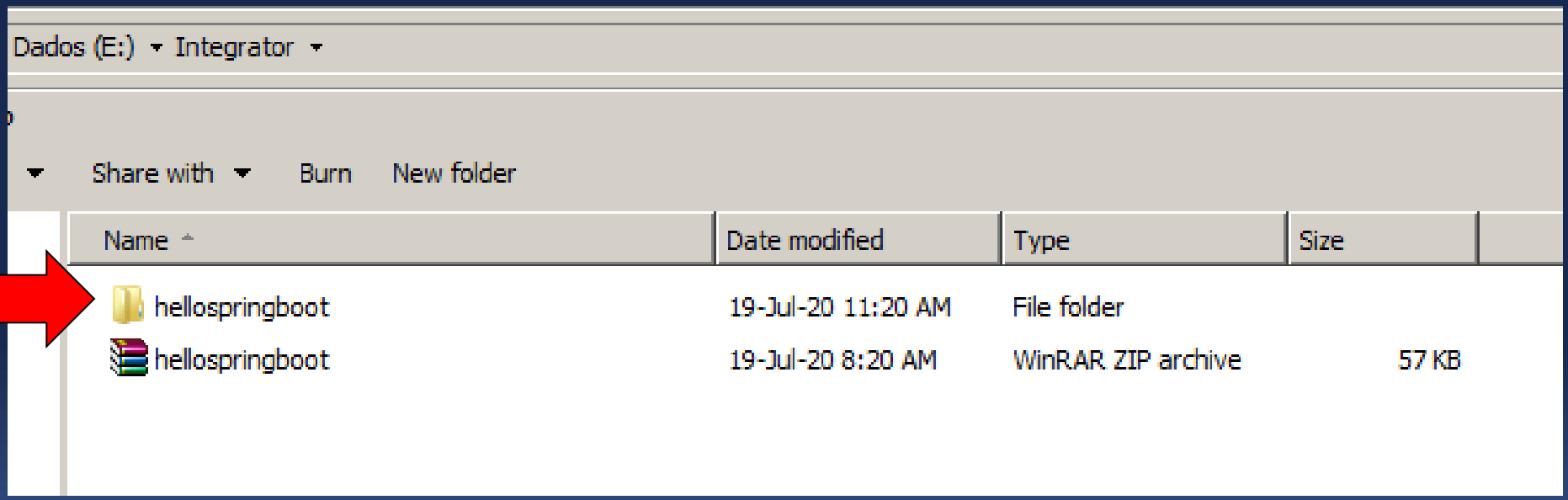
Initializr gera projeto em arquivo .zip

- Projeto Maven criado !
- Salvaremos o projeto na pasta E:\Integrator



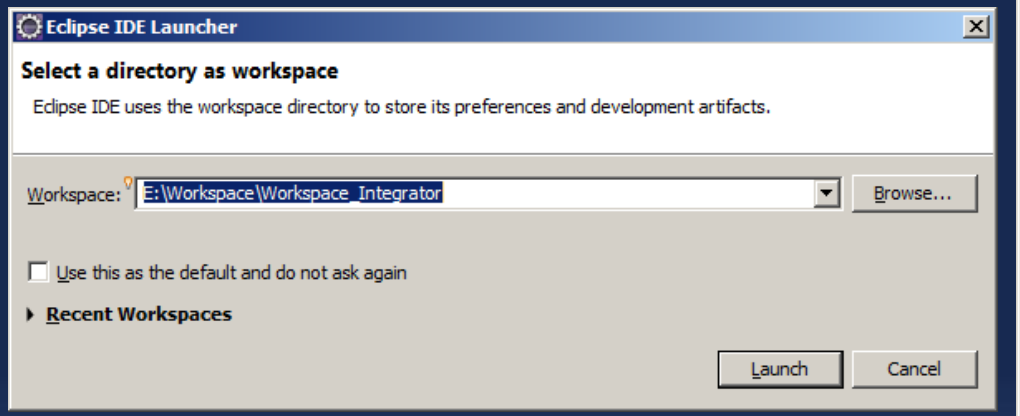
Initializr gera projeto em arquivo .zip

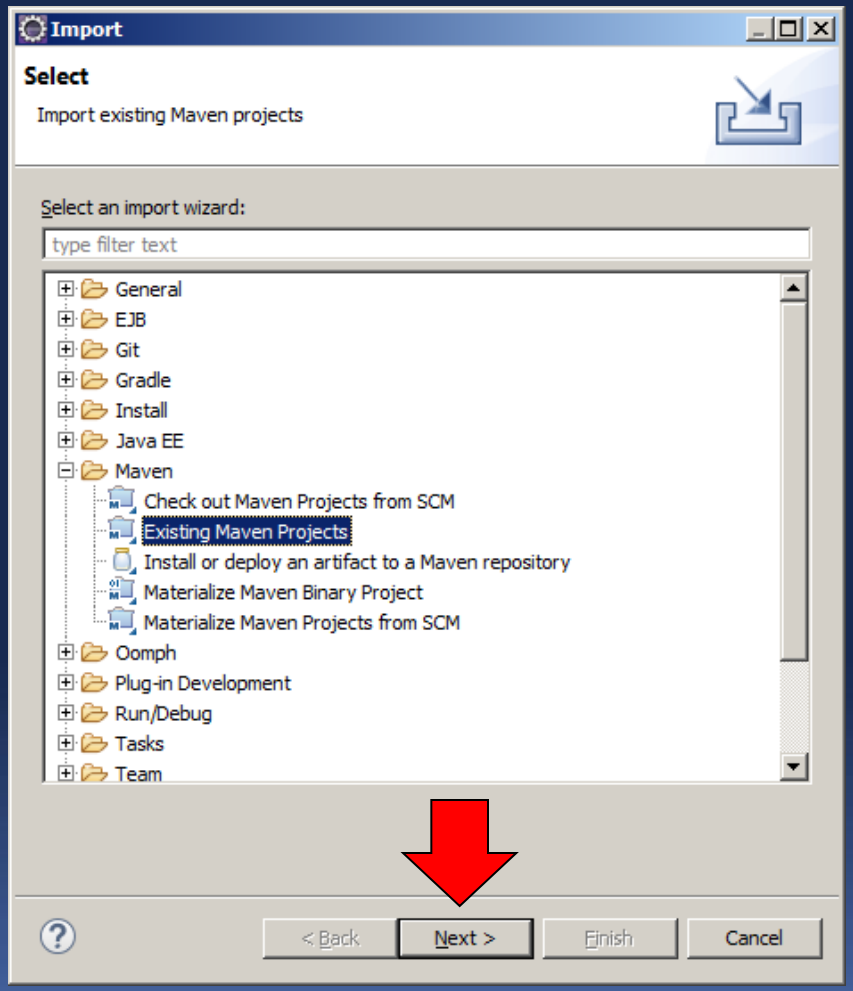
- Esse arquivo **.zip** é o projeto na estrutura do Maven;
- Descompactar o projeto em alguma pasta;
- Descompactaremos na pasta E:\Integrator



Abrindo o projeto no Eclipse

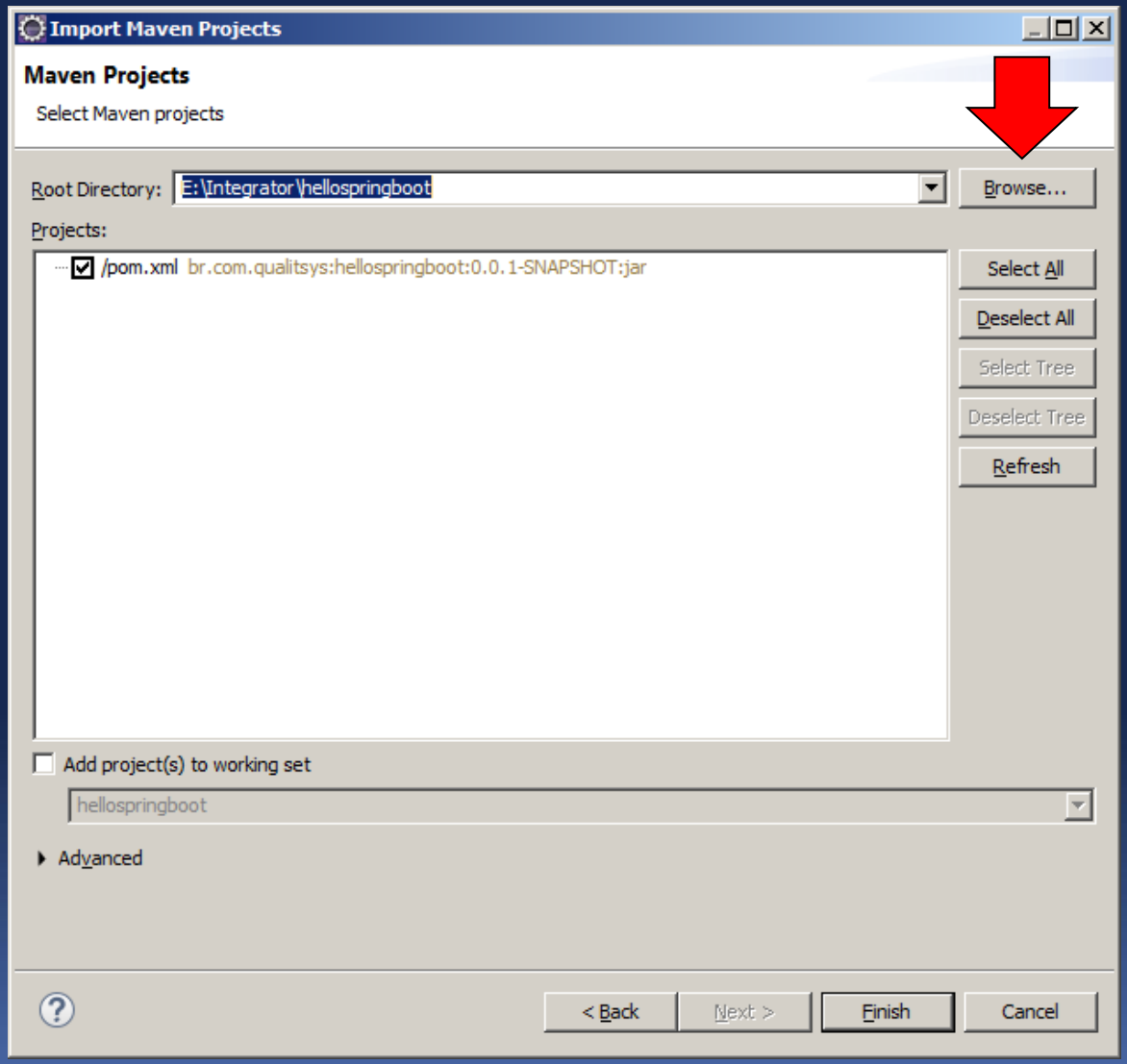
- Abriremos o Eclipse no **Workspace/Integrator**
- Import > Existing Maven Projects



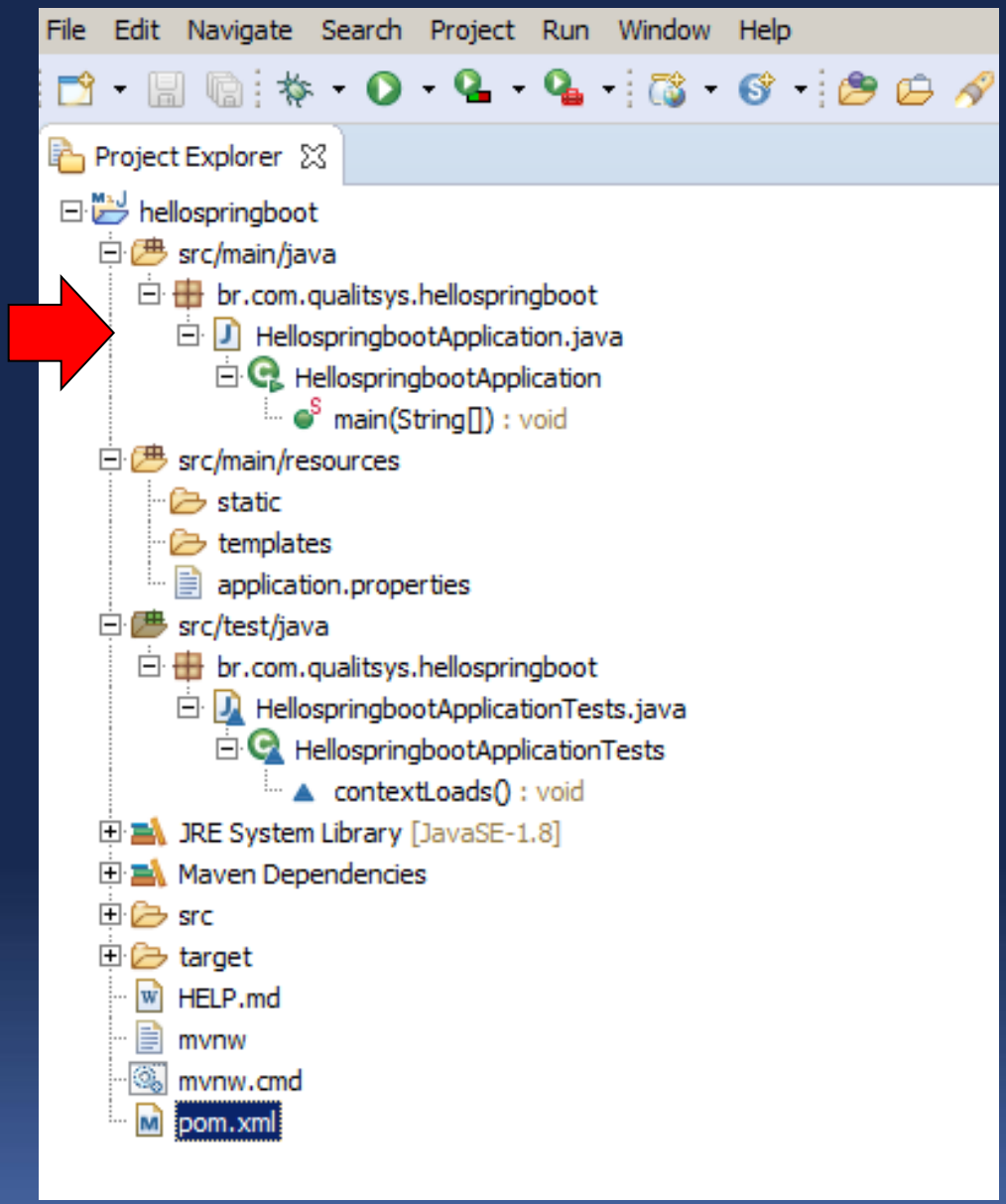


Abrindo o projeto no Eclipse

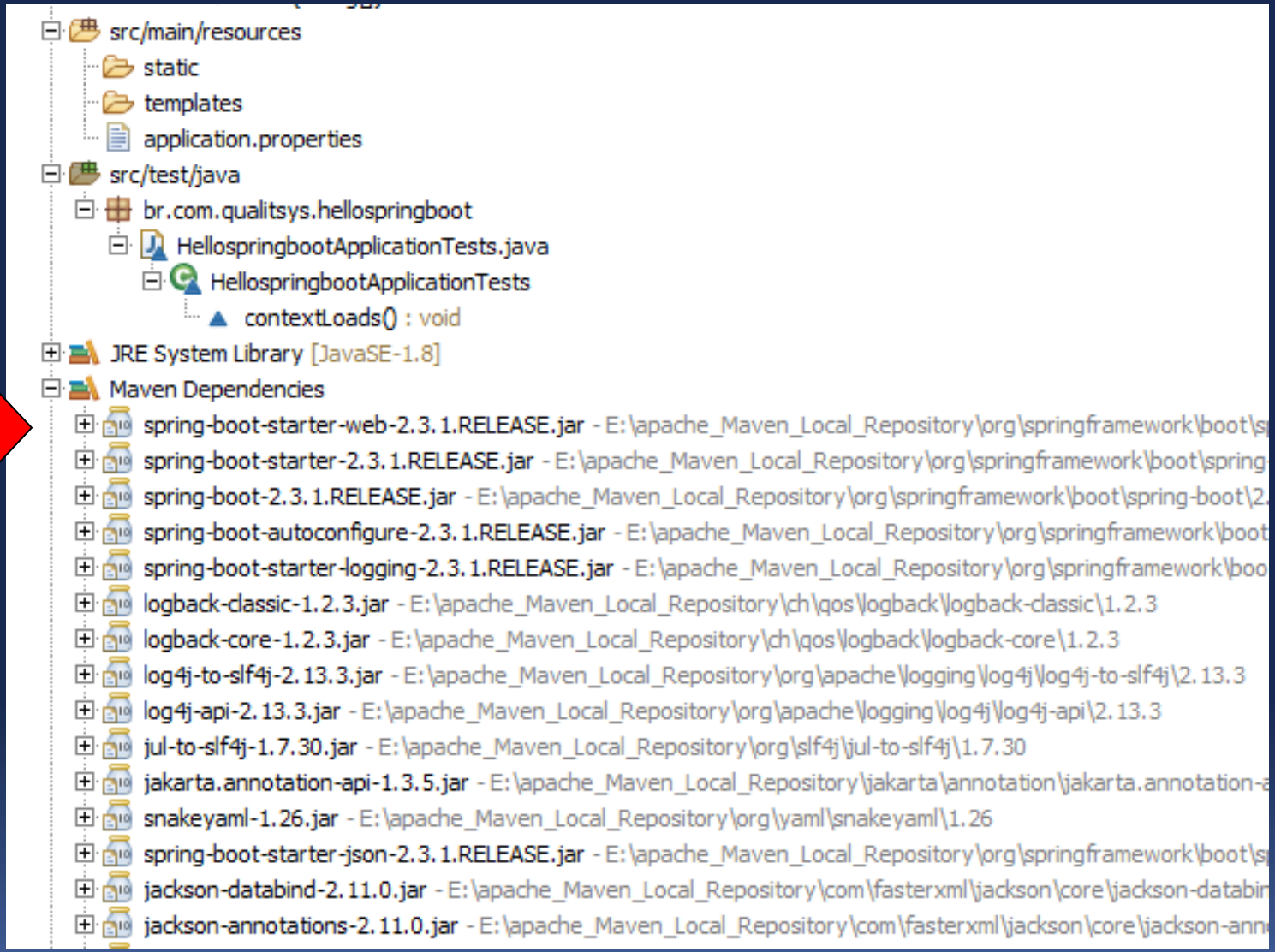
- Selecionar a pasta do projeto **Maven**.



Projeto criado no Eclipse



Módulos Spring importados



Arquivo pom.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.3.1.RELEASE</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>
    <groupId>br.com.qualitsys</groupId>
    <artifactId>hellospringboot</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>hellospringboot</name>
    <description>Projeto Helle World - Spring Boot</description>

    <properties>
        <java.version>1.8</java.version>
    </properties>
```

Arquivo pom.xml

```
<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>

    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-test</artifactId>
        <scope>test</scope>
        <exclusions>
            <exclusion>
                <groupId>org.junit.vintage</groupId>
                <artifactId>junit-vintage-engine</artifactId>
            </exclusion>
        </exclusions>
    </dependency>
</dependencies>
```

Arquivo pom.xml

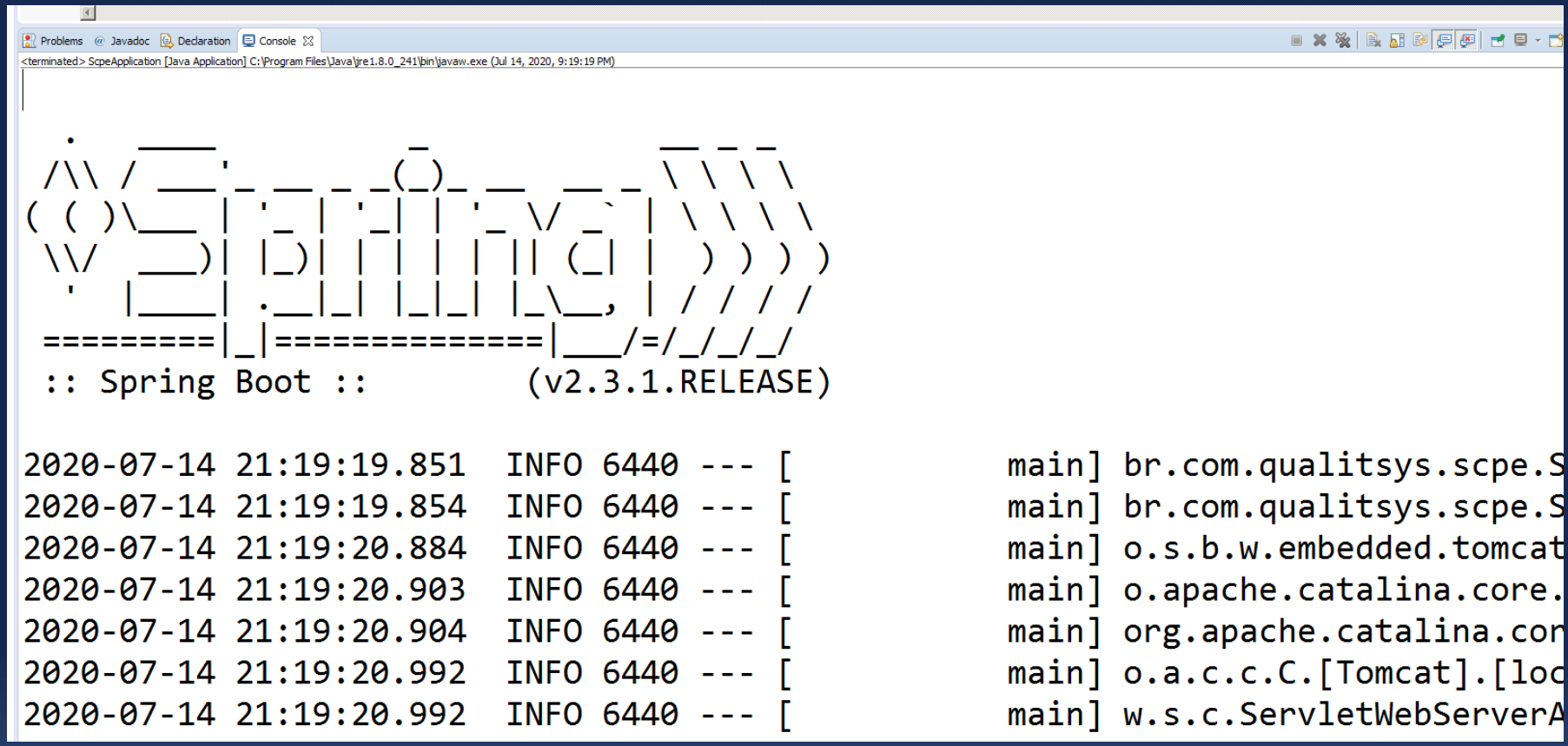
```
    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
        </plugins>
    </build>

</project>
```

Sprint criou uma classe....

```
package br.com.qualitsys.hellospringboot;

import org.springframework.boot.SpringApplication;

@SpringBootApplication
public class HellospringbootApplication {

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

}
```

- A classe criada foi **HelloSpringBootApplication** com o método **main**;
- O método **main** chama o método static **run** da classse **SpringApplication** e essa classe tem a anotação @SpringBootApplication;
- Essa classe é a classe de execução do projeto (**Tomcat** embarcado no **Spring Boot**).
- No **Spring Boot** essa é a classe principal. É a classe que contém o método main!

Executando a aplicação



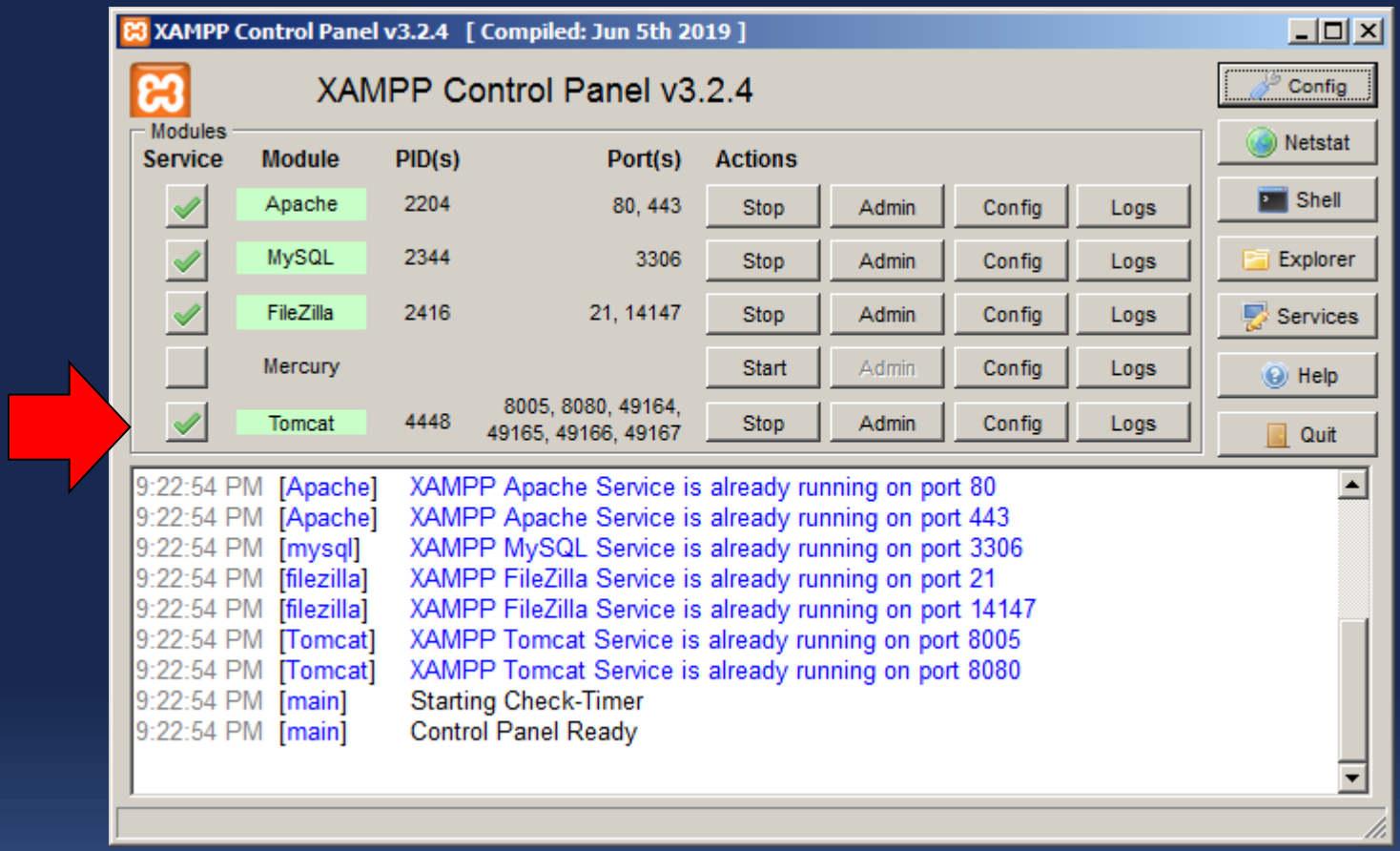
Executando a aplicação

Problems Javadoc Declaration Console

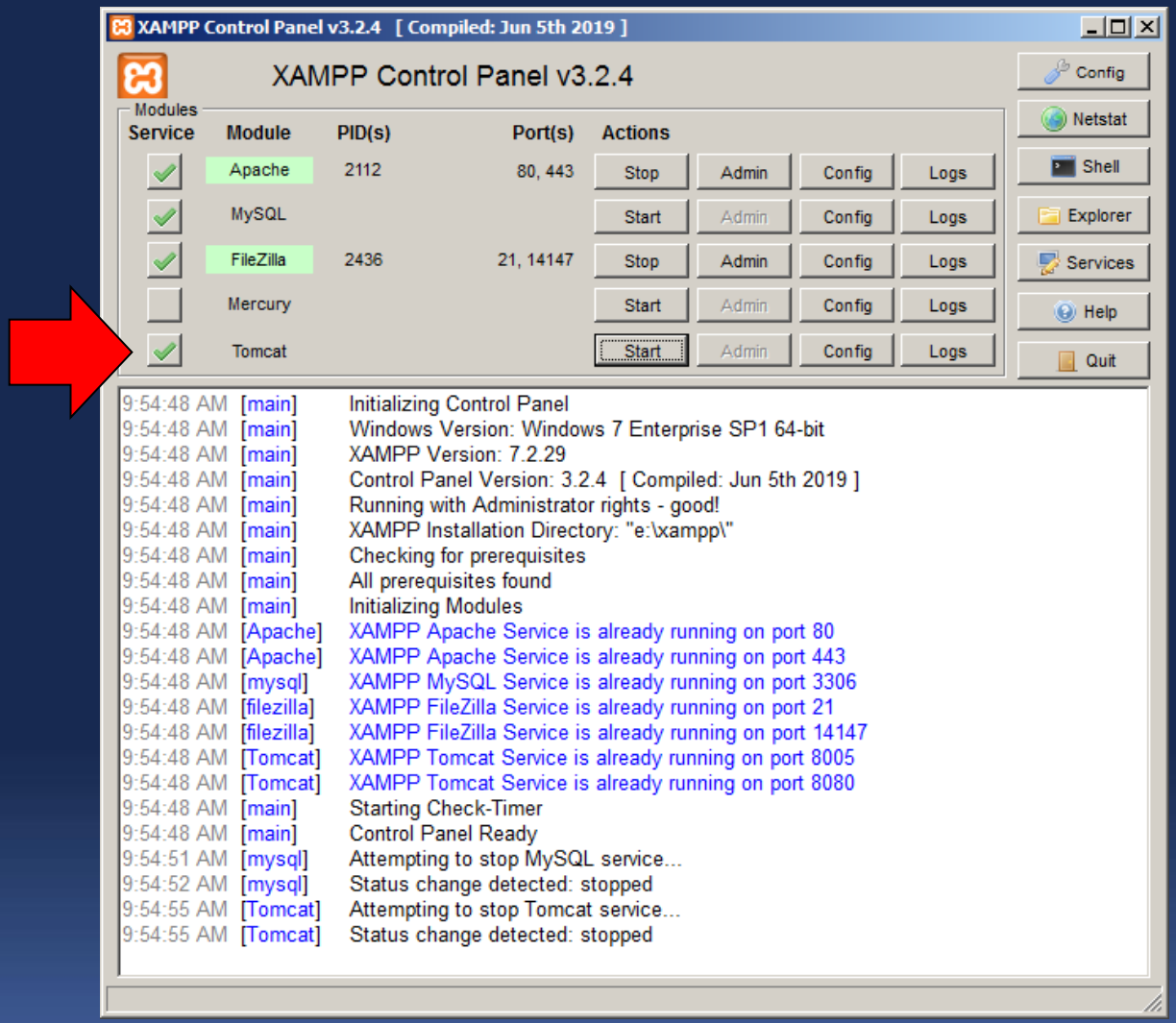
<terminated> ScpeApplication [Java Application] C:\Program Files\Java\jre1.8.0_241\bin\javaw.exe (Jul 14, 2020, 9:19:19 PM)

```
2020-07-14 21:19:21.447  INFO 6440 --- [           main] ConditionEvaluationReportl

Error starting ApplicationContext. To display the conditions report re-run your ap
2020-07-14 21:19:21.452 ERROR 6440 --- [           main] o.s.b.d.LoggingFailureAna

***************************
APPLICATION FAILED TO START
***************************

Description:

Web server failed to start. Port 8080 was already in use.

Action:

Identify and stop the process that's listening on port 8080 or configure this appl
```

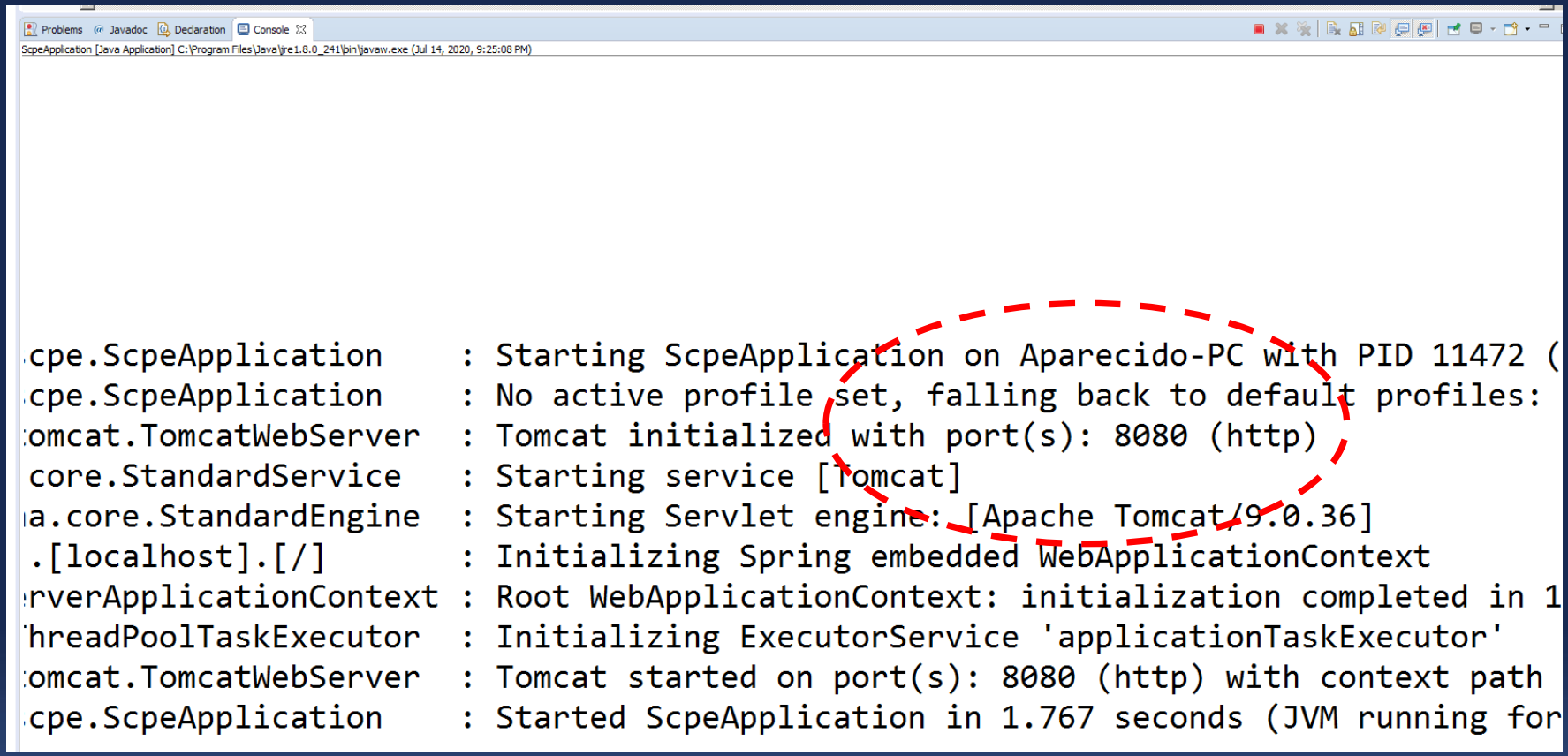
Desativando Tomcat - XAMPP



Desativando Tomcat - XAMPP

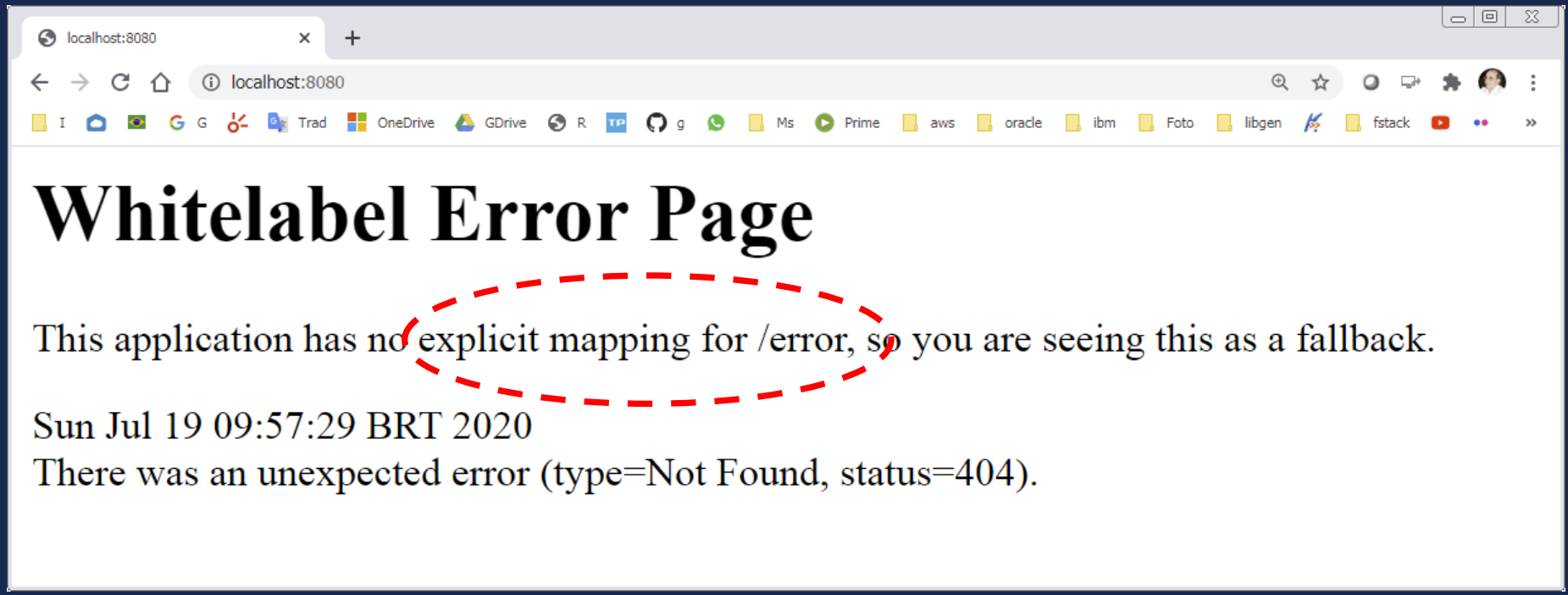


Reexecutando a aplicação



Executando

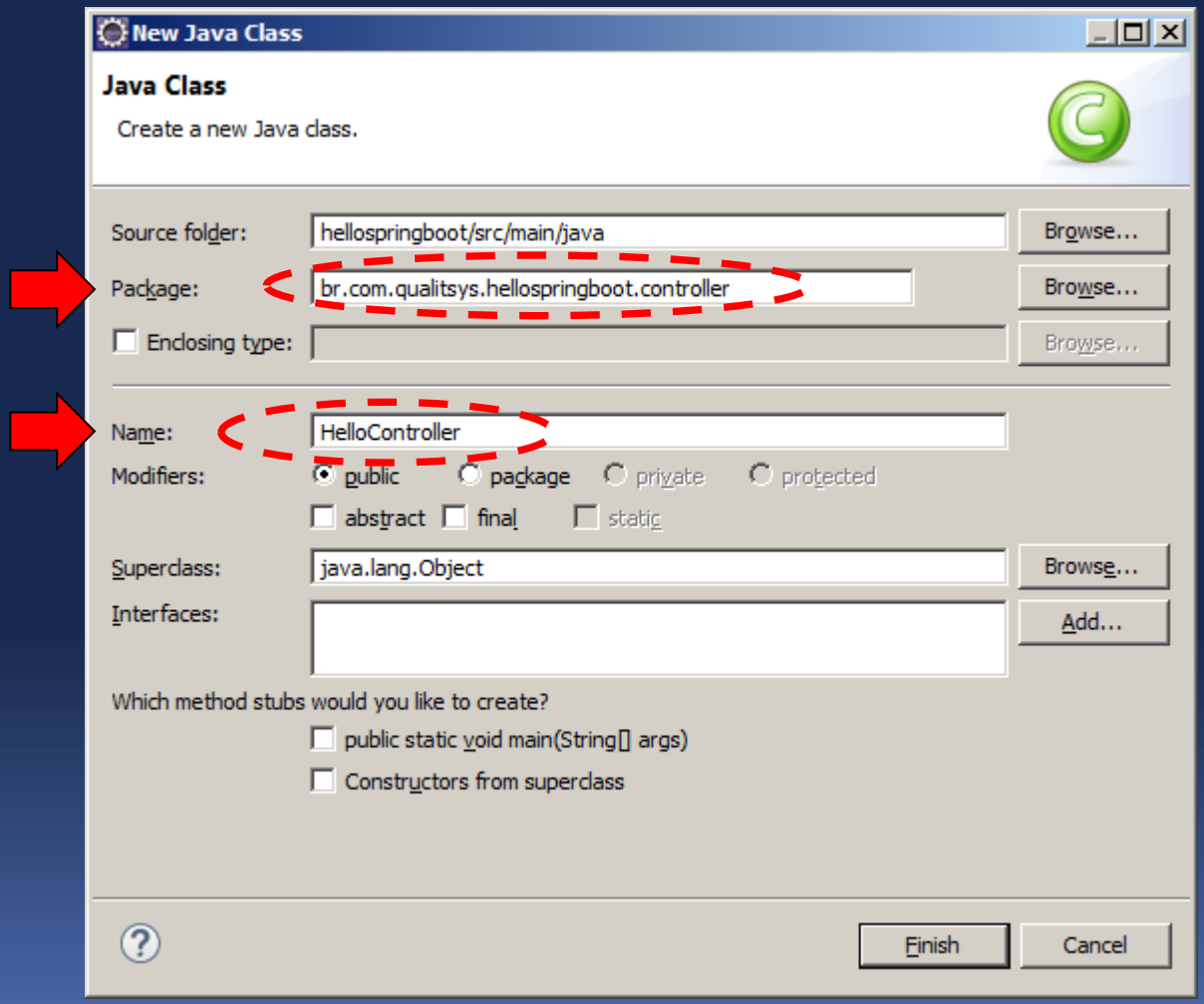
http://localhost:8080



- **Tomcat já está rodando !!!!**
- Foi retornada uma tela de erro do **Spring Boot** pois o contexto **/** da aplicação **não** está **mapeado**;
- Mas, o **Spring Boot** foi executado com sucesso!

Criando um Controller

- Criaremos um **controller** e o mapearemos para o endereço "/";
- Verificaremos se o **Spring** faz a chamada desse controller;

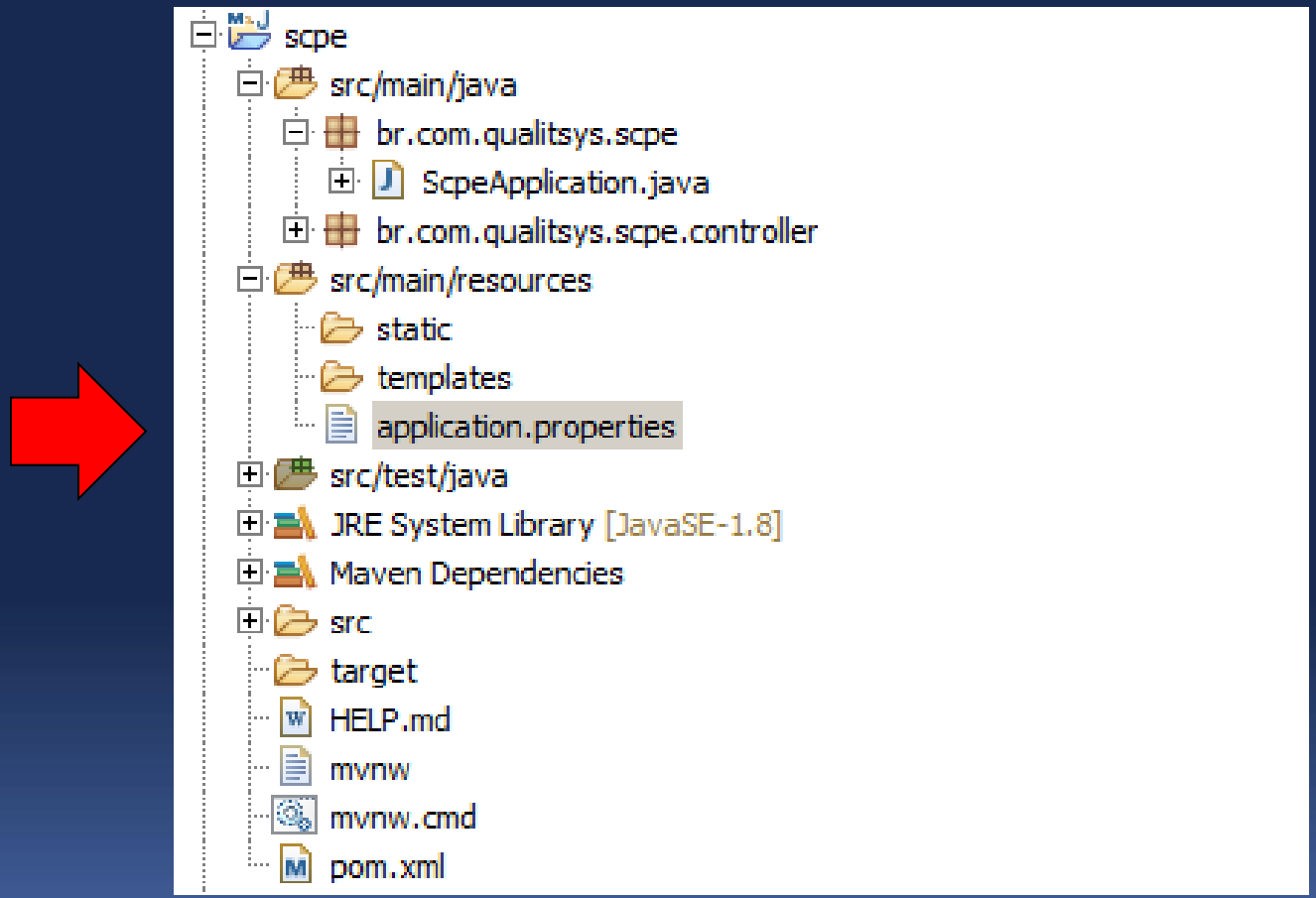


Criando um Controller

```
1 package br.com.qualitsys.hellospringboot.controller;
2
3 public class HelloController {
4
5 }
```

Mudando a porta do Tomcat

- A porta do **Tomcat** pode ser configurada no arquivo **application.properties**, **na pasta src/main/resources**.

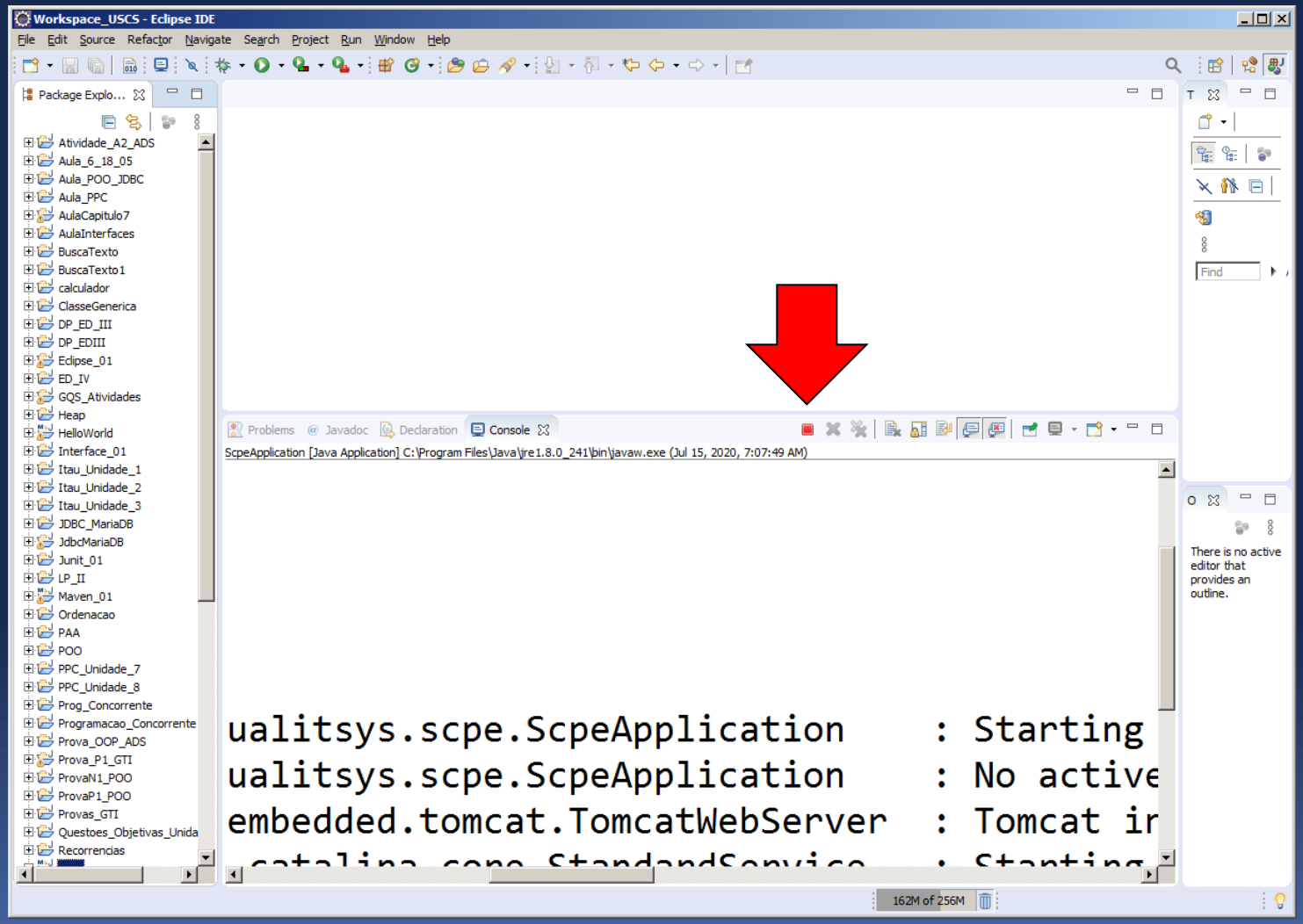


Mudando a porta do Tomcat

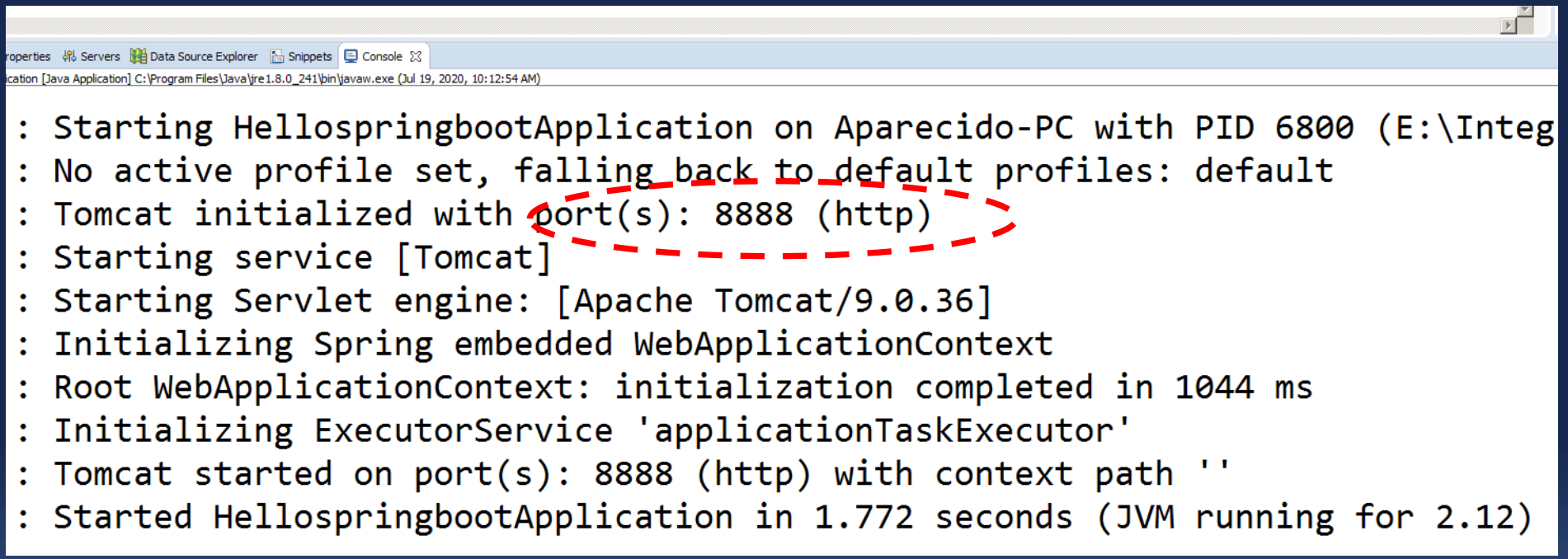
- A porta do Tomcat pode ser configurada no arquivo **application.properties**.

Stop Tomcat na porta 8080

- Encerramento pode ser feito a partir da IDE Eclipse

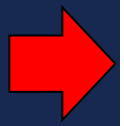


Start Tomcat na porta 8888
Reiniciando a aplicação



Criando um Controller

- O Spring Boot usa nos bastidores o **Spring MVC**;
- Para que essa classe seja considerada um controller pelo **Spring MVC** a anotaremos com a anotation **@Controller**.



```
package br.com.qualitsys.hellospringboot.controller;

import org.springframework.stereotype.Controller;

@Controller
public class HelloController {

}
```

Criando um Controller

- Na classe **HelloController** incluiremos um método chamado hello() que irá retornar o string "Hello Spring Boot...".

```
package br.com.qualitsys.hellospringboot.controller;

import org.springframework.stereotype.Controller;

@Controller
public class HelloController {

    public String hello() {

        return("Hello Spring Boot...");
    }

}
```

Criando um Controller

- Para fazer o mapeamento da **url** que irá estar associada à esse método, incluiremos a annotation @ RequestMapping("/") ;
- Com isso, quando a requisição for feita na porta 8080 ("/") esse método deverá ser chamado pelo Spring Boot;

```
package br.com.qualitsys.hellospringboot.controller;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;

@Controller
public class HelloController {

    @RequestMapping("/")
    public String hello() {

        return("Hello Spring Boot...");
    }

}
```

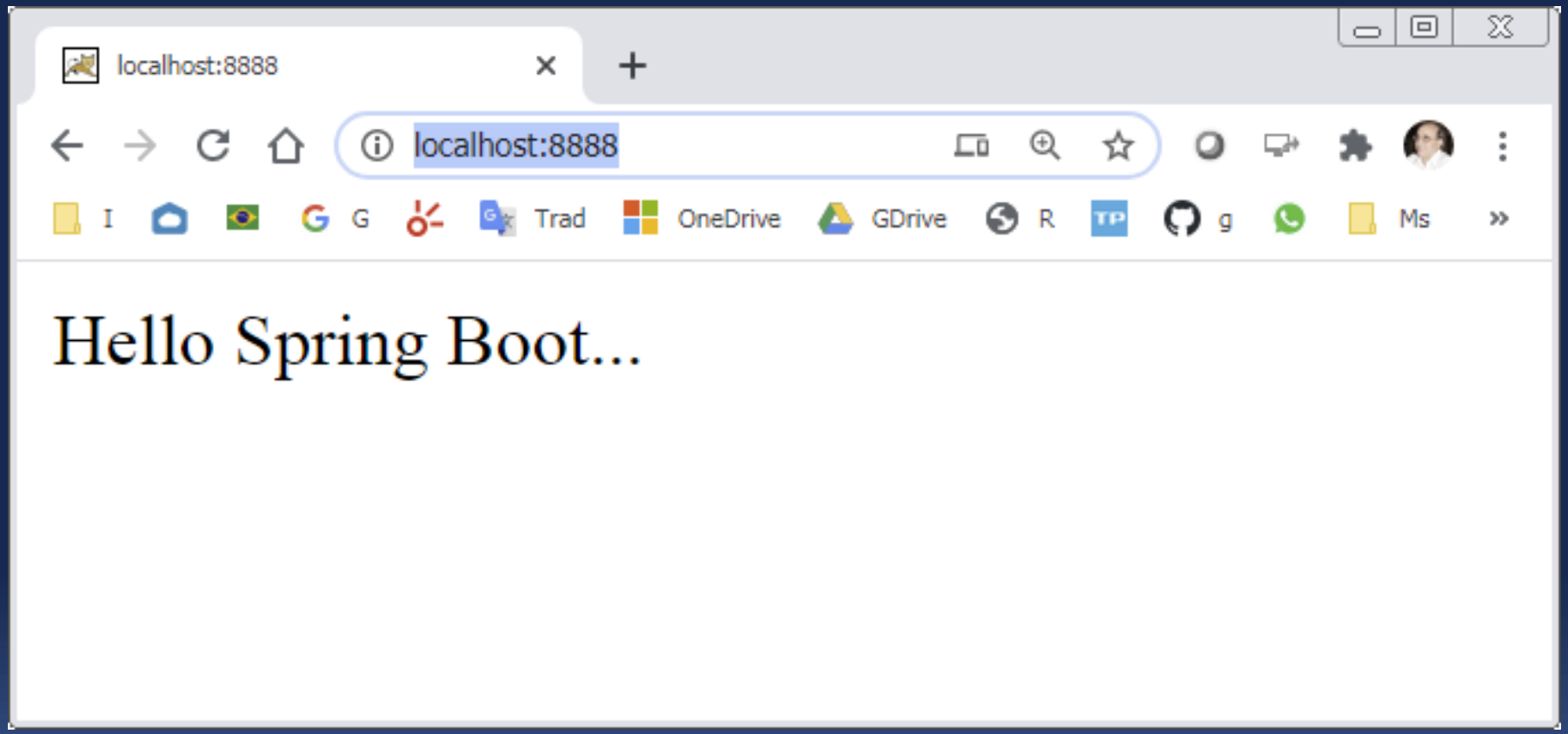
Criando um Controller

- O mapeamento que iremos fazer não é para uma nova página;
- Queremos nesse caso apenas retornar o string "Hello Spring Boot";
- Para isso iremos incluir a annotation **@ResponseBody**
- Sem essa anotação, o Spring irá tratar o string "Hello Spring Boot" como uma nova página JSP ou Thymeleaf.

```
package br.com.qualitsys.hellospringboot.controller;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.ResponseBody;

@Controller
public class HelloController {

    @RequestMapping("/")
    @ResponseBody
    public String hello() {

        return("Hello Spring Boot...");
    }

}
```

Executando ...

http://localhost:8888



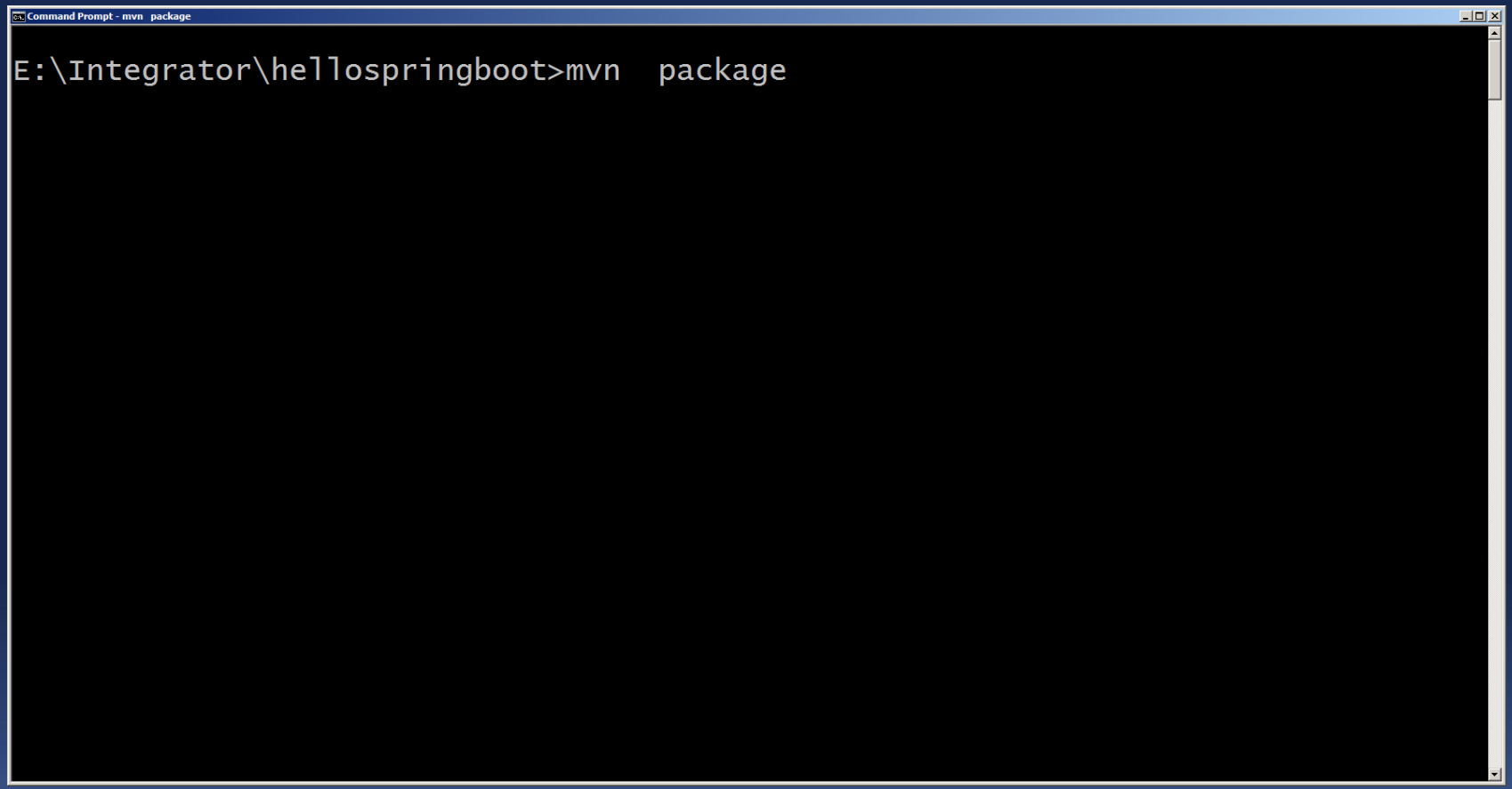
Gerando arquivo .jar da aplicação...

- Para se criar um jar executável, deve-se incluir no arquivo **pom.xml** a dependência **spring-boot-maven-plugin**.

```
<build>
    <plugins>
        <plugin>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-maven-plugin</artifactId>
        </plugin>
    </plugins>
</build>
```

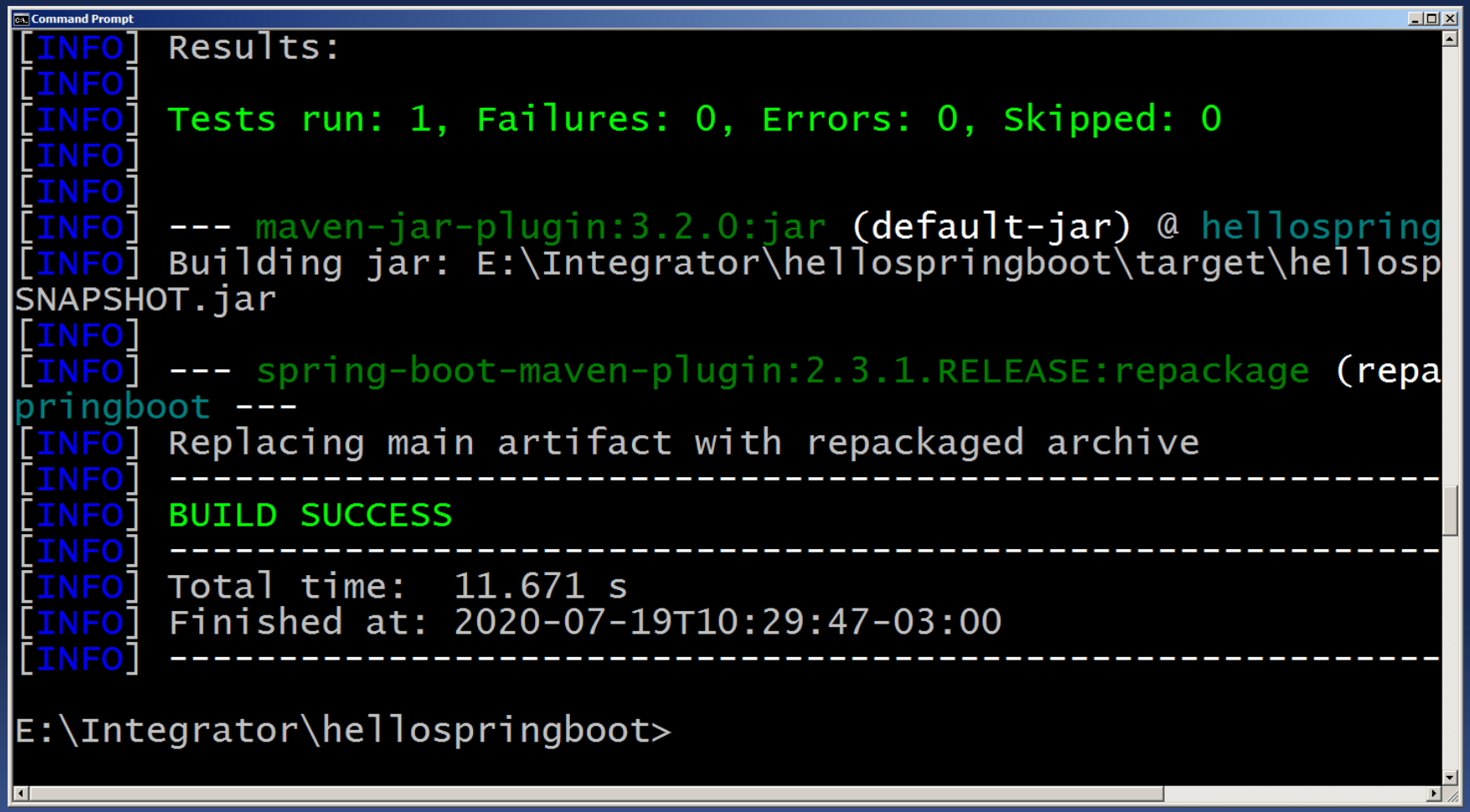
Gerando arquivo .jar da aplicação...

- Na pasta do projeto, executar o comando: $ mvn package



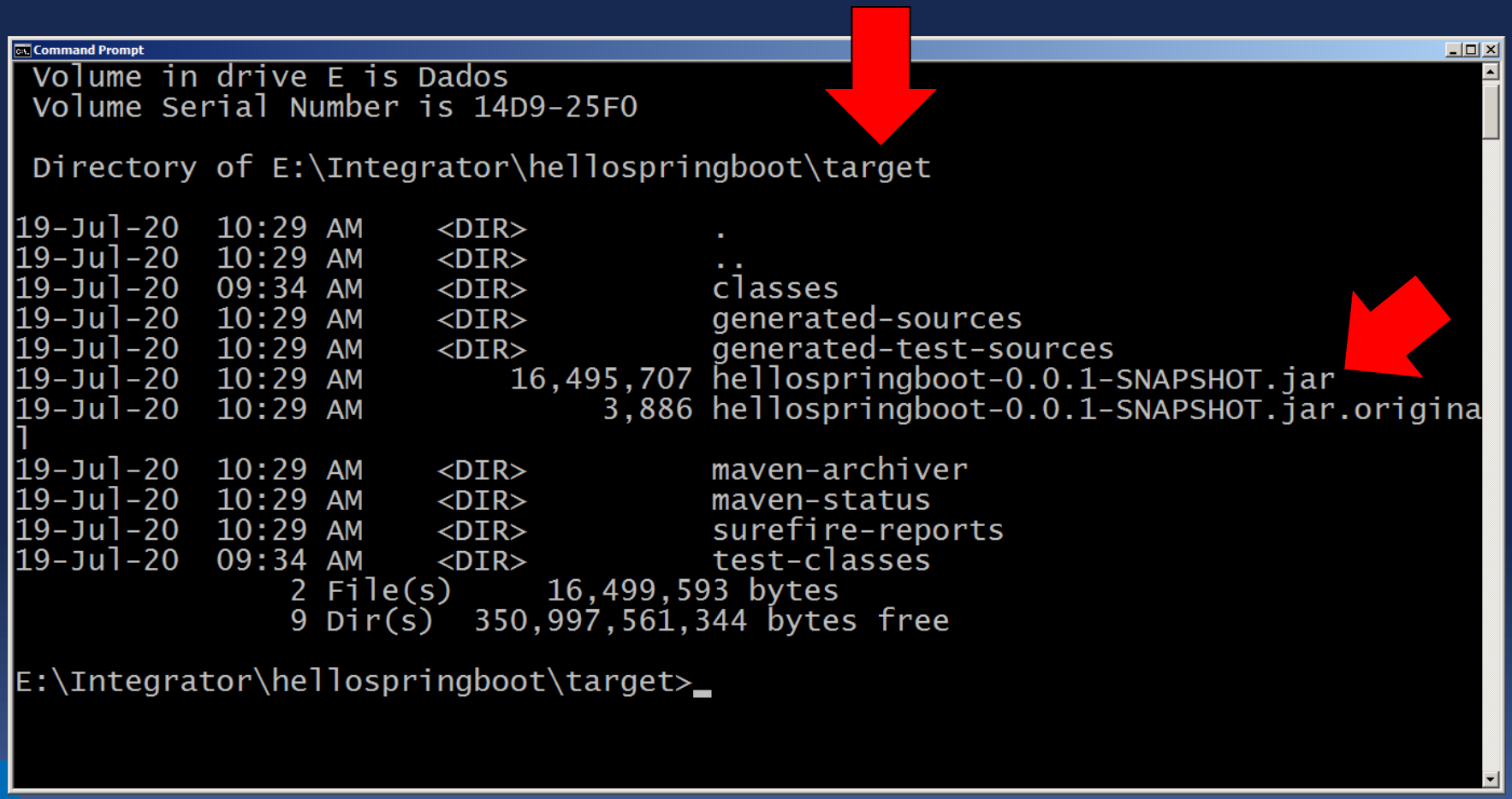
Gerando arquivo .jar da aplicação...

- Na pasta do projeto, executar o comando: $ mvn package



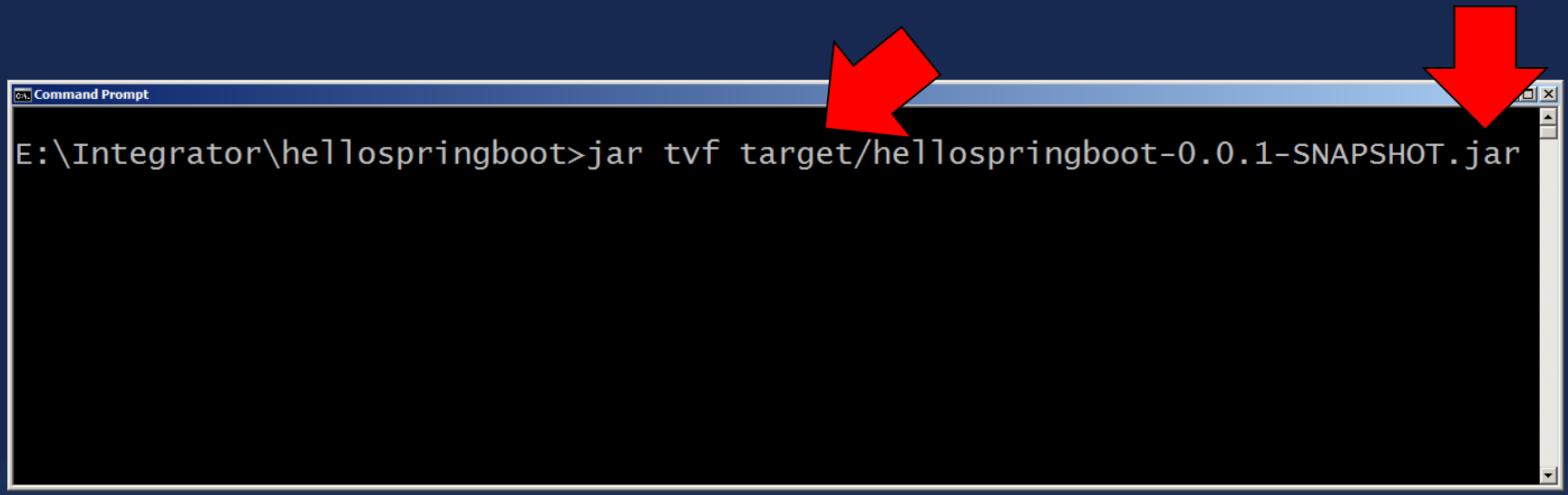
Gerando arquivo .jar da aplicação...

- Na pasta **target** do projeto, pode-se ver o arquivo myproject-0.0.1-SNAPSHOT.jar;
- Este arquivo deve ter em torno de 16 MB de tamanho.



Gerando arquivo .jar da aplicação...

- Vamos dar uma "espiada" no arquivo .jar gerado pelo Maven;
- $ jar tvf target/hellospringboot-0.0.1-SNAPSHOT.jar



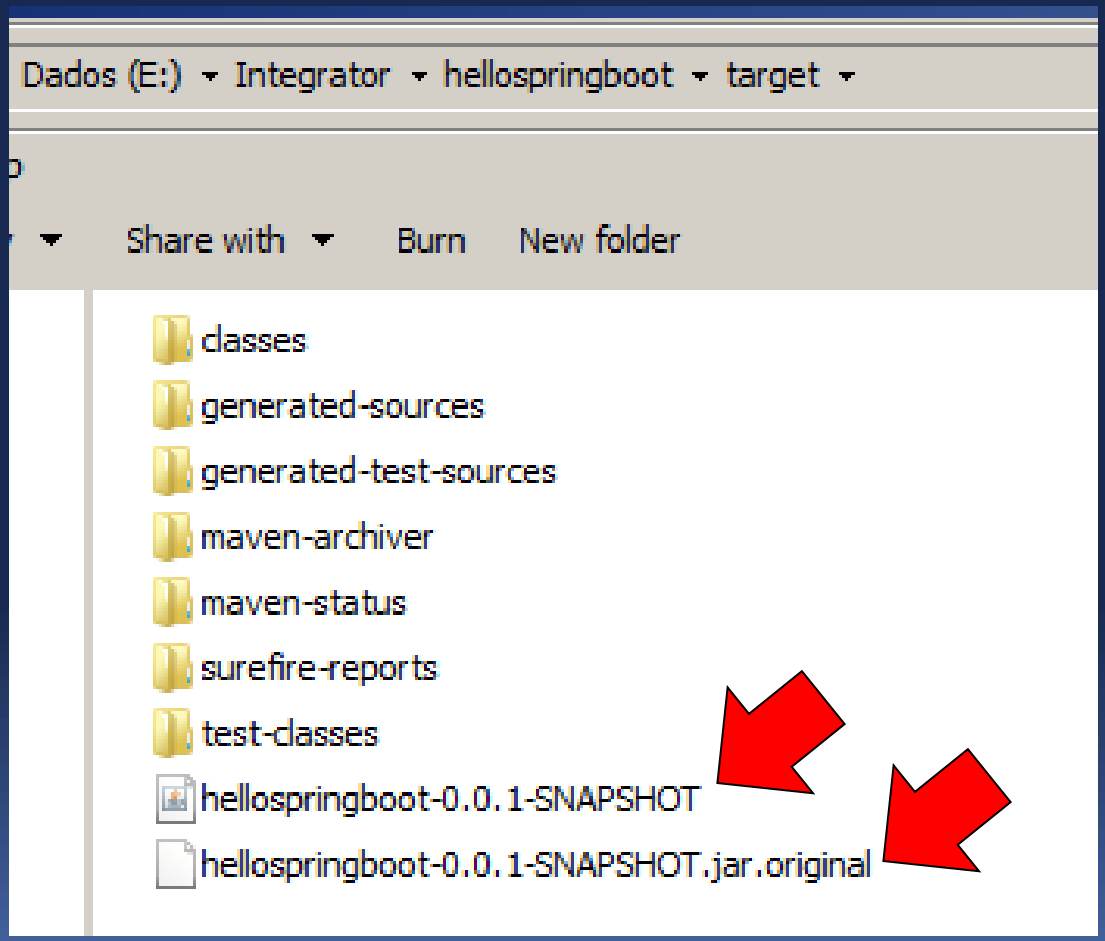
Gerando arquivo .jar da aplicação...

- Vamos dar uma "espiada" no arquivo .jar gerado pelo Maven;
- $ jar tvf target/hellospringboot-0.0.1-SNAPSHOT.jar

```
Command Prompt
  4789 Thu Jun 11 22:50:44 BRT 2020 BOOT-INF/lib/spring-boot-starter-tomcat-2.3.
1.RELEASE.jar
3373581 Wed Jun 03 18:03:54 BRT 2020 BOOT-INF/lib/tomcat-embed-core-9.0.36.jar
237826 Mon Aug 26 10:58:30 BRT 2019 BOOT-INF/lib/jakarta.el-3.0.3.jar
268491 Wed Jun 03 18:03:54 BRT 2020 BOOT-INF/lib/tomcat-embed-websocket-9.0.36.j
ar
1440651 Tue Jun 09 06:43:38 BRT 2020 BOOT-INF/lib/spring-web-5.2.7.RELEASE.jar
688811 Tue Jun 09 06:43:06 BRT 2020 BOOT-INF/lib/spring-beans-5.2.7.RELEASE.jar
956463 Tue Jun 09 06:43:42 BRT 2020 BOOT-INF/lib/spring-webmvc-5.2.7.RELEASE.jar

372705 Tue Jun 09 06:43:12 BRT 2020 BOOT-INF/lib/spring-aop-5.2.7.RELEASE.jar
1227929 Tue Jun 09 06:43:30 BRT 2020 BOOT-INF/lib/spring-context-5.2.7.RELEASE.j
ar
282183 Tue Jun 09 06:43:00 BRT 2020 BOOT-INF/lib/spring-expression-5.2.7.RELEASE
.jar
 41472 Mon Dec 16 22:03:32 BRT 2019 BOOT-INF/lib/slf4j-api-1.7.30.jar
1441820 Tue Jun 09 06:43:00 BRT 2020 BOOT-INF/lib/spring-core-5.2.7.RELEASE.jar
 23961 Tue Jun 09 06:42:50 BRT 2020 BOOT-INF/lib/spring-jcl-5.2.7.RELEASE.jar
  1135 Sun Jul 19 10:29:46 BRT 2020 BOOT-INF/classpath.idx

E:\Integrator\hellospringboot>
```

Gerando arquivo .jar da aplicação...

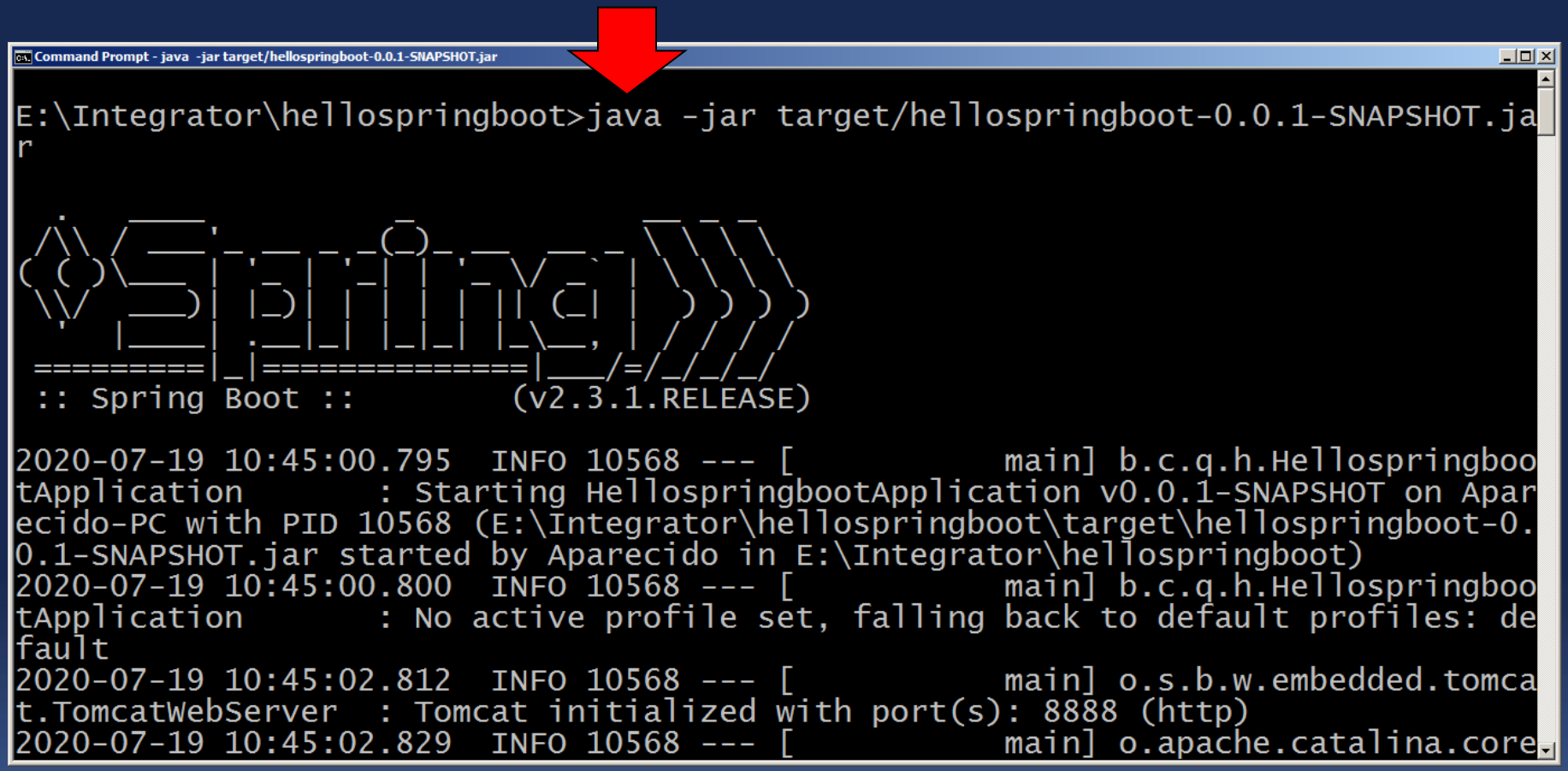
- Pode-se também observar na pasta target um arquivo menor chamado **hellospringboot-0.0.1-SNAPSHOT.jar.original**;
- Esse arquivo corresponde ao arquivo jar original criado pelo Maven antes de ser reempacotado pelo **Spring Boot**.



Rodando a aplicação

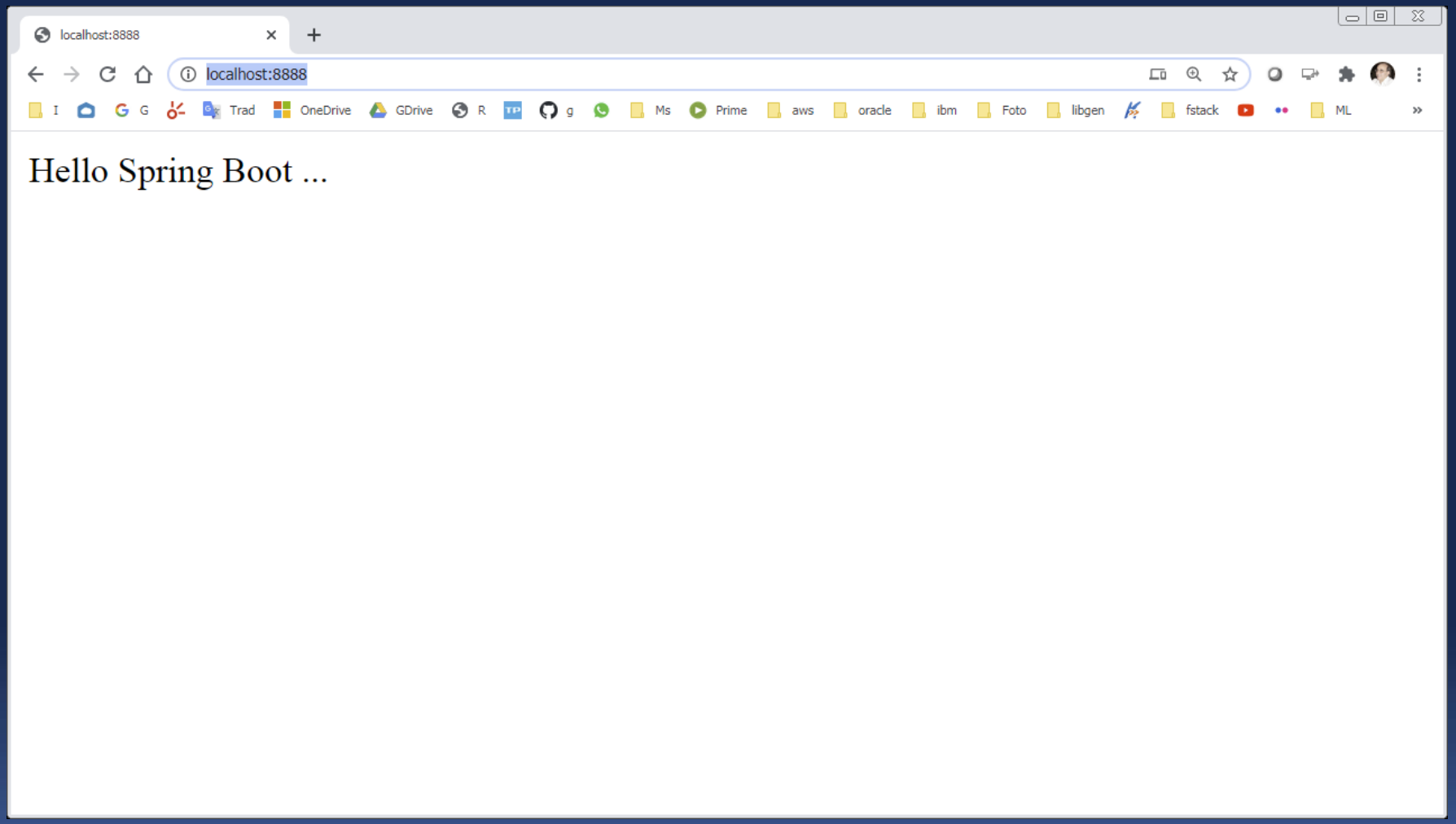
- Para executar a aplicação execute o comando:

$ java -jar target/myproject-0.0.1-SNAPSHOT.jar.



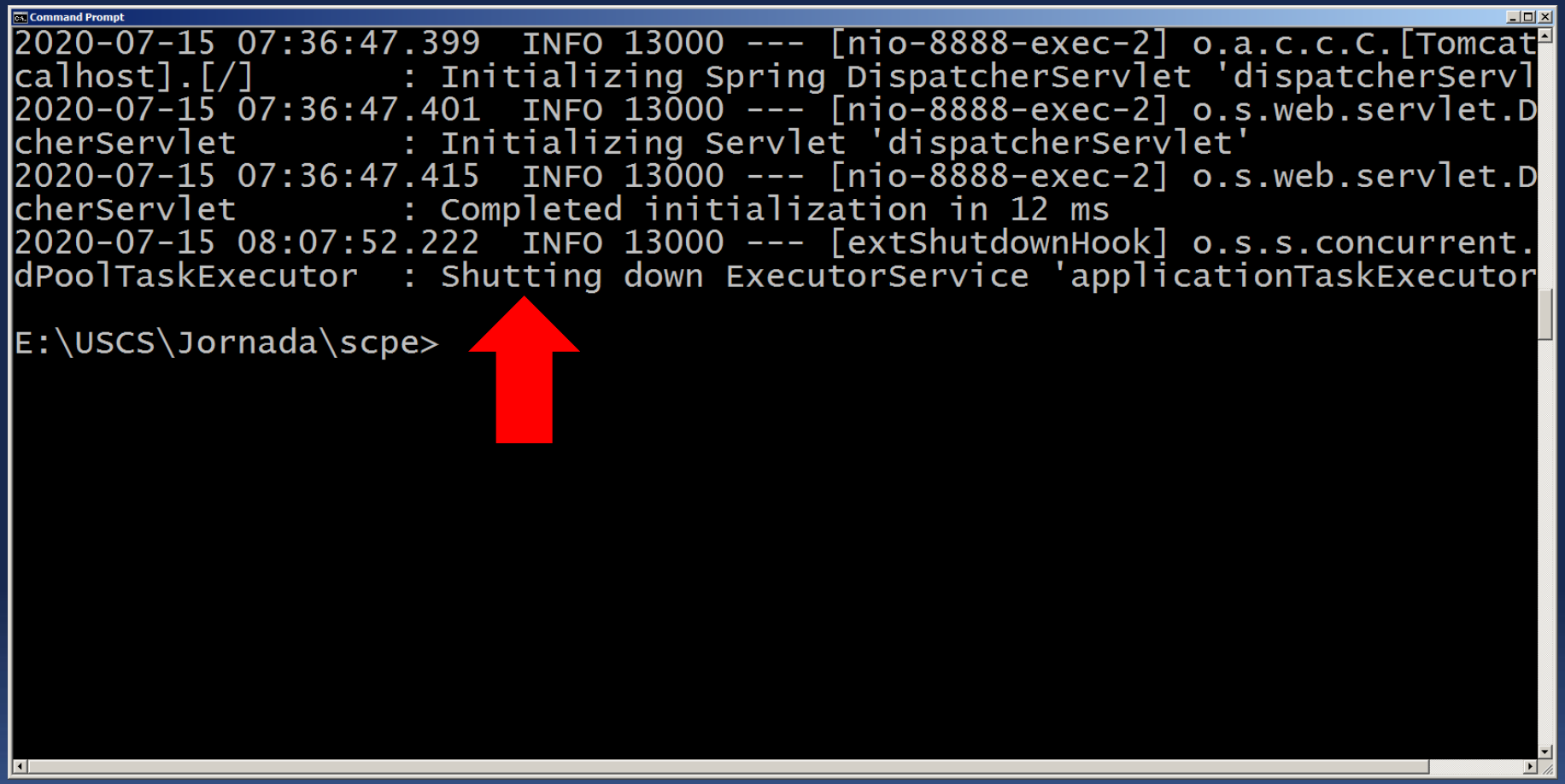
Chamando a aplicação pelo Browser

- http://localhost:8888



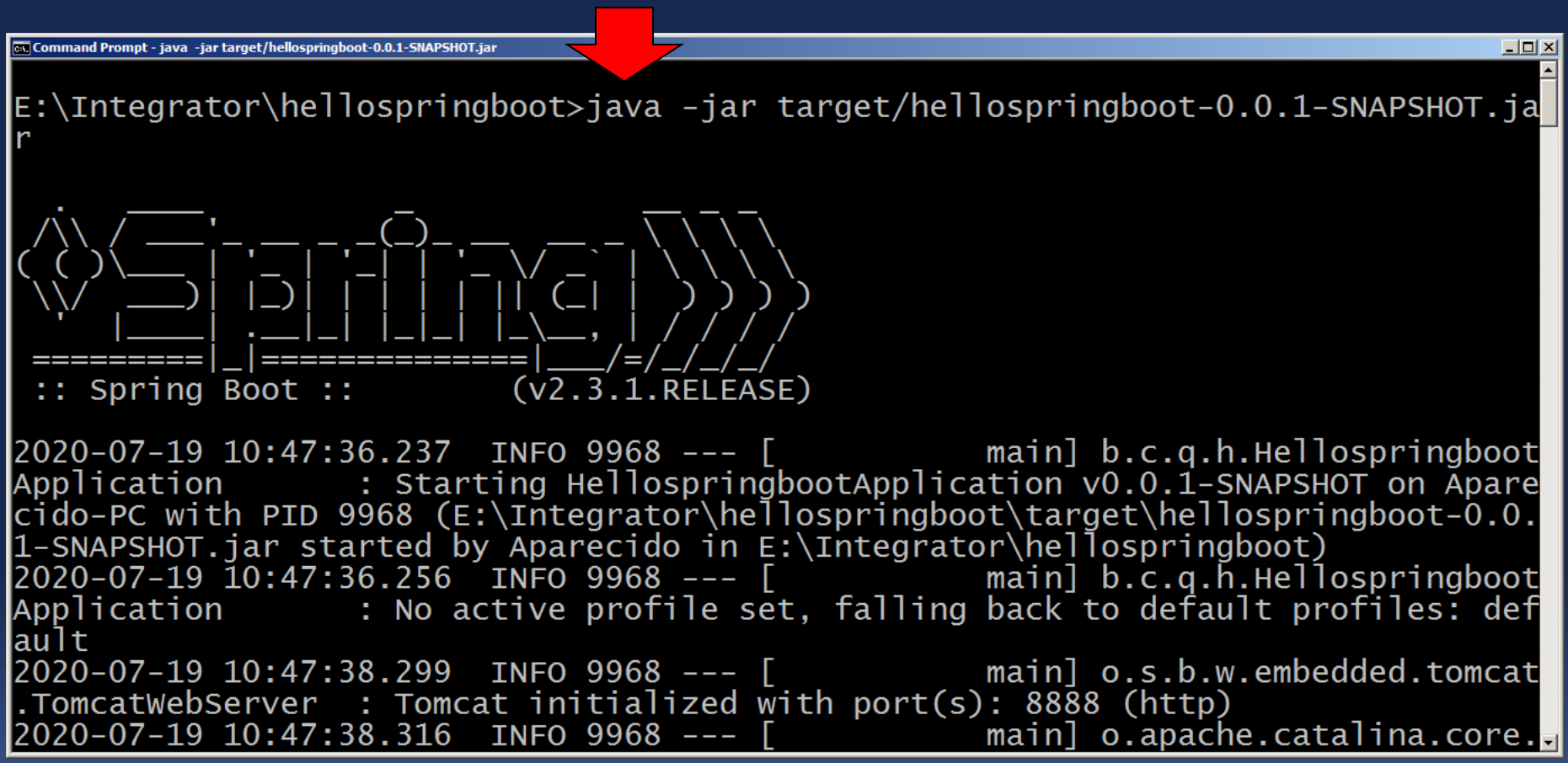
Encerrando a aplicação

Ctrl + C



Usando postman para enviar requisições http

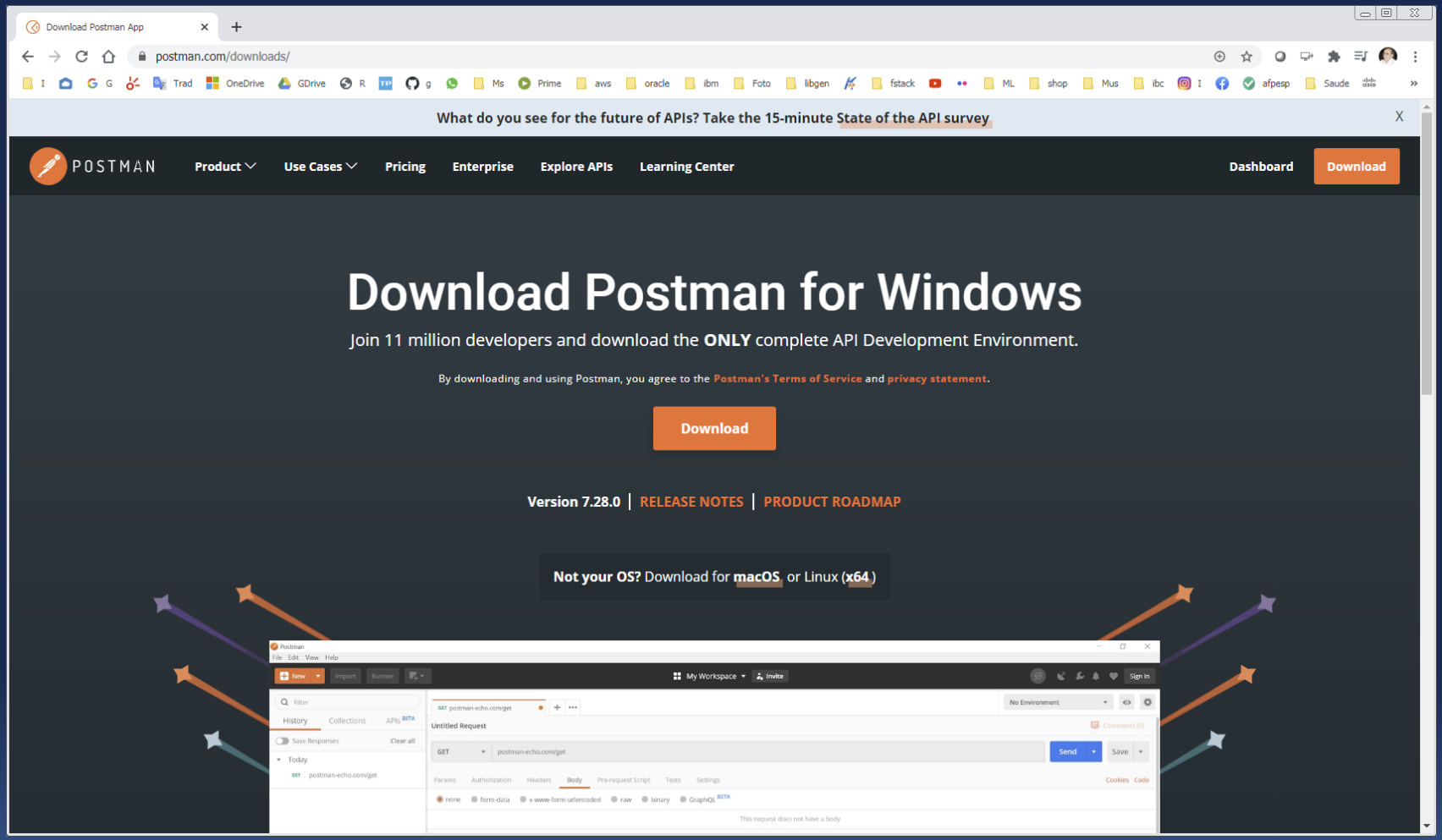
Reiniciando a aplicação



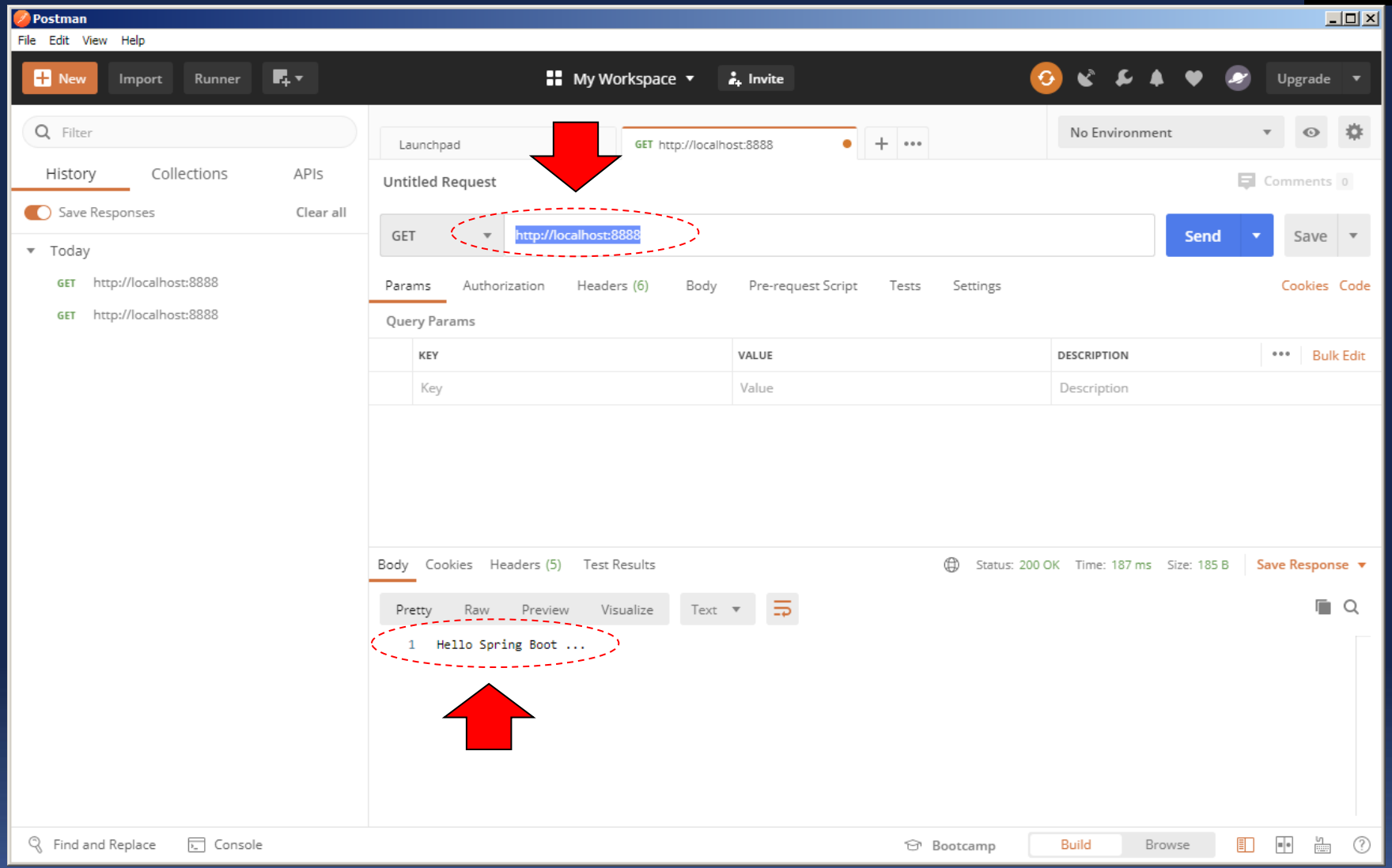
Postman

- O **Postman** é uma ferramenta para teste de API's que utilizam o padrão REST;
- Muito útil, pois é independente de linguagem, podendo assim efetuar testes diversificados, além de combinação de parâmetros e retornos;
- Faça o download do Postman pelo link: **https://www.getpostman.com/downloads/**

Postman


Download Postman App
postman.com/downloads/
What do you see for the future of APIs? Take the 15-minute State of the API survey
POSTMAN
Product
Use Cases
Pricing
Enterprise
Explore APIs
Learning Center
Dashboard
Download
Download Postman for Windows
Join 11 million developers and download the ONLY complete API Development Environment.
By downloading and using Postman, you agree to the Postman's Terms of Service and privacy statement.
Download
Version 7.28.0 | RELEASE NOTES | PRODUCT ROADMAP
Not your OS? Download for macOS or Linux (x64)

Enviando requisição ao Postman



Criando .war da aplicação Spring Boot

- Modificar o arquivo pom.xml definindo empacotamento **war**

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 ht
    <modelVersion>4.0.0</modelVersion>
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.3.1.RELEASE</version>
        <relativePath/> <!-- lookup parent from repository -
    </parent>
    <groupId>br.com.qualitsys</groupId>
    <artifactId>scpe</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <packaging>war</packaging>
    <name>scpe</name>
    <description>Demo project for Spring Boot</description>

    <properties>
        <java.version>1.8</java.version>
    </properties>

    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
```

Criando .war da aplicação Spring Boot

- Vamos manter o **Tomcat** configurado para o ambiente de desenvolvimento, mas não o utilizaremos no empacotamento final (**war**);
- Para isso, vamos incluir nova dependência no arquivo **pom.xml**.

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-tomcat</artifactId>
    <scope>provided</scope>
</dependency>
```

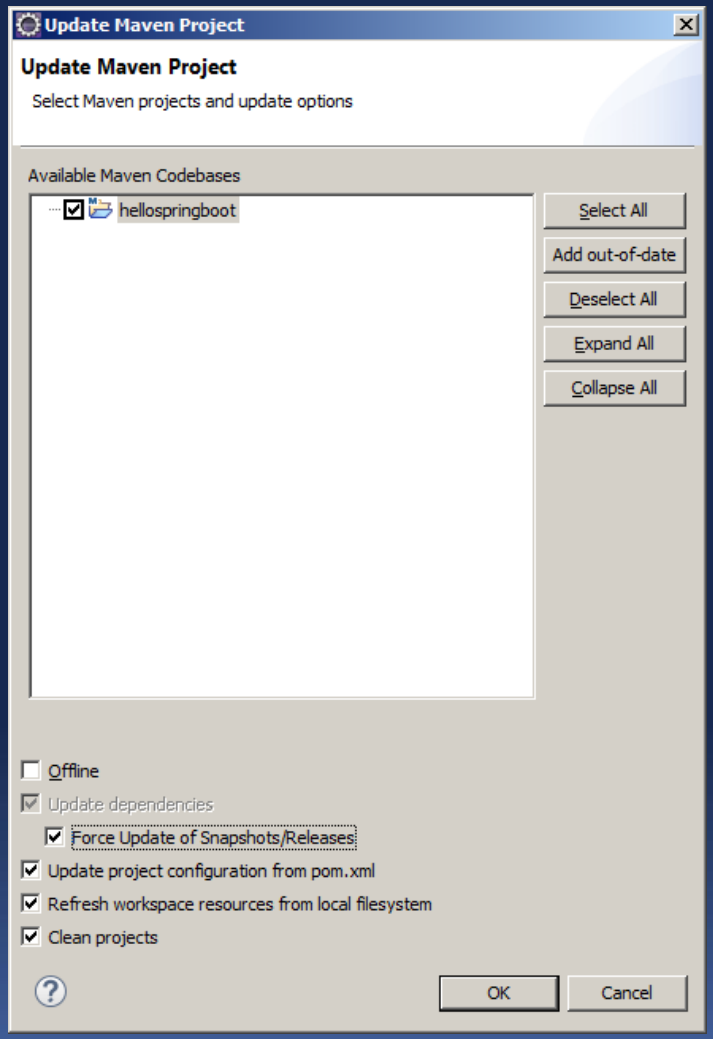
Criando .war da aplicação Spring Boot

- Arquivo pom.xml modificado

```
                <artifactId>junit-vintage-engine</artifactId>
            </exclusion>
        </exclusions>
    </dependency>

    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-tomcat</artifactId>
        <scope>provided</scope>
    </dependency>

</dependencies>

<build>
    <plugins>
        <plugin>
            <groupId>org.springframework.boot</groupId>
```

Atualizando o projeto

- Botão direito no projeto
- Maven > Update Project



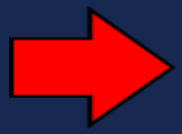
Alterando a classe principal

- Modificaremos a classe principal da aplicação (onde está o método **main**) para que a aplicação possa operar com um servidor de aplicações Tomcat externo;
- Isso é feito definindo-se a classe principal como filha da classe **SprintBootServletInitializer**;

```
package br.com.qualitsys.hellospringboot;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.boot.web.servlet.support.SpringBootServletInitializer;

@SpringBootApplication
public class HellospringbootApplication extends SpringBootServletInitializer{

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

}
```

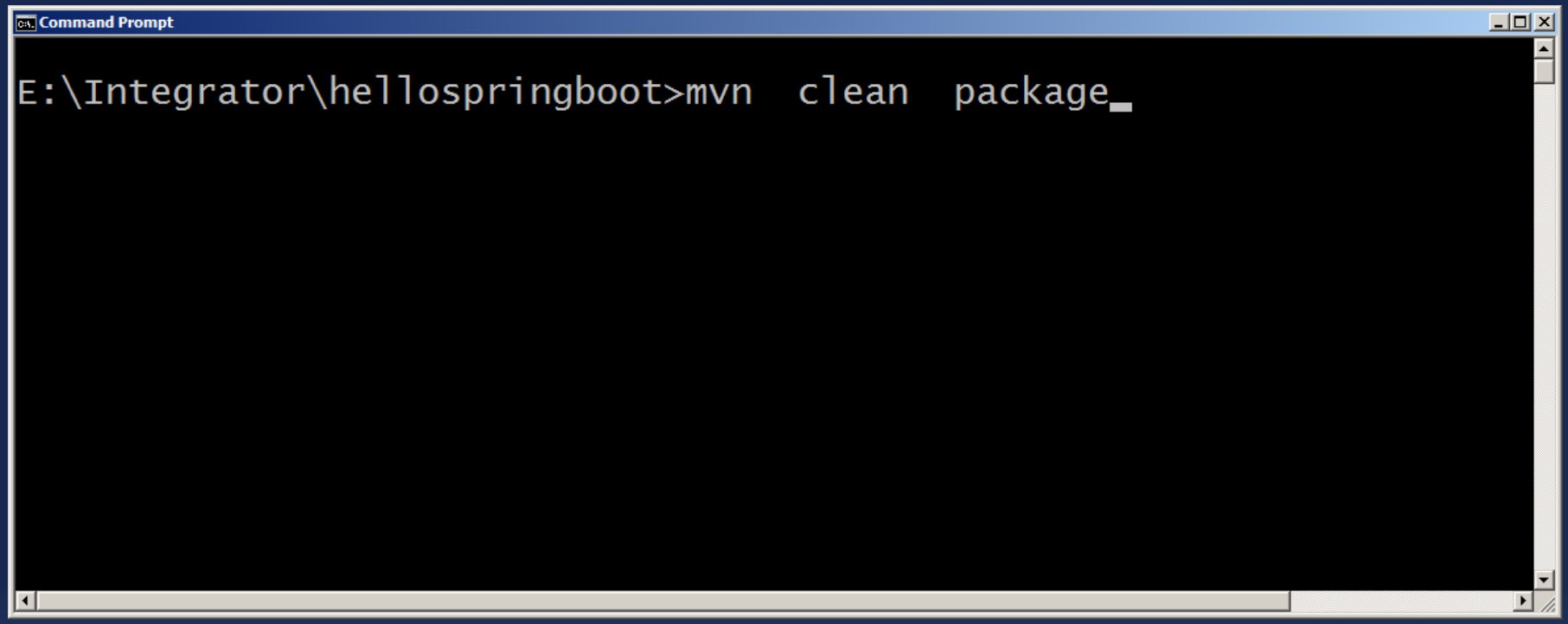
Gerando o arquivo war

Na pasta da aplicação, executar o comando:

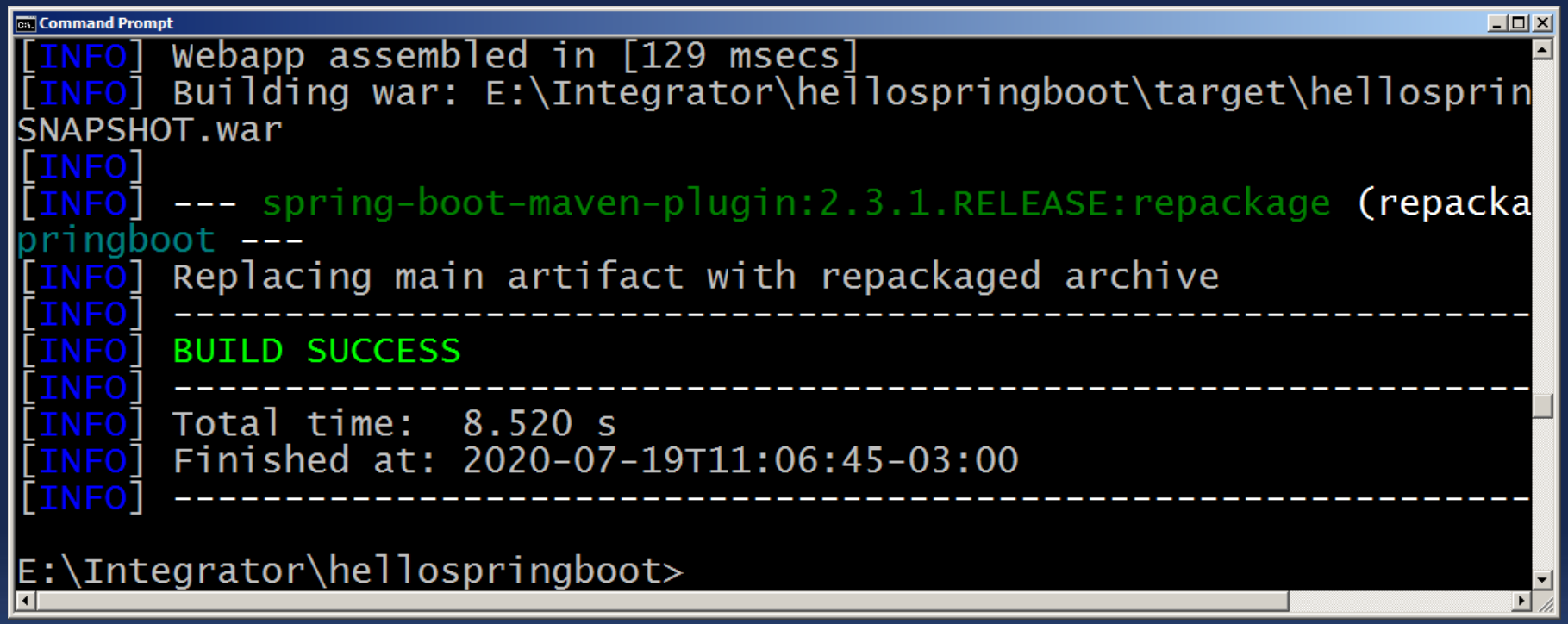


```
$ mvn  clean   package
```

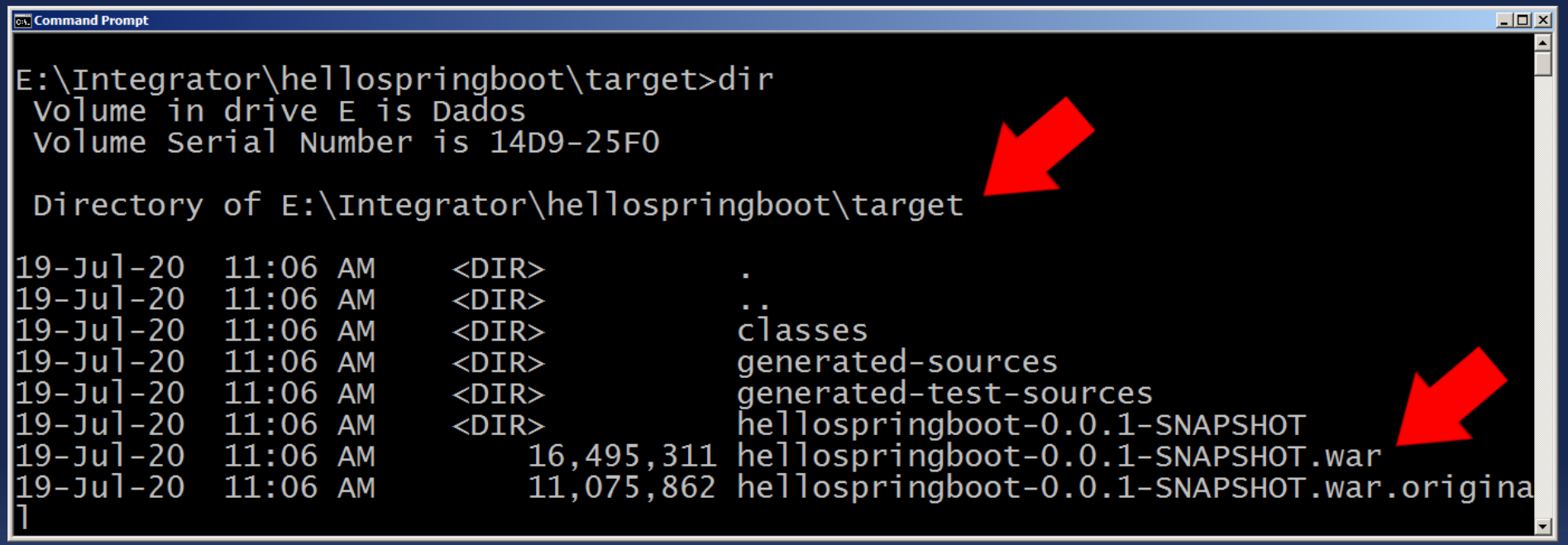
Gerando o arquivo war



Gerando o arquivo war



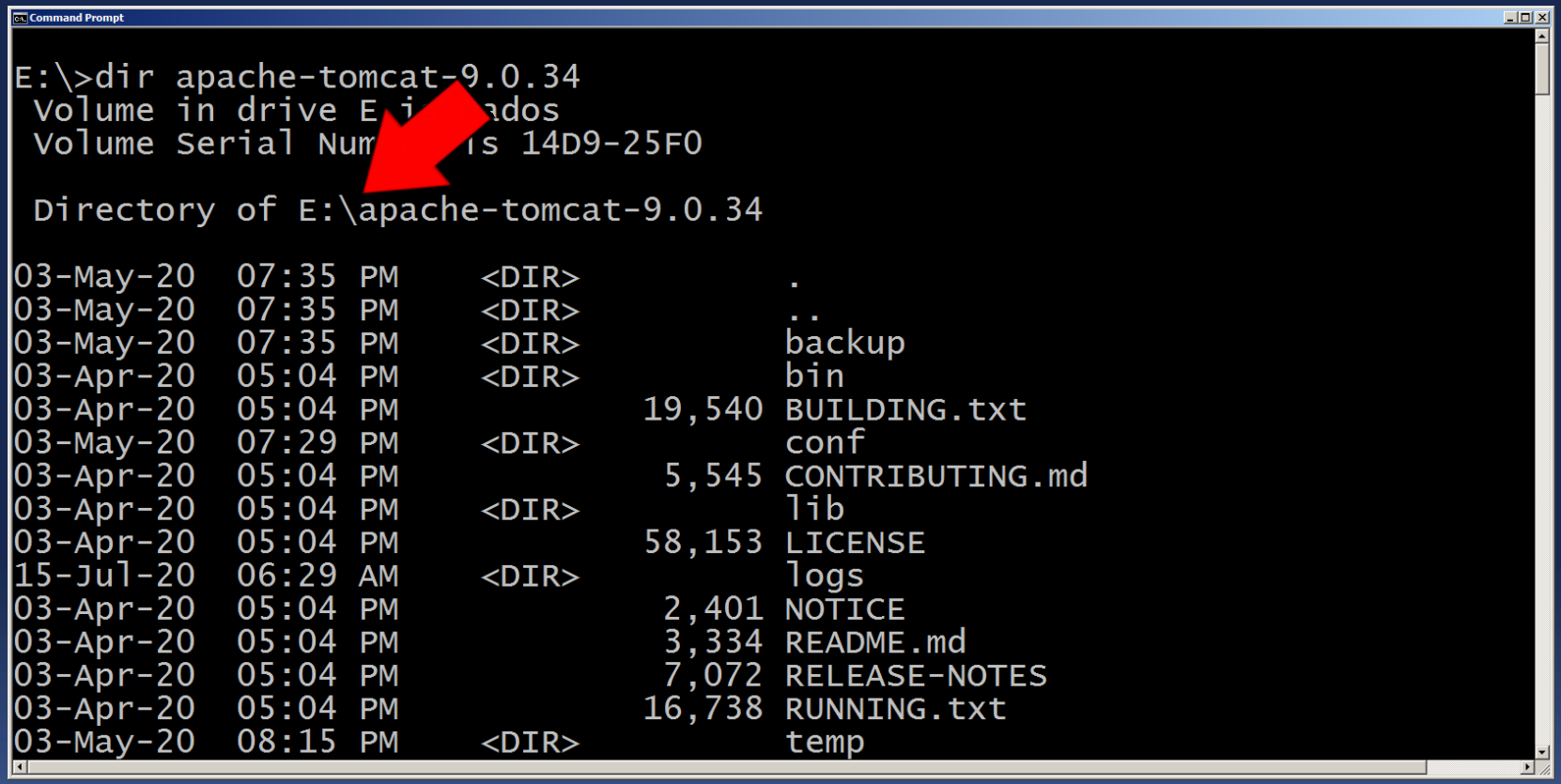
arquivo war gerado na pasta target



Iniciando um Tomcat externo

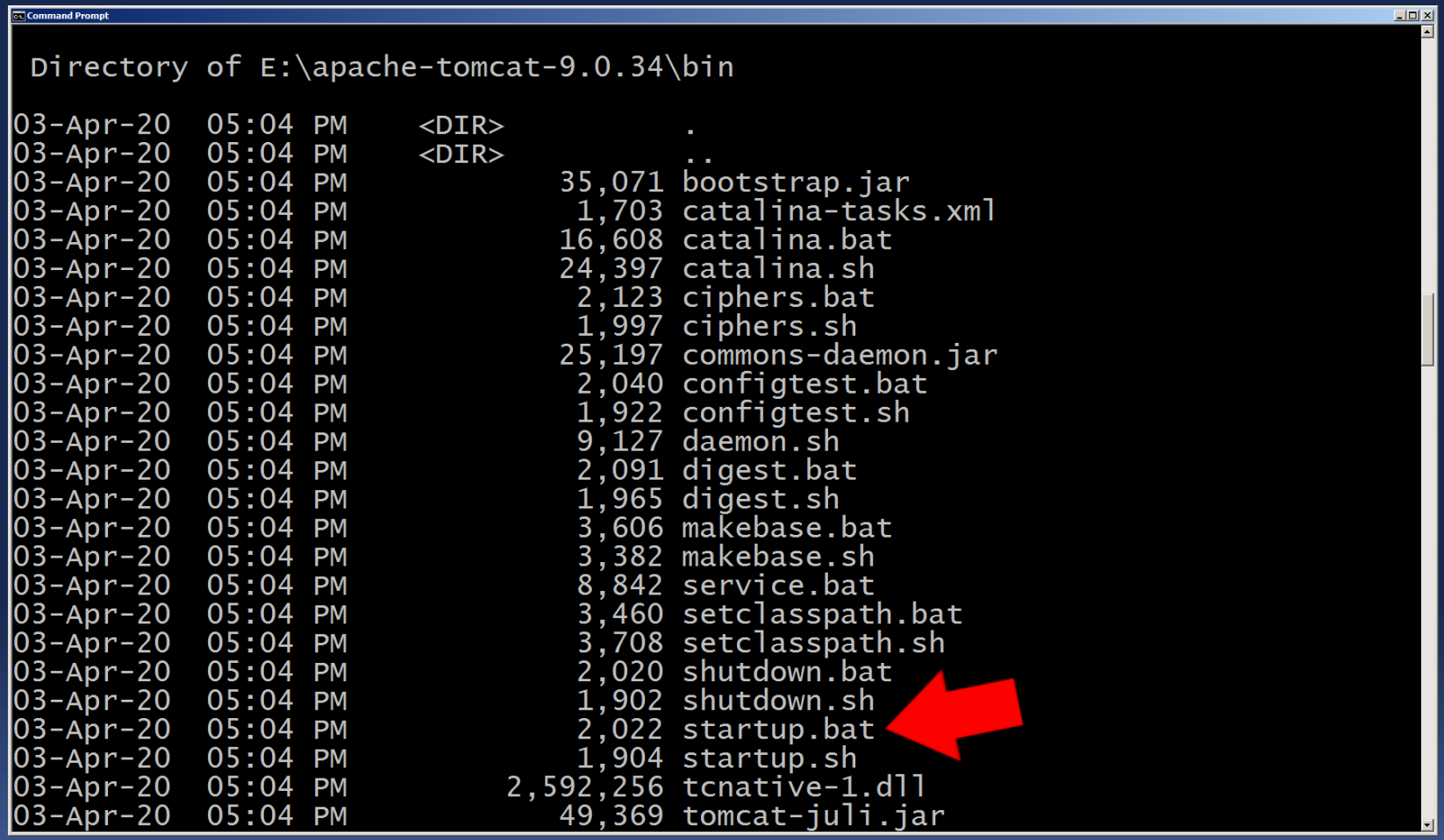
Subindo um Tomcat externo

- Para iniciar um servidor Tomcat, posicionar-se na pasta onde o Tomcat foi instalado.



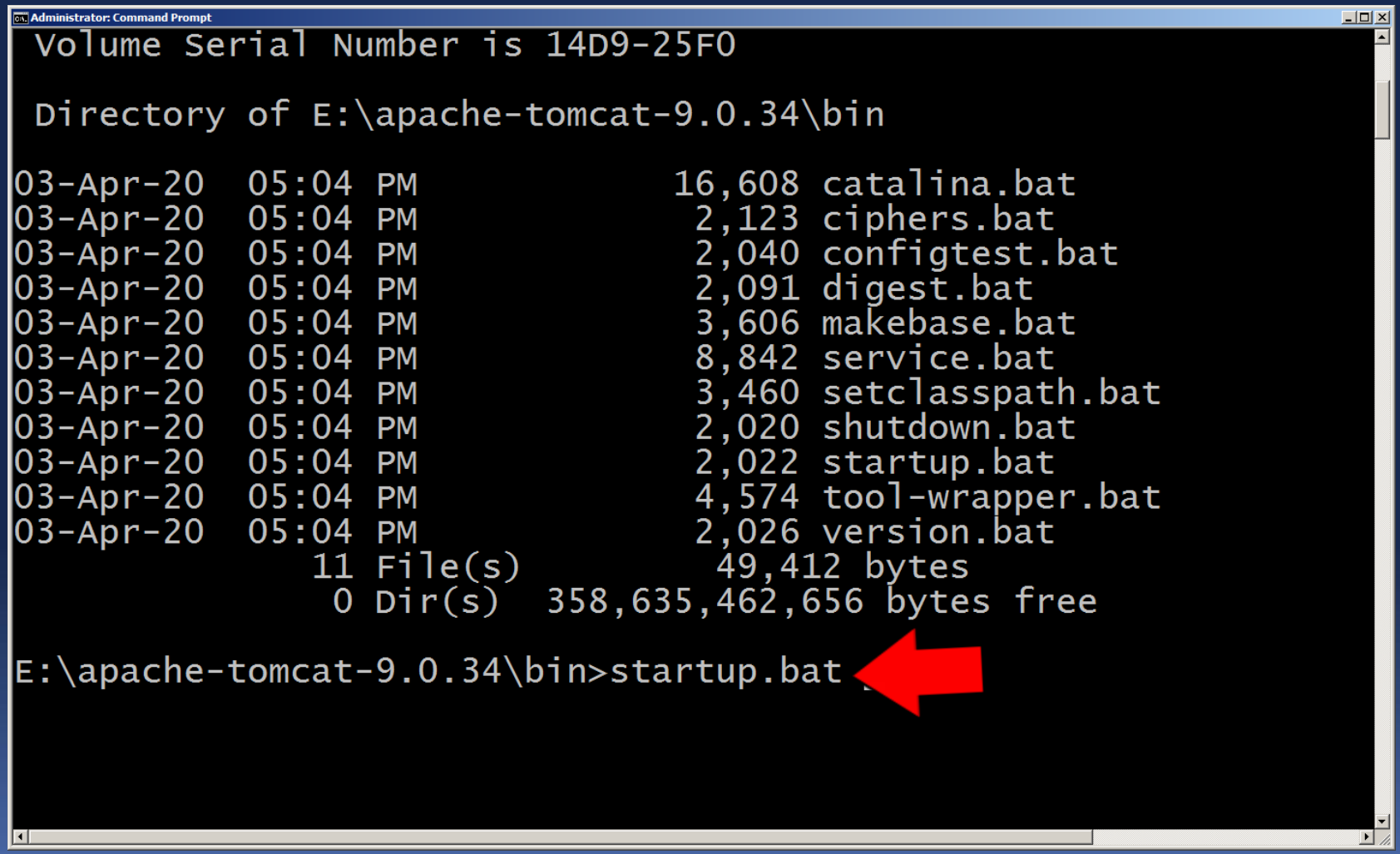
Subindo um Tomcat externo

- No diretório bin, executar - como administrador -- o procedimento **start.bat**



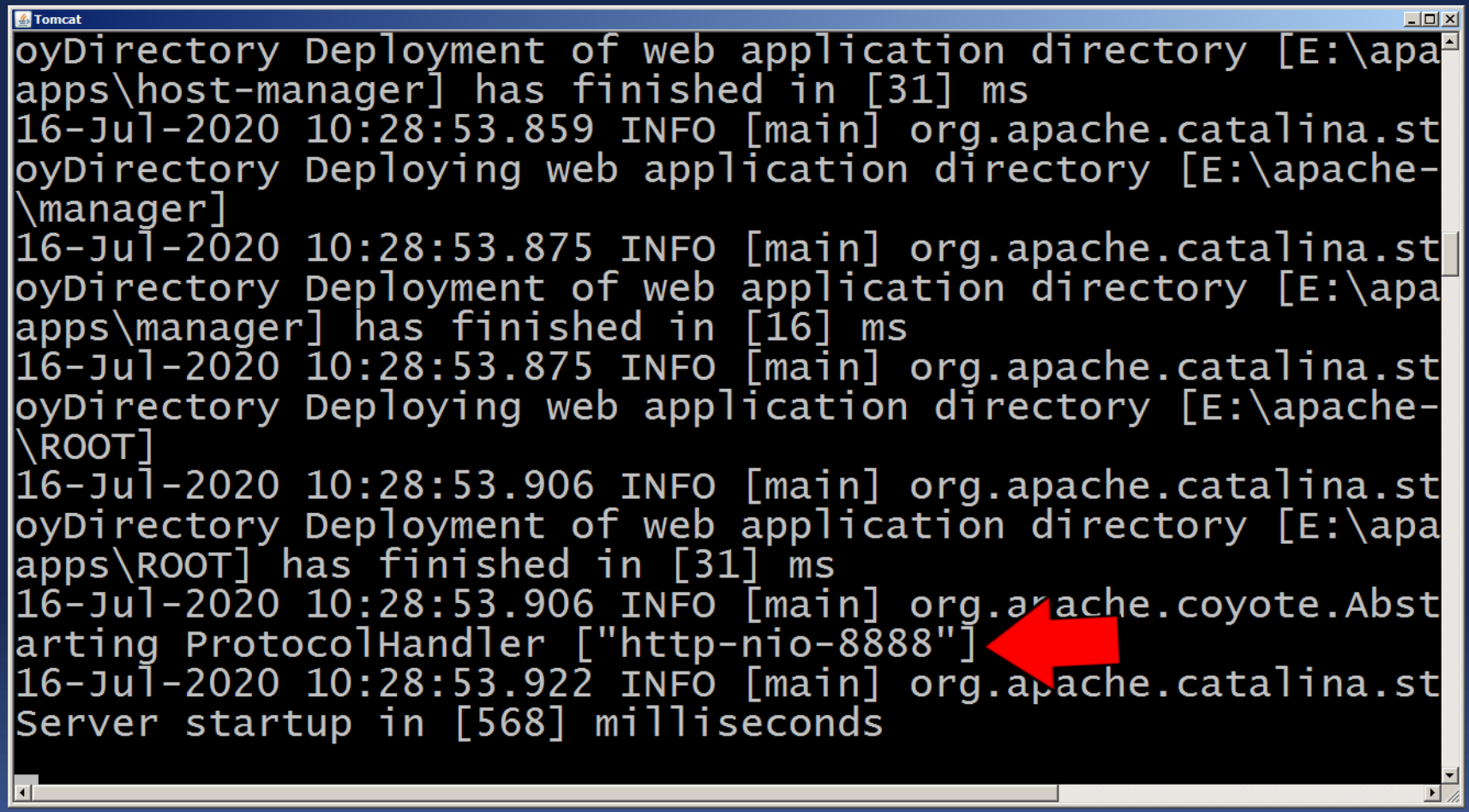
Subindo um Tomcat externo

- No diretório bin, executar - como administrador -- o procedimento **startup.bat**



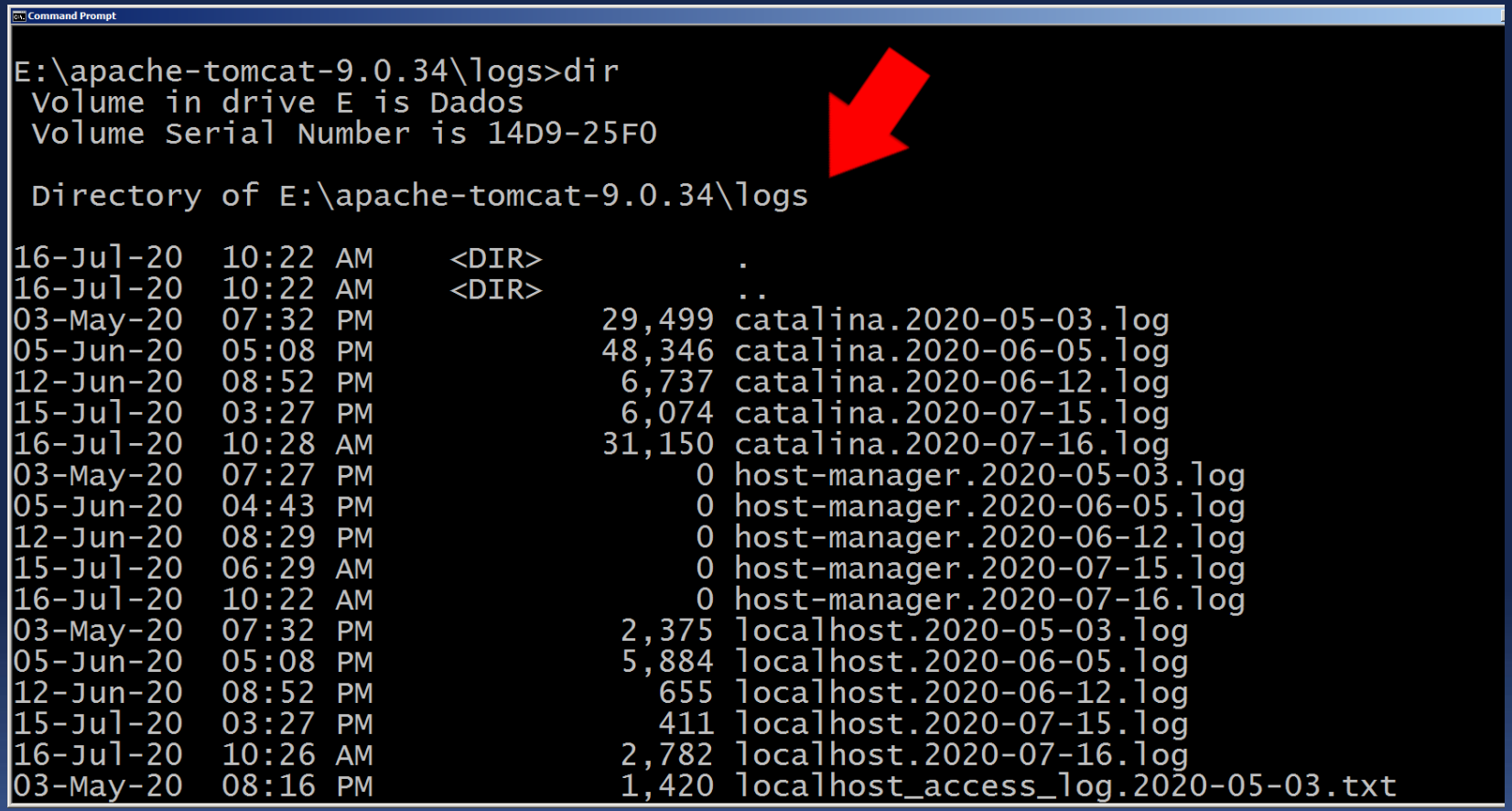
Tomcat rodando na porta 8888

- Tomcat aguardando requisições na porta **8888**



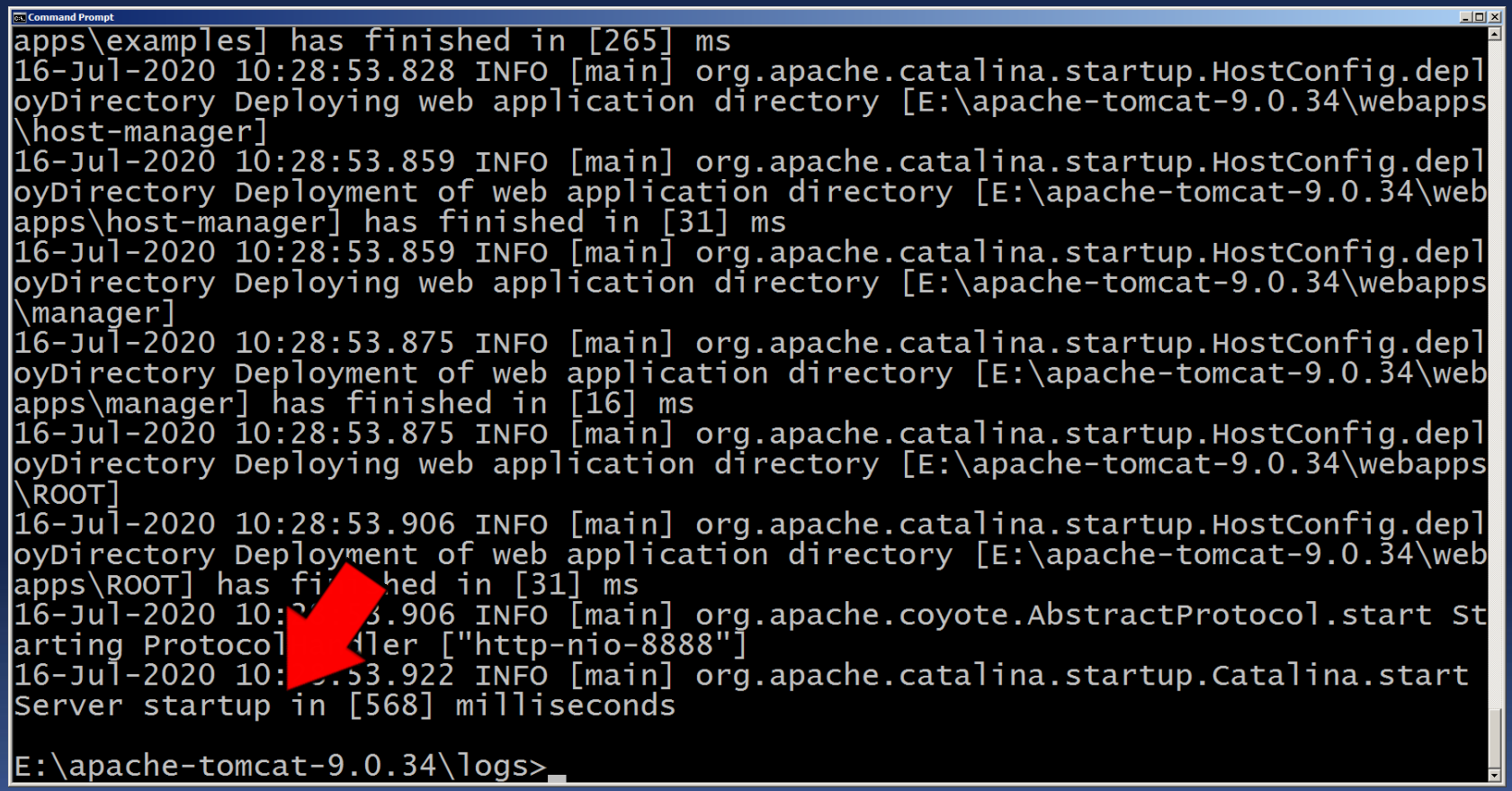
Acessando arquivo de log do Tomcat

- Tomcat registra ocorrências no servidor na pasta **logs**.



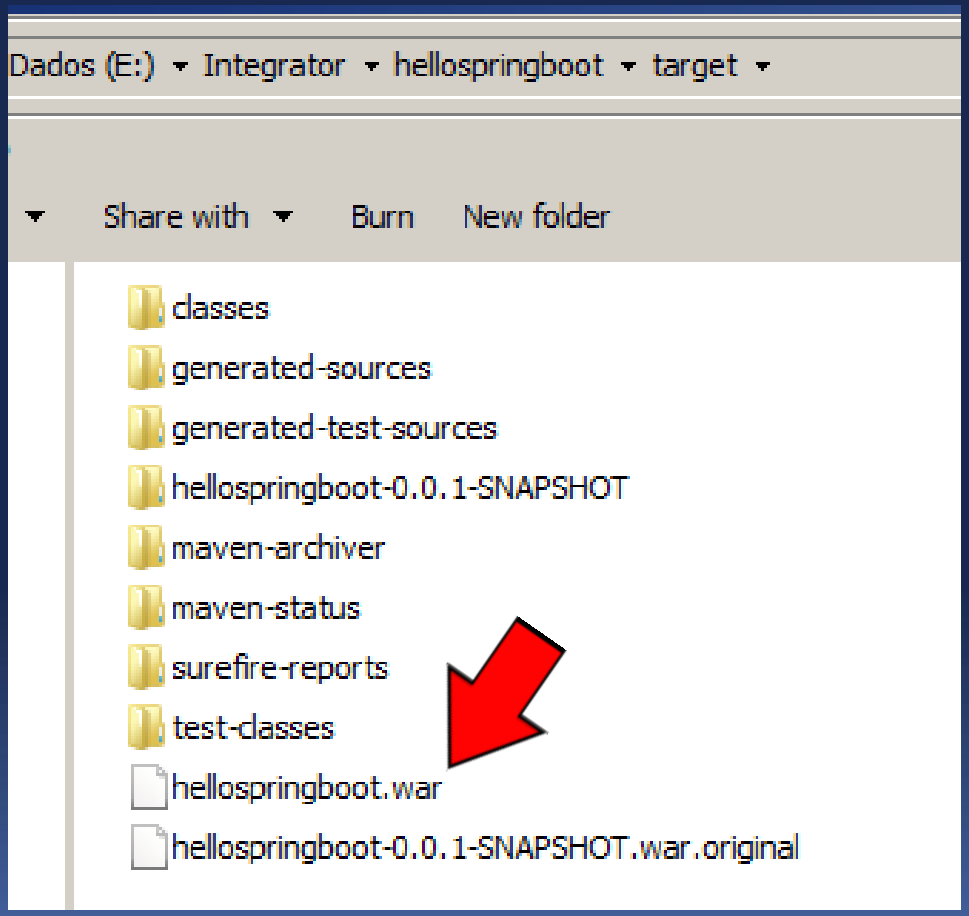
Acessando arquivo de log do Tomcat

- Tomcat registra ocorrências no servidor no arquivo **catalina.2020-07-16.log** na pasta **logs**.



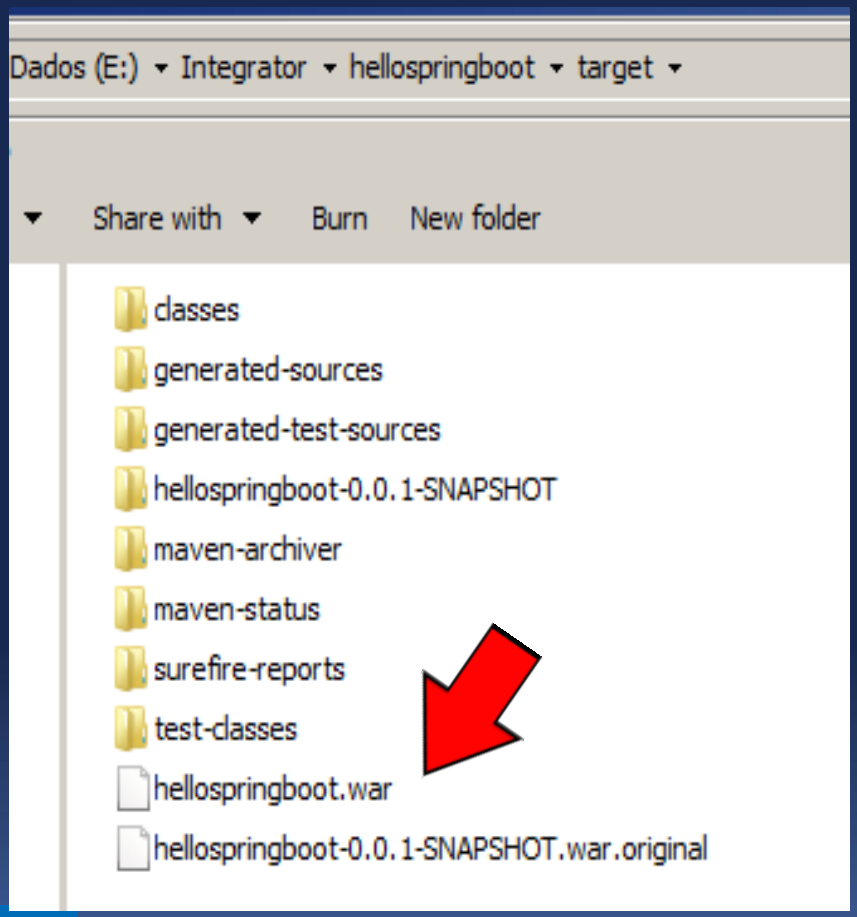
Instalando a aplicação no Tomcat

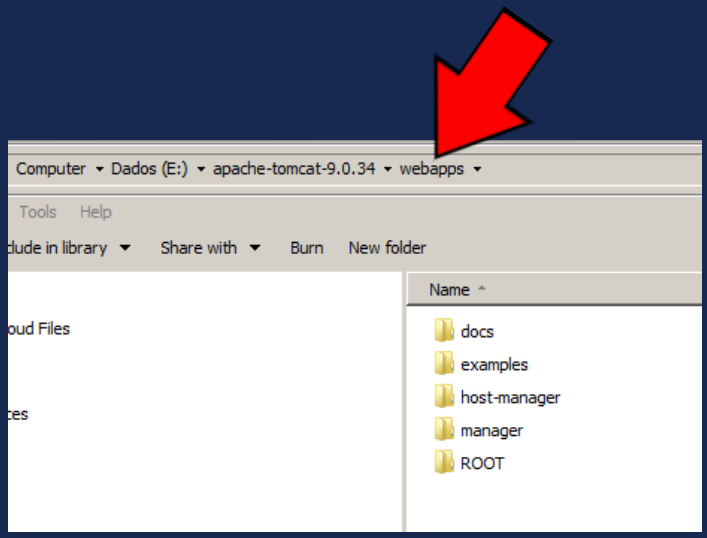
- Para simplificar o manuseio do .war faremos um rename de hellospringboot-0.0.1-SNAPSHOT.war para **hellospringboot.war** na pasta target do projeto



Instalando a aplicação no Tomcat

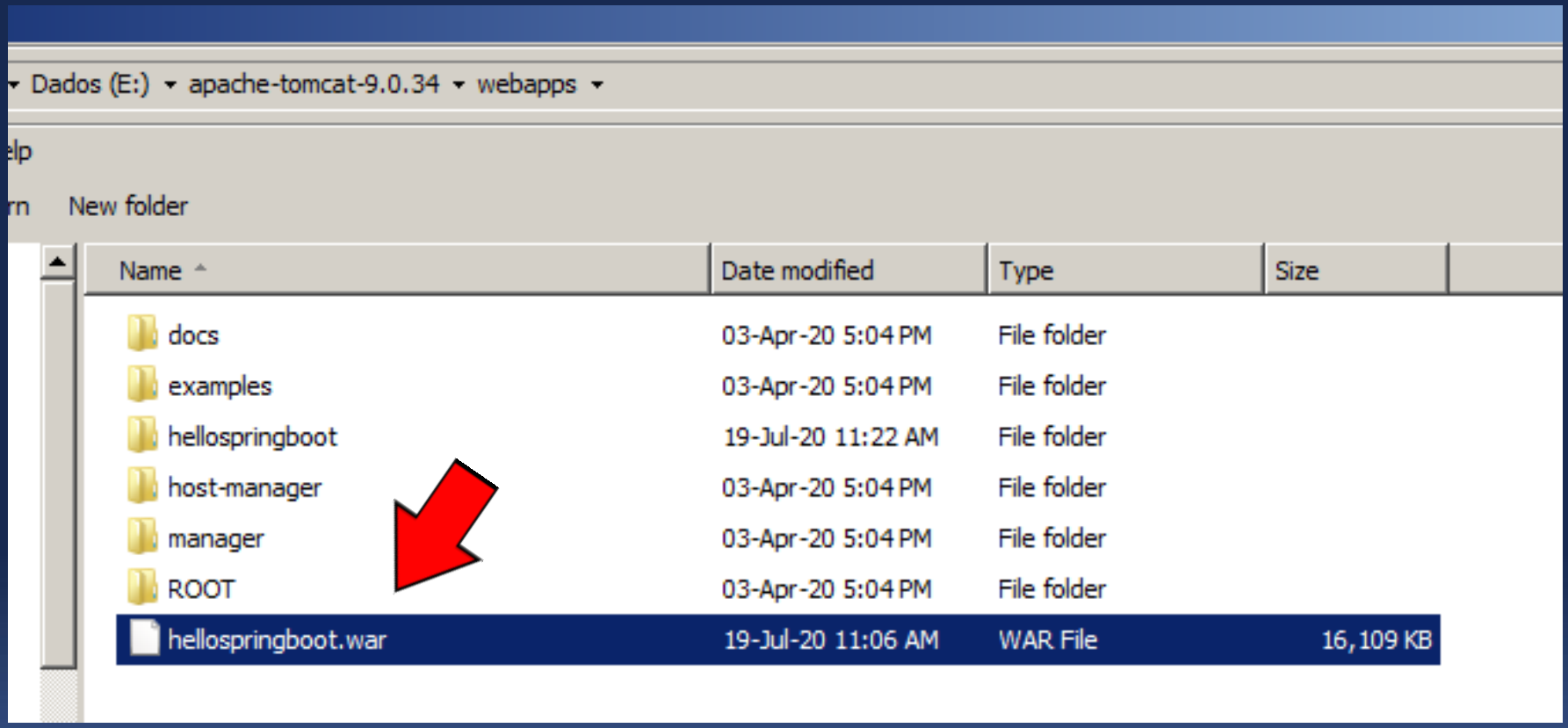
- A nossa aplicação (materializada no arquivo .war) deverá ser copiada para a pasta webapps do Tomcat.





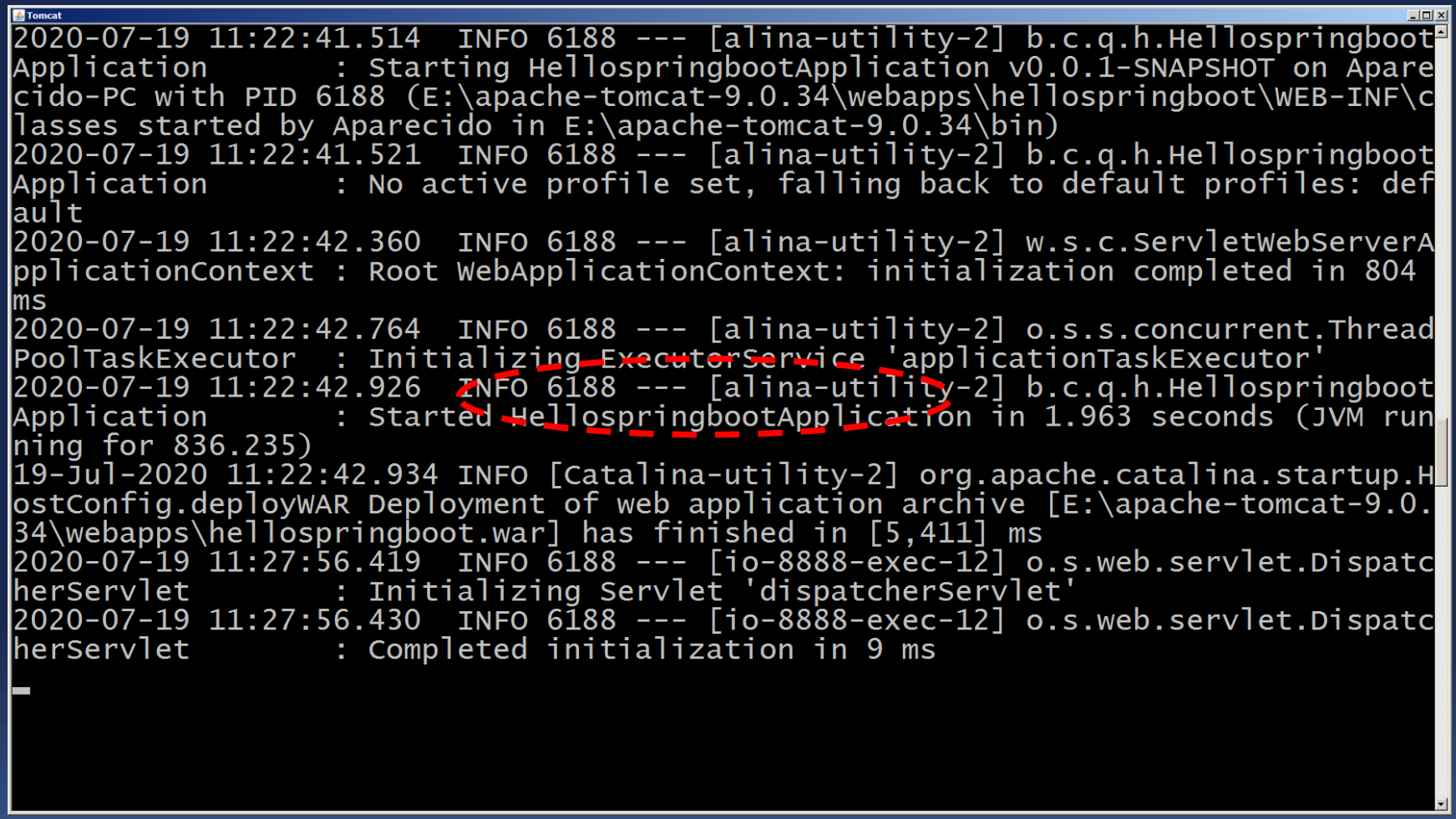
Instalando a aplicação no Tomcat

- Aplicação copiada na pasta **webapps** do Tomcat



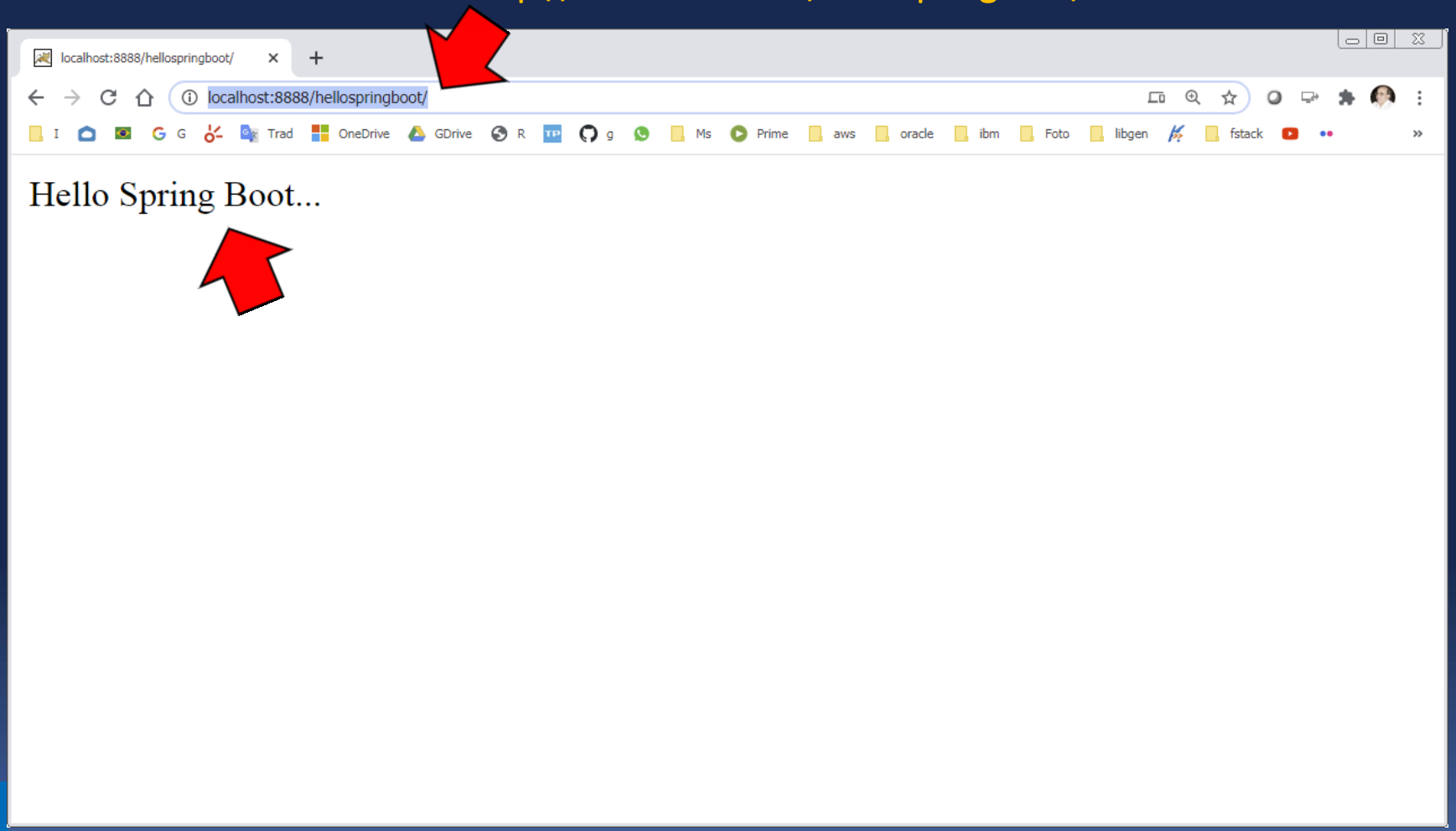
Observando a log do Tomcat

- Visualizando a log do Tomcat após a **cópia** da aplicação (arquivo .war)



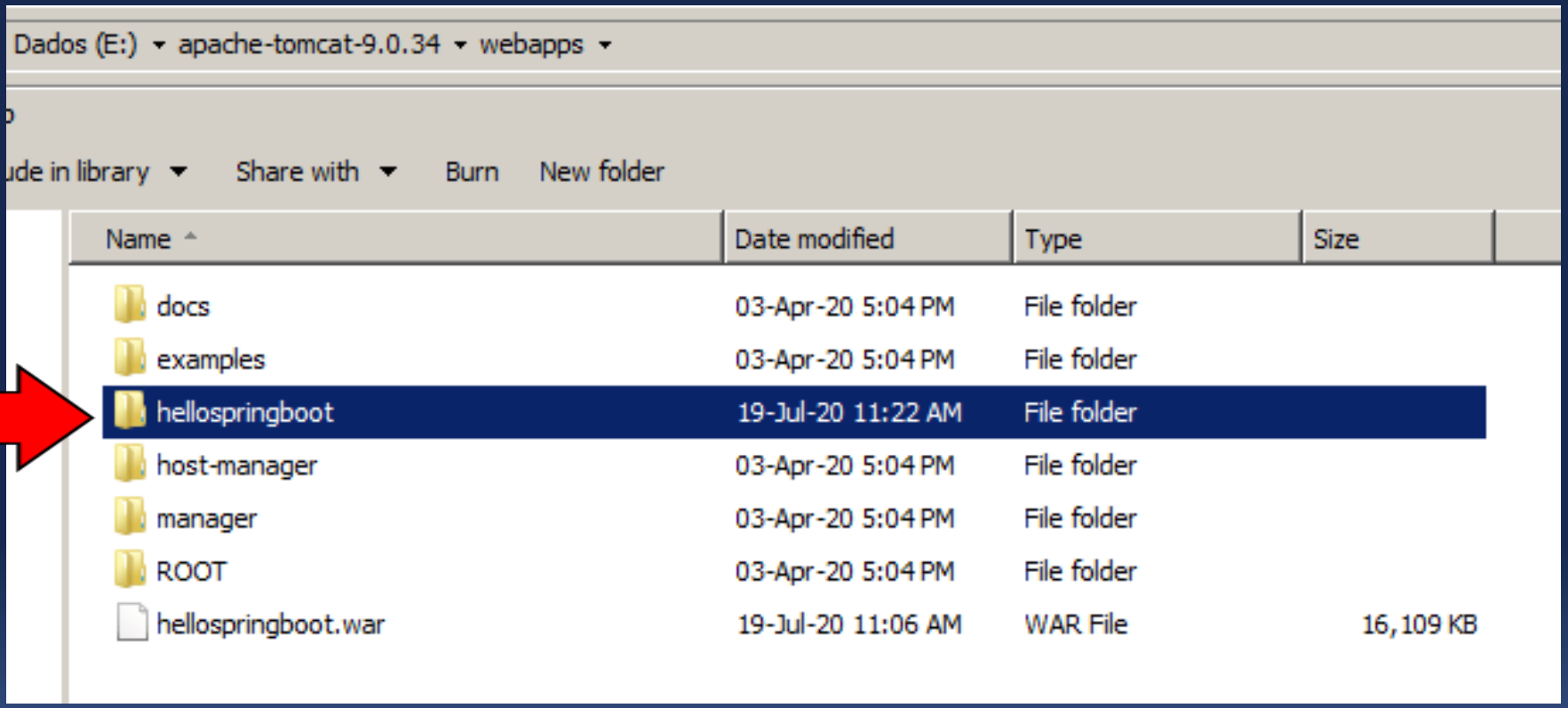
Reexecutando a aplicação no Tomcat externo

- Visualizando a pasta webapps após a cópia do arquivo .war
- A pasta do projeto relativa ao arquivo .war foi criada pelo **Tomcat;**
- No browser entrar com => http://localhost:8888/hellospringboot/



Renomeando o nome da aplicação

- Tomcat criou na pasta webapps um outra pasta chamada hellospringboot que corresponde à pasta do nosso projeto no Tomcat.



Fazendo deploy na nuvem (Integrator.com.br)

Portando a aplicação para o Host Integrator

- Ativando **CPANEL**

DADOS DE ACESSO AO PAINEL DE CONTROLE cPanel:

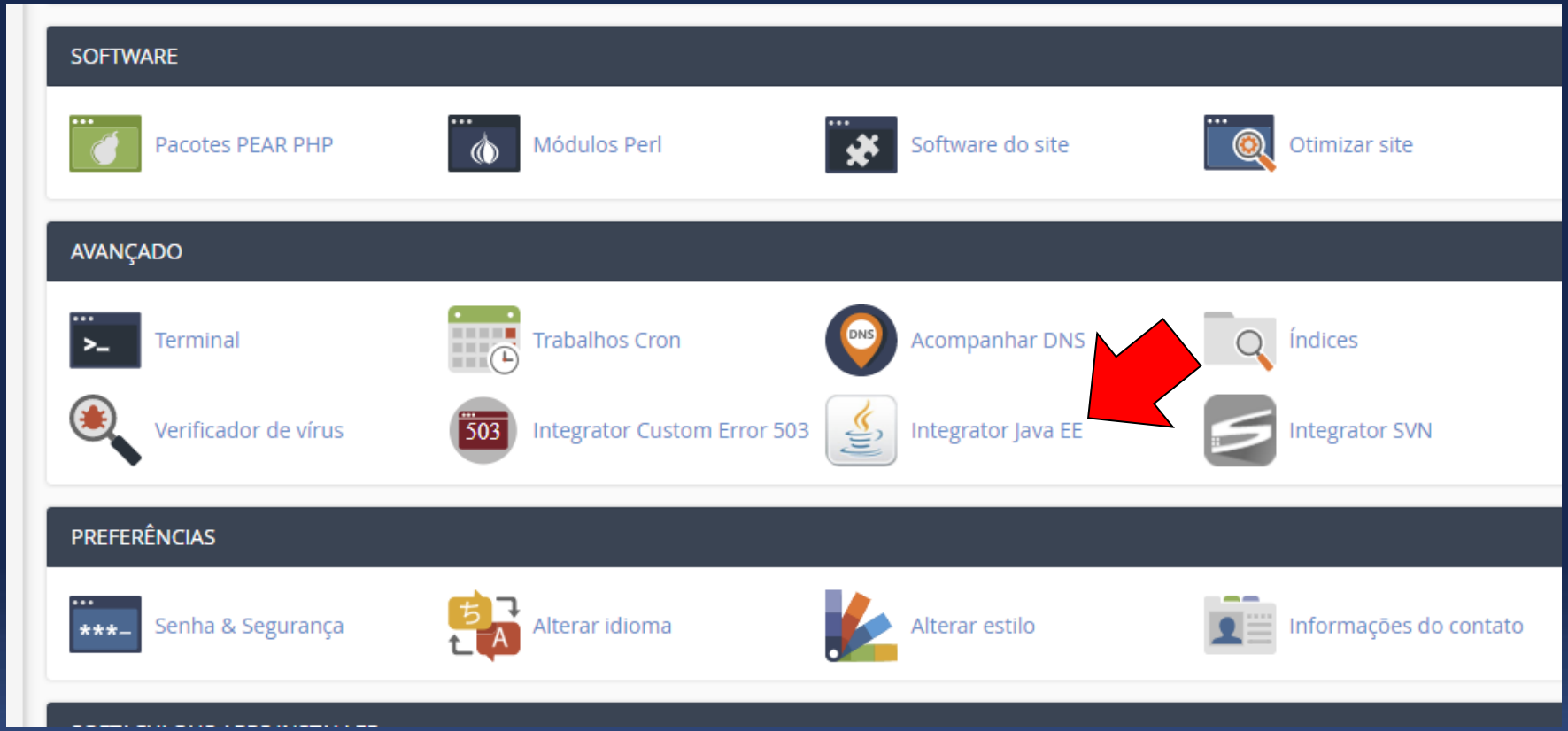
==========

URL: https://209.172.51.58:2083/ *
Usuário: qualitsys
Senha: yqK#-R32Ay7H

Caso o domínio aponte para o servidor da Integrator, acesse:
URL: https://cpanel.qualitsys.net/

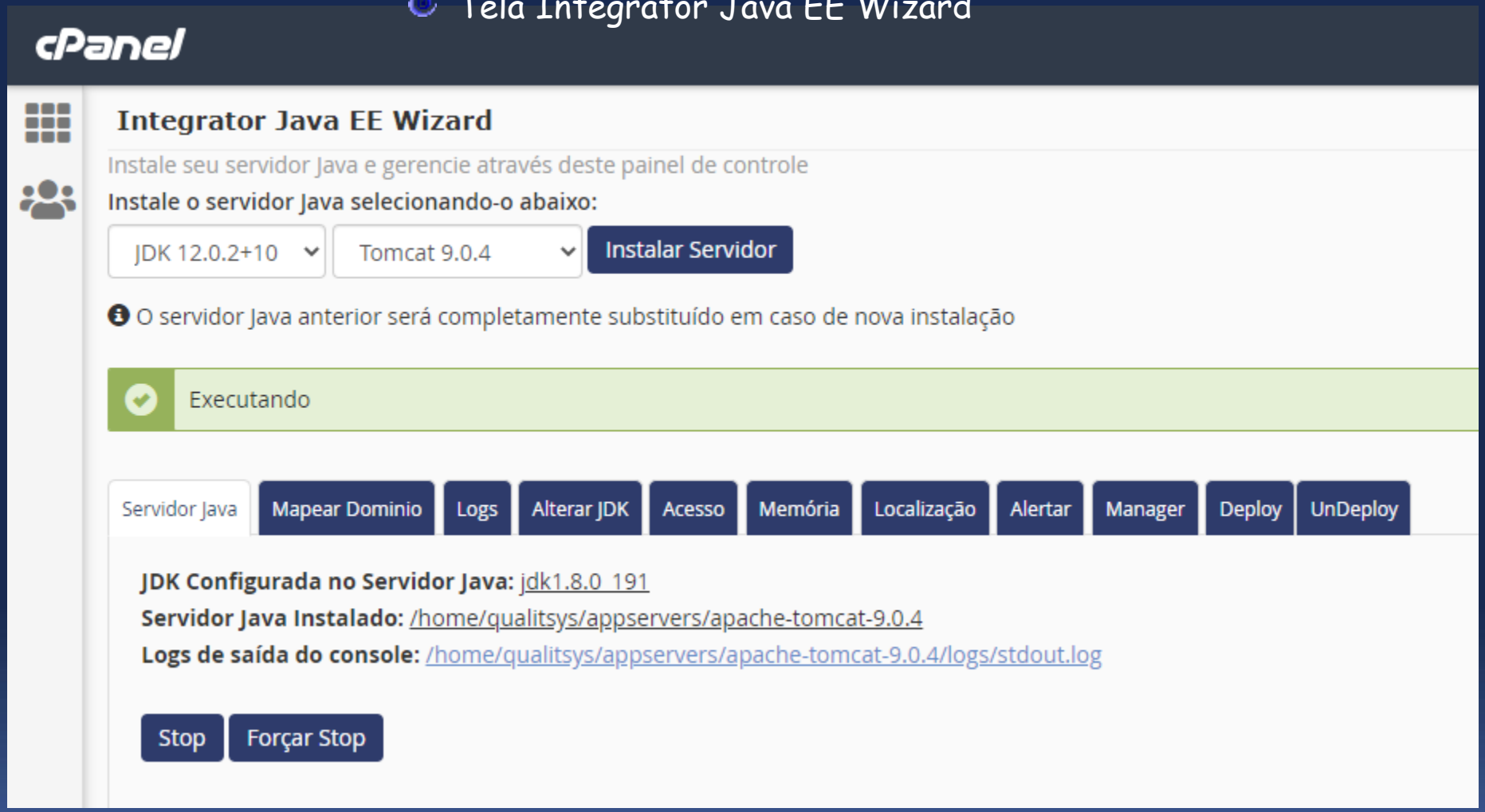
Portando a aplicação para o Host Integrator

- Após ativação do **CPANEL**, acessar o menu AVANÇADO > Integrator Java EE



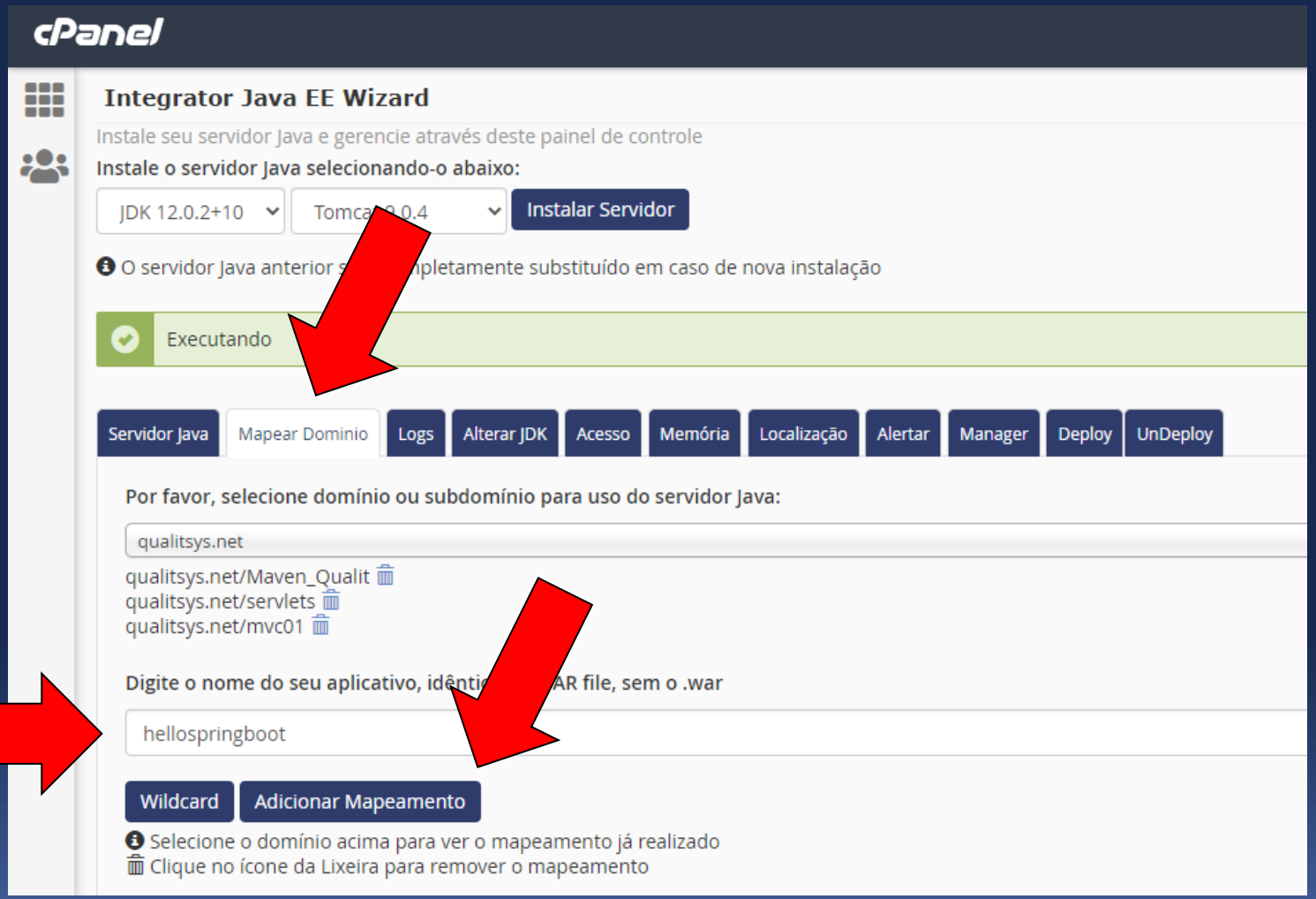
Portando a aplicação para o Host Integrator

- Tela Integrator Java EE Wizard



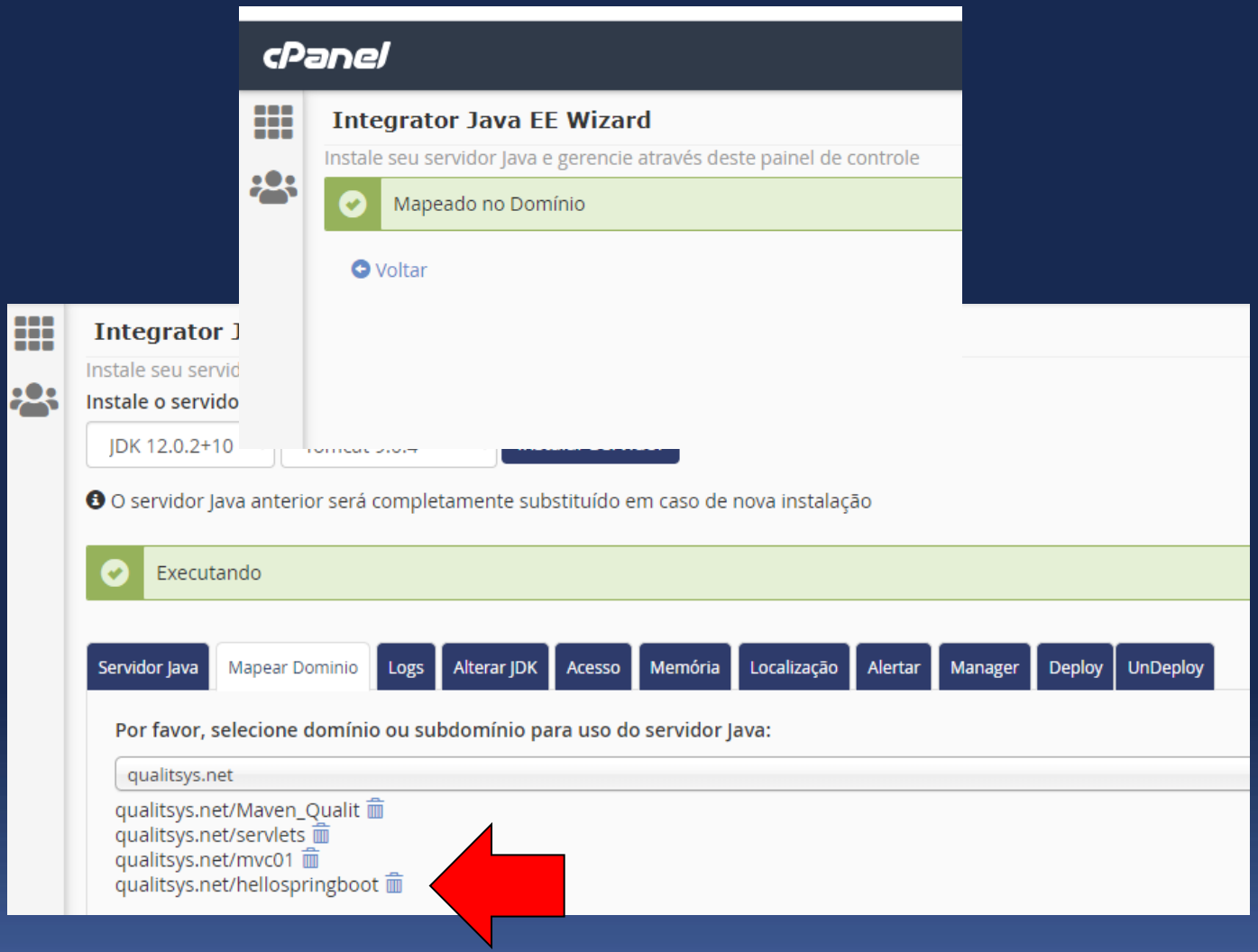
Mapeamento de Domínio

- No cPanel > Integrator Java EE Wizard > Mapear Dominio > Nome do aplicativo
- Clicar em Adicionar Mapeamento



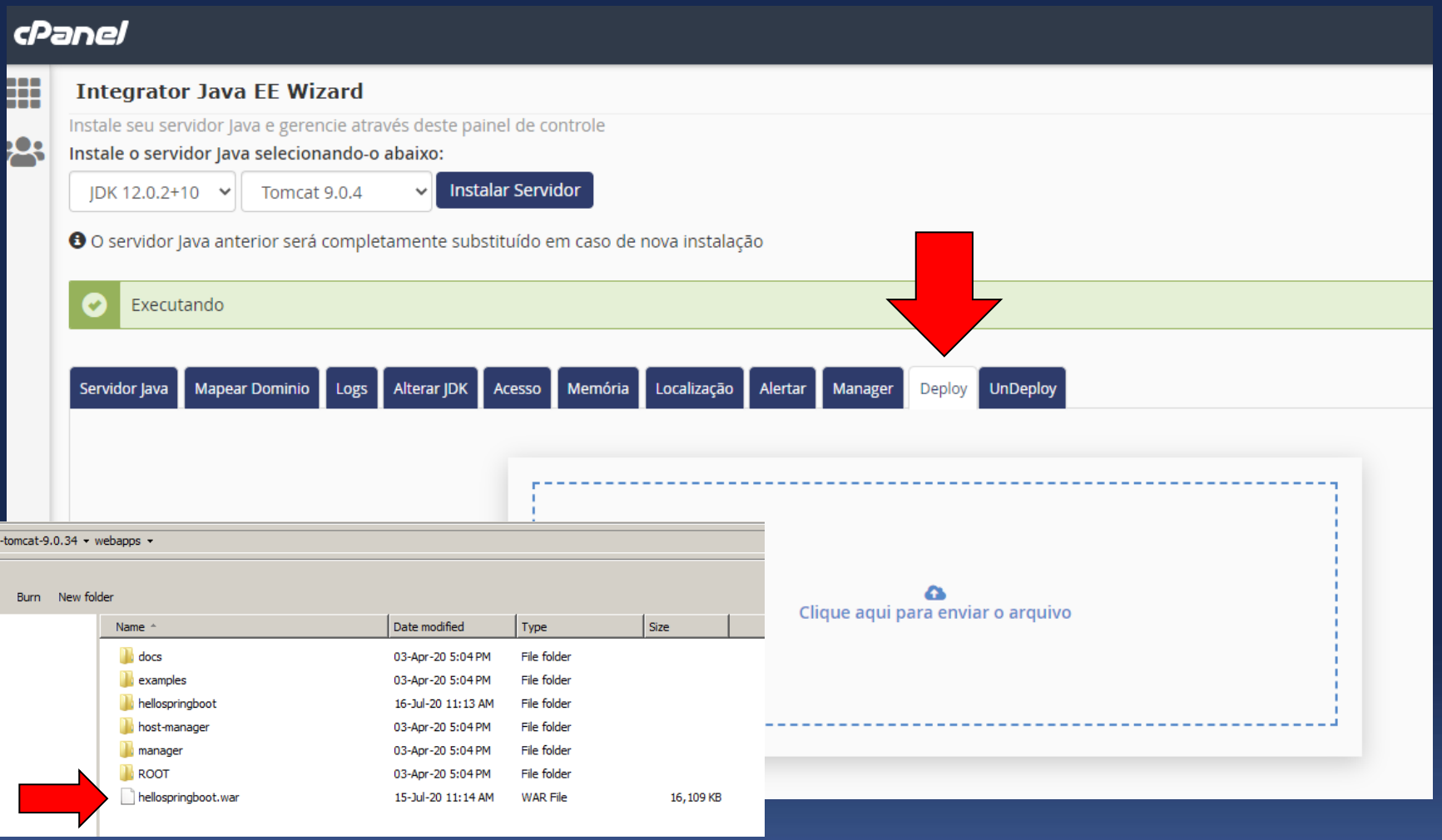
Mapeamento de Domínio

- Mapeamento realizado



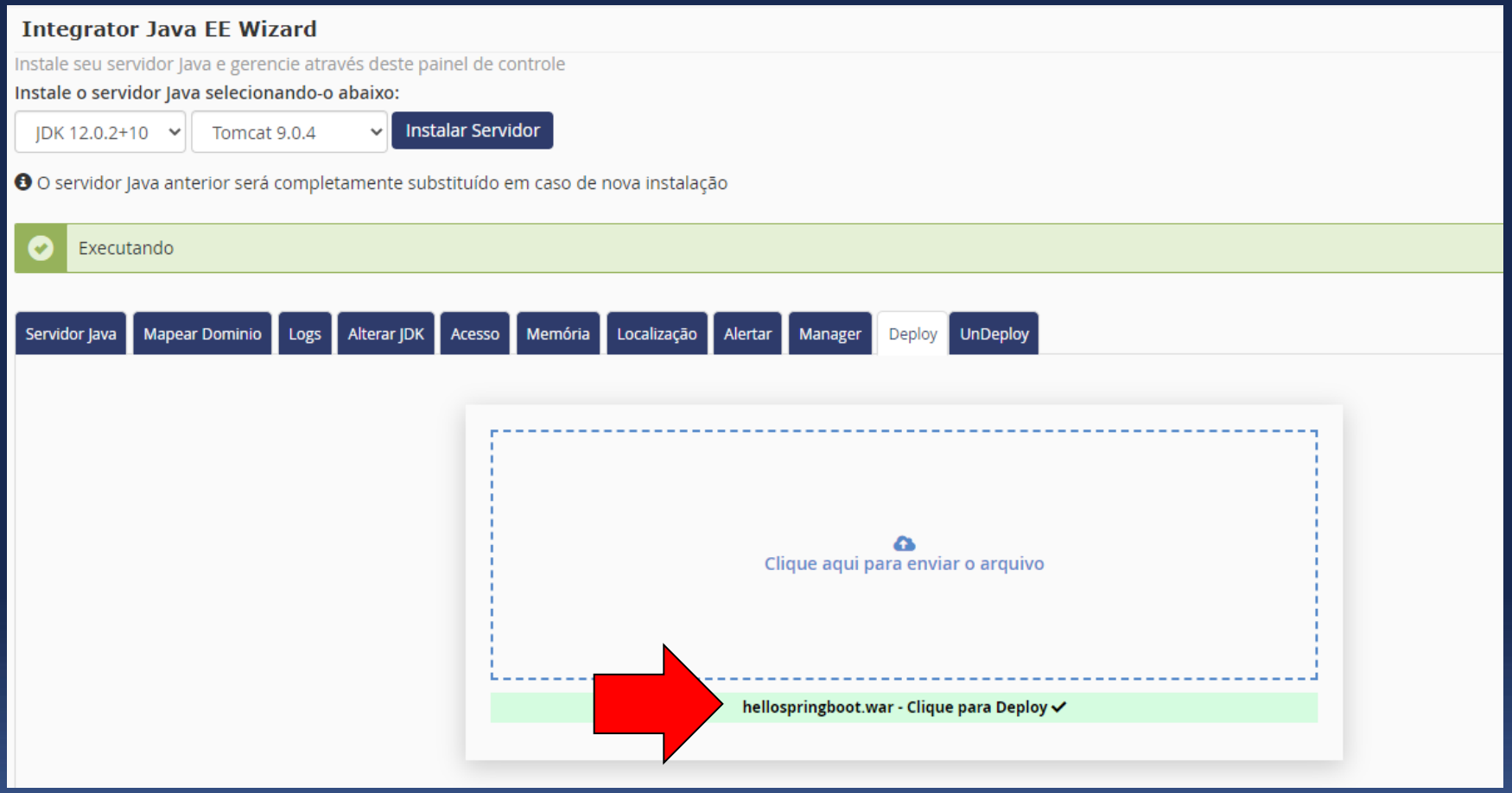
Deploy do Aplicativo

- Upload arquivo .war



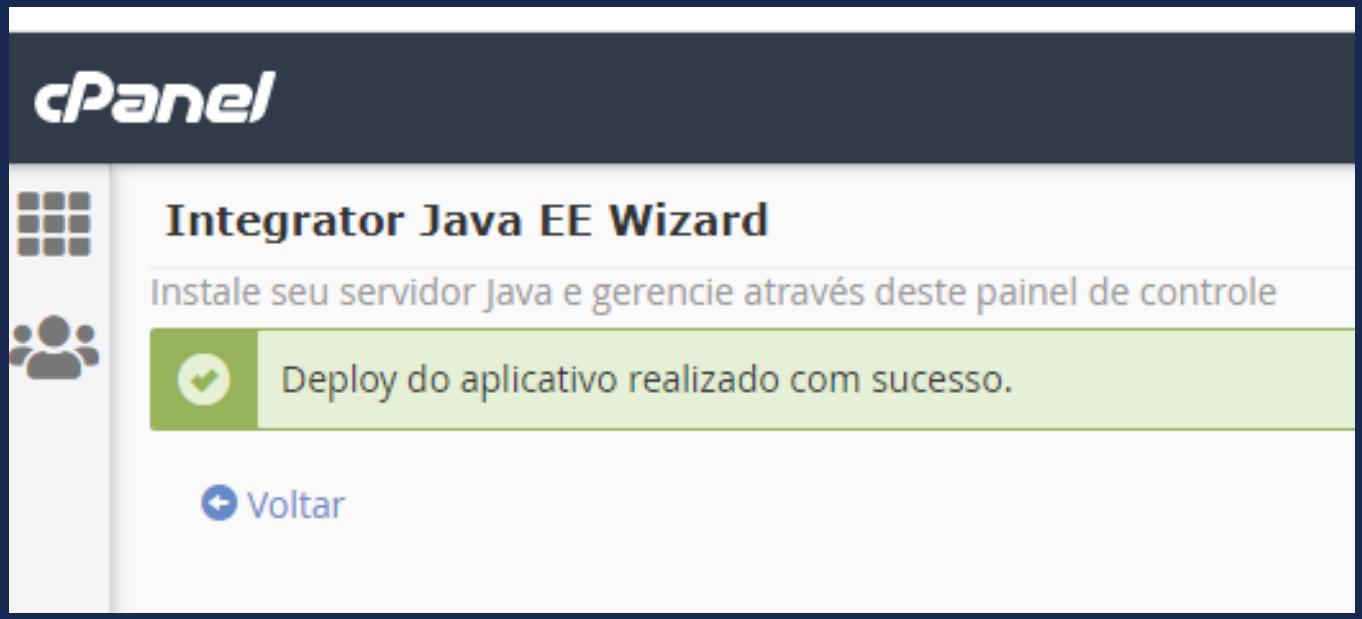
Deploy do Aplicativo

- Clique para Deploy



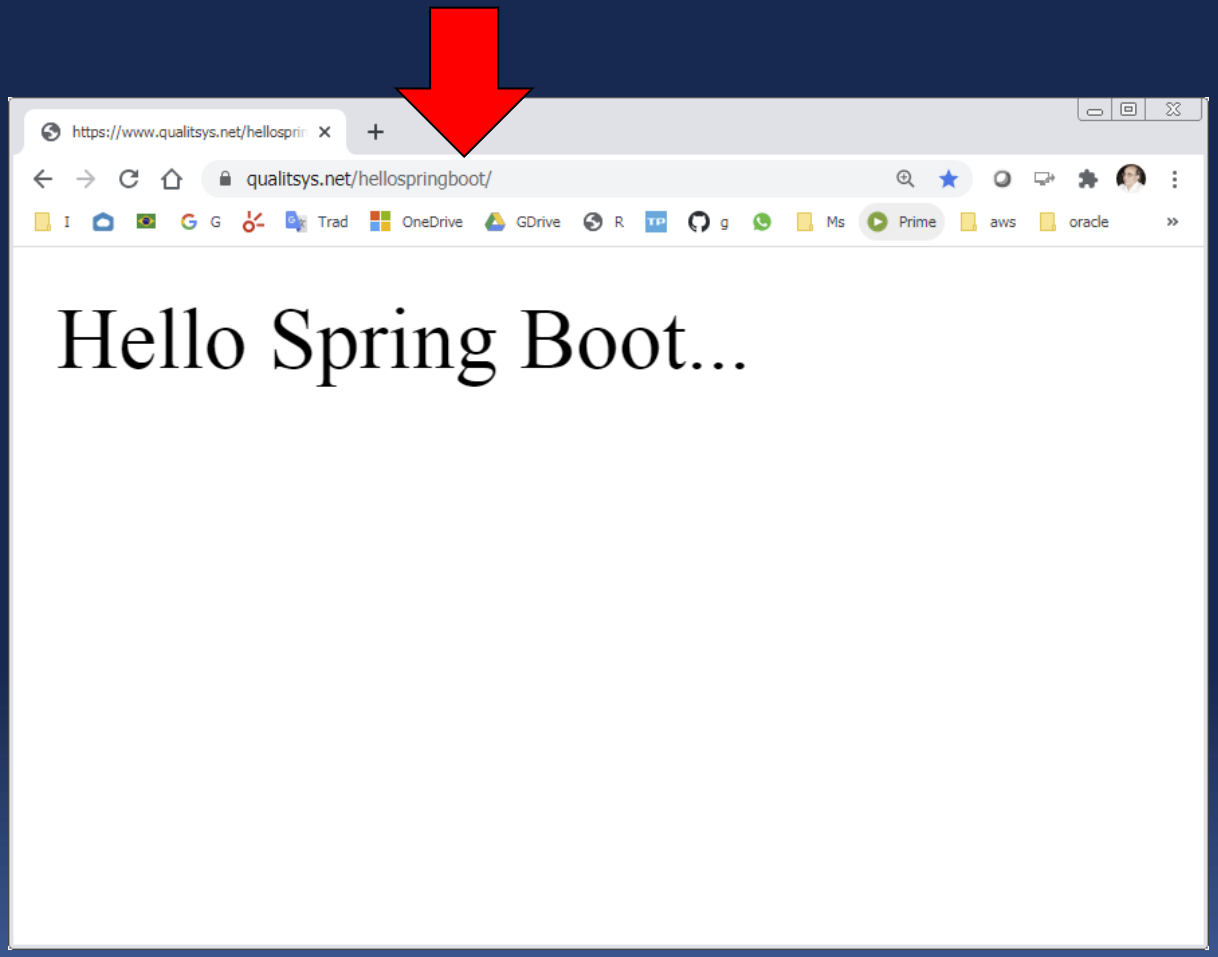
Deploy do Aplicativo

- Deploy realizado com sucesso!



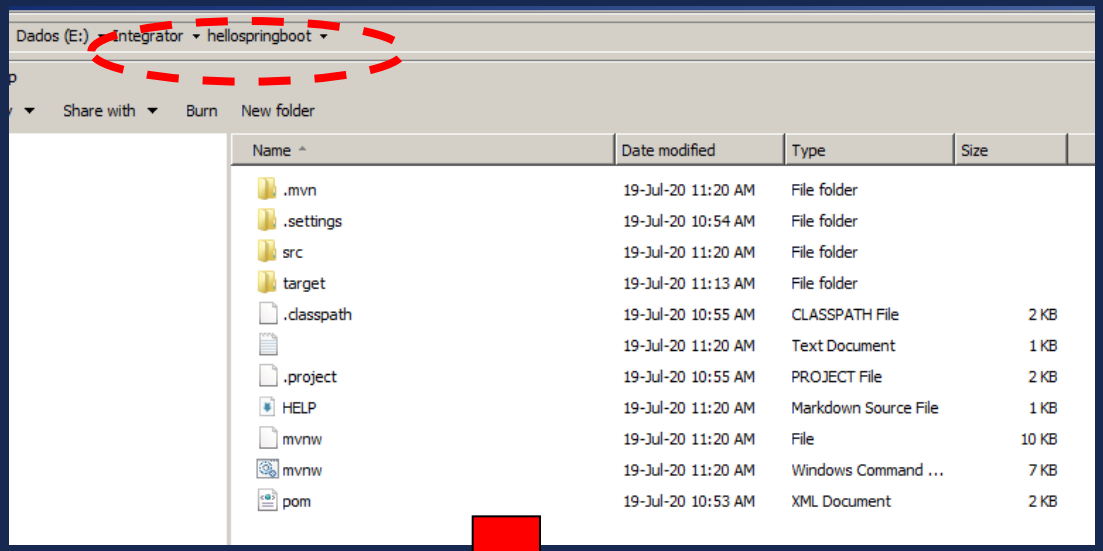
Testando aplicação na nuvem

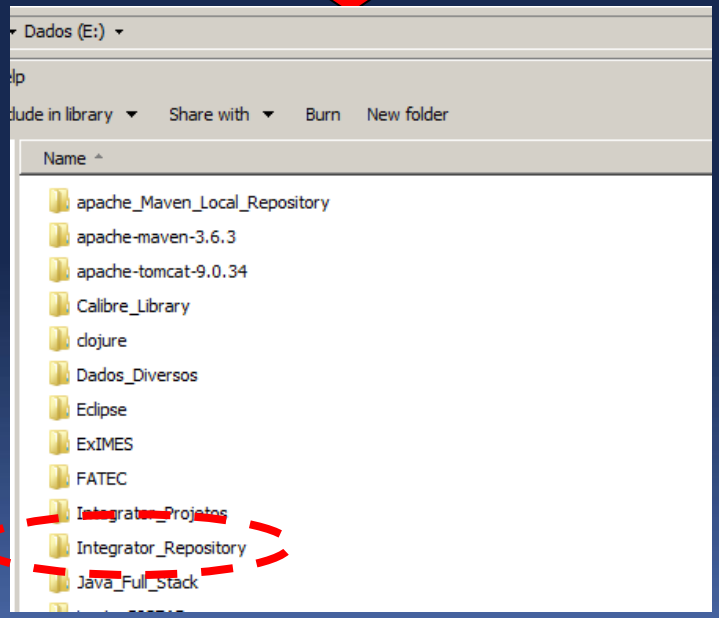
- https://www.qualitsys.net/hellospringboot/



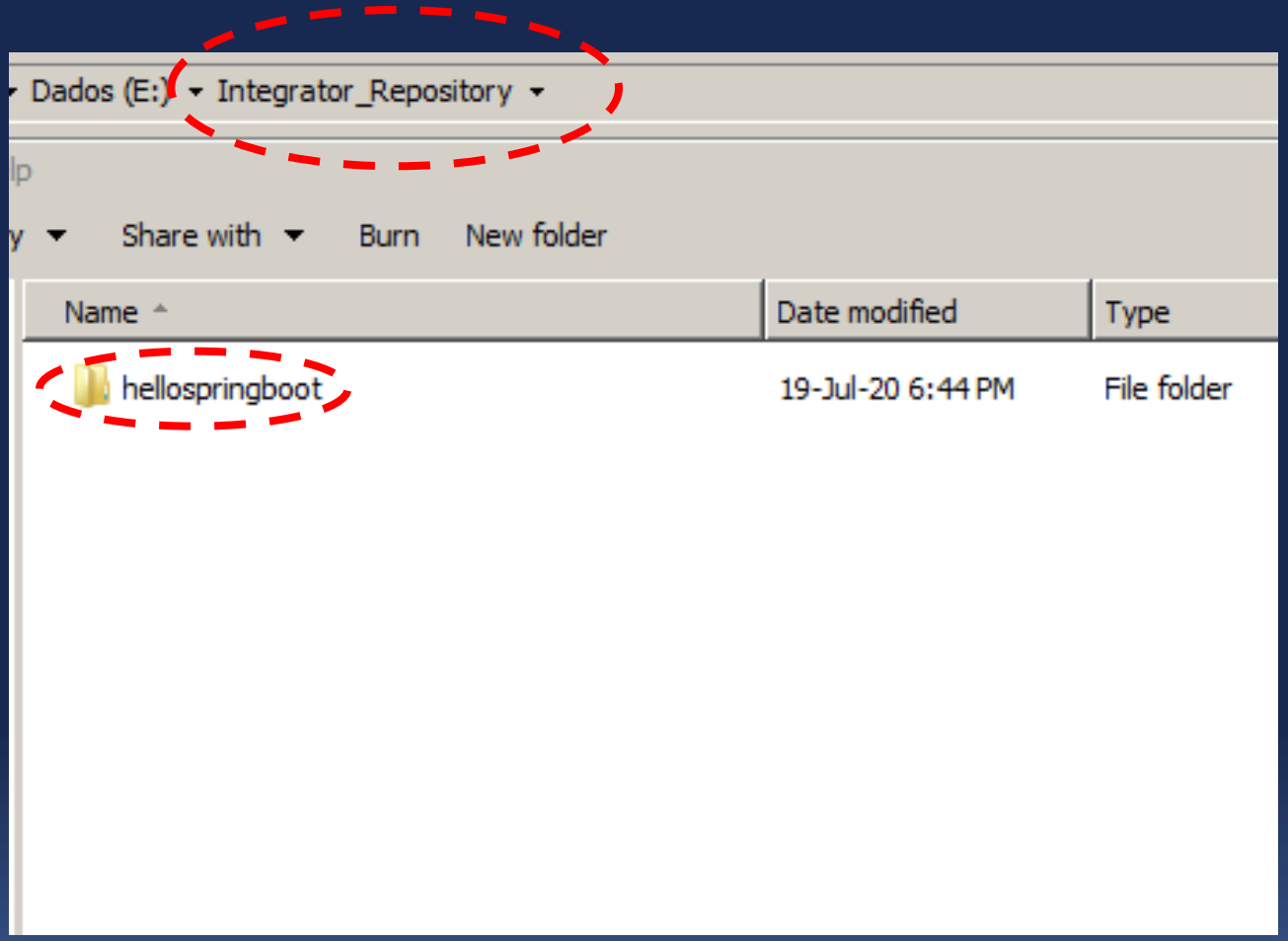
Salvando projeto no github

Salvando projeto no Repositório git local

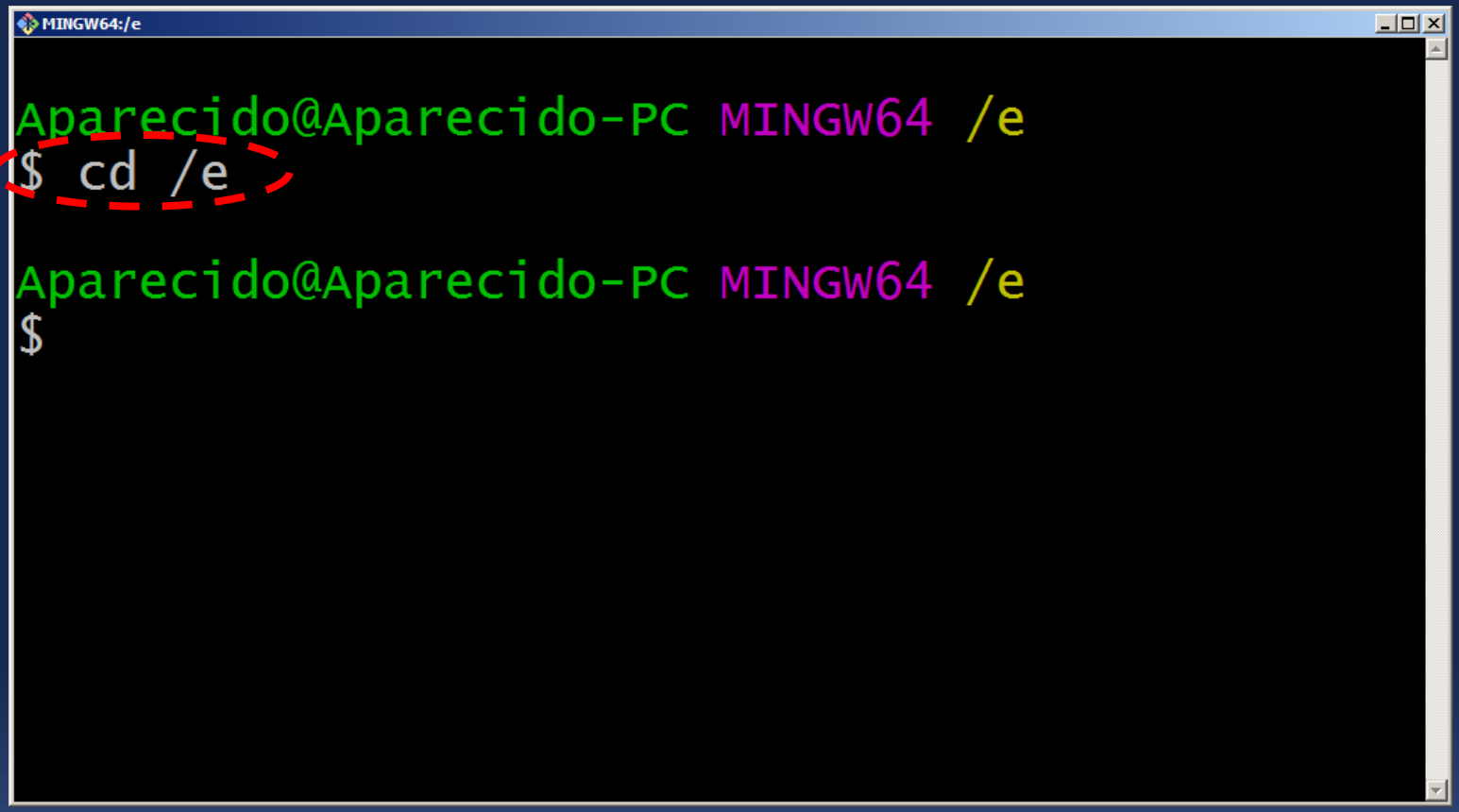




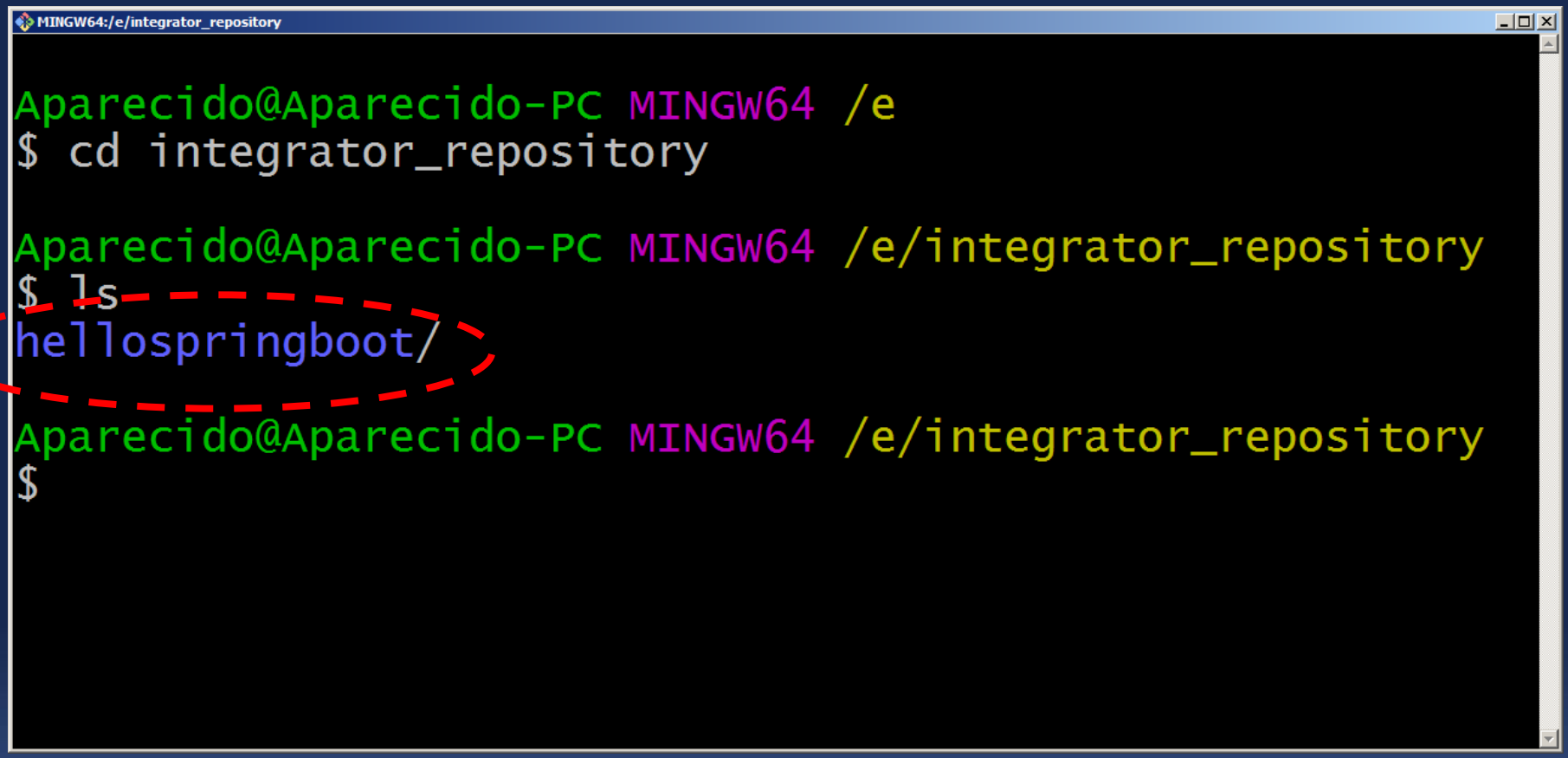
Projeto salvo no repositório local



Acessando o repositorio git local c/gitbash

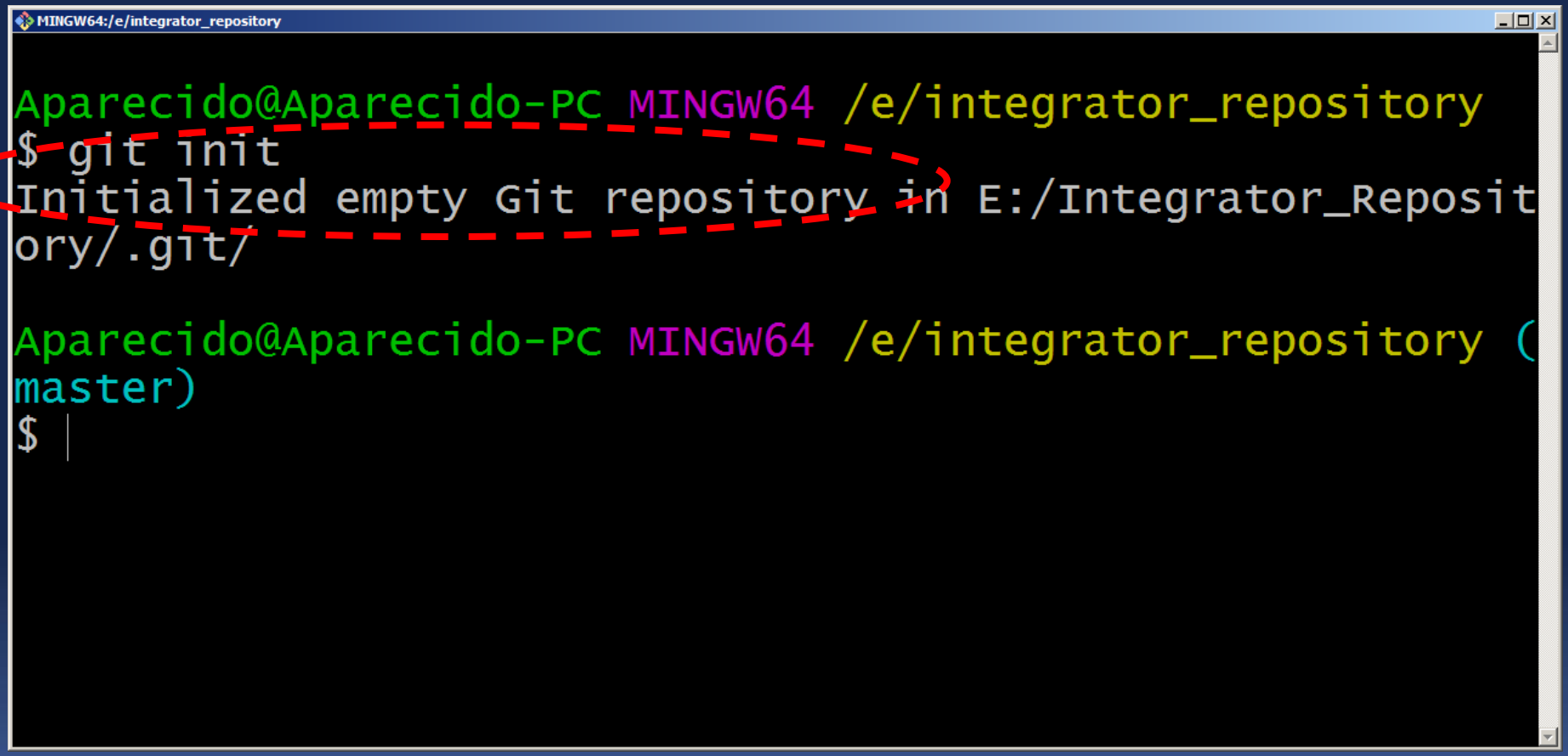


Acessando o repositorio git local c/gitbash



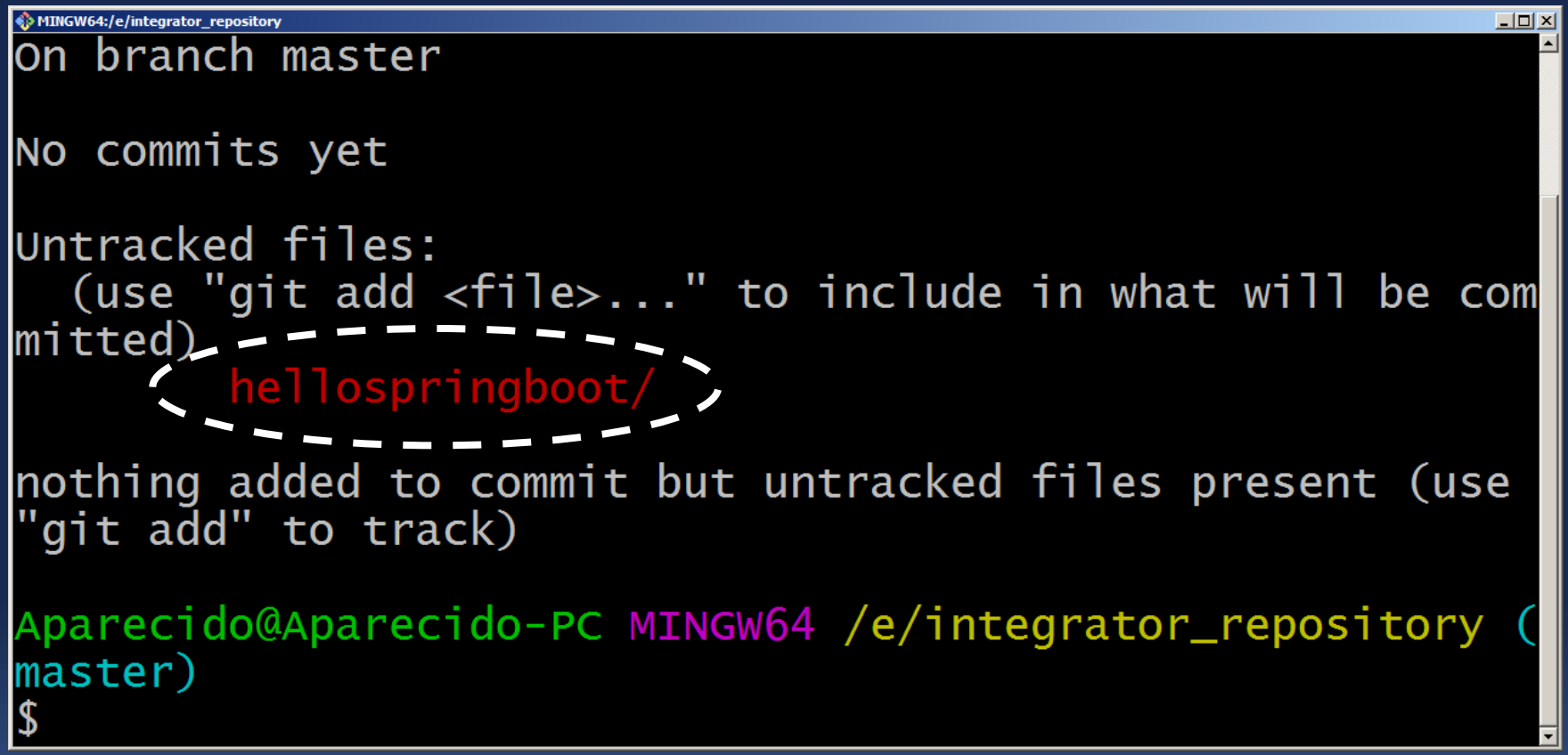
Monitoração do repositório pelo git

git init



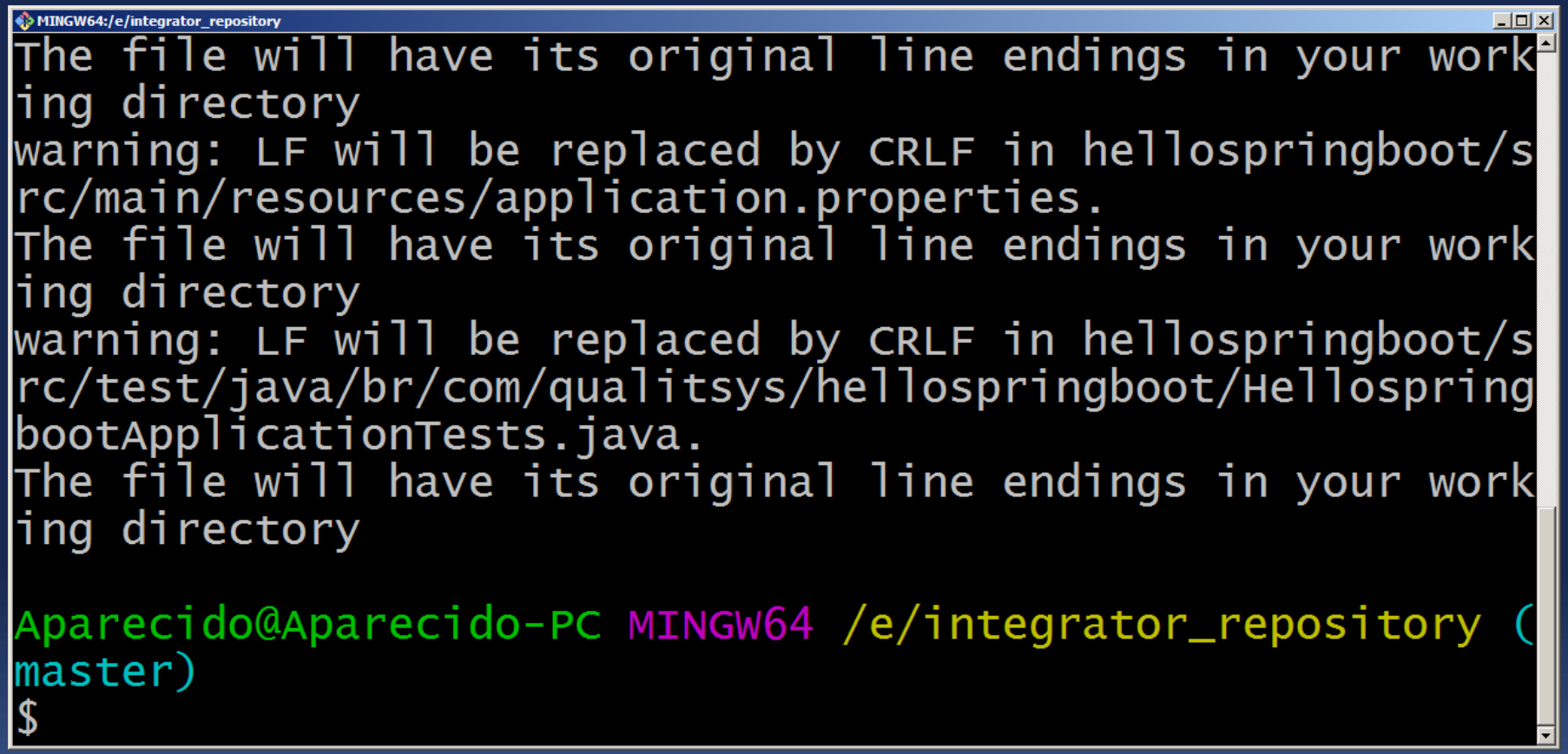
git status

- A cor vermelha indica que a pasta do nosso projeto (hellospringboot) ainda não está sendo monitorada pelo git.



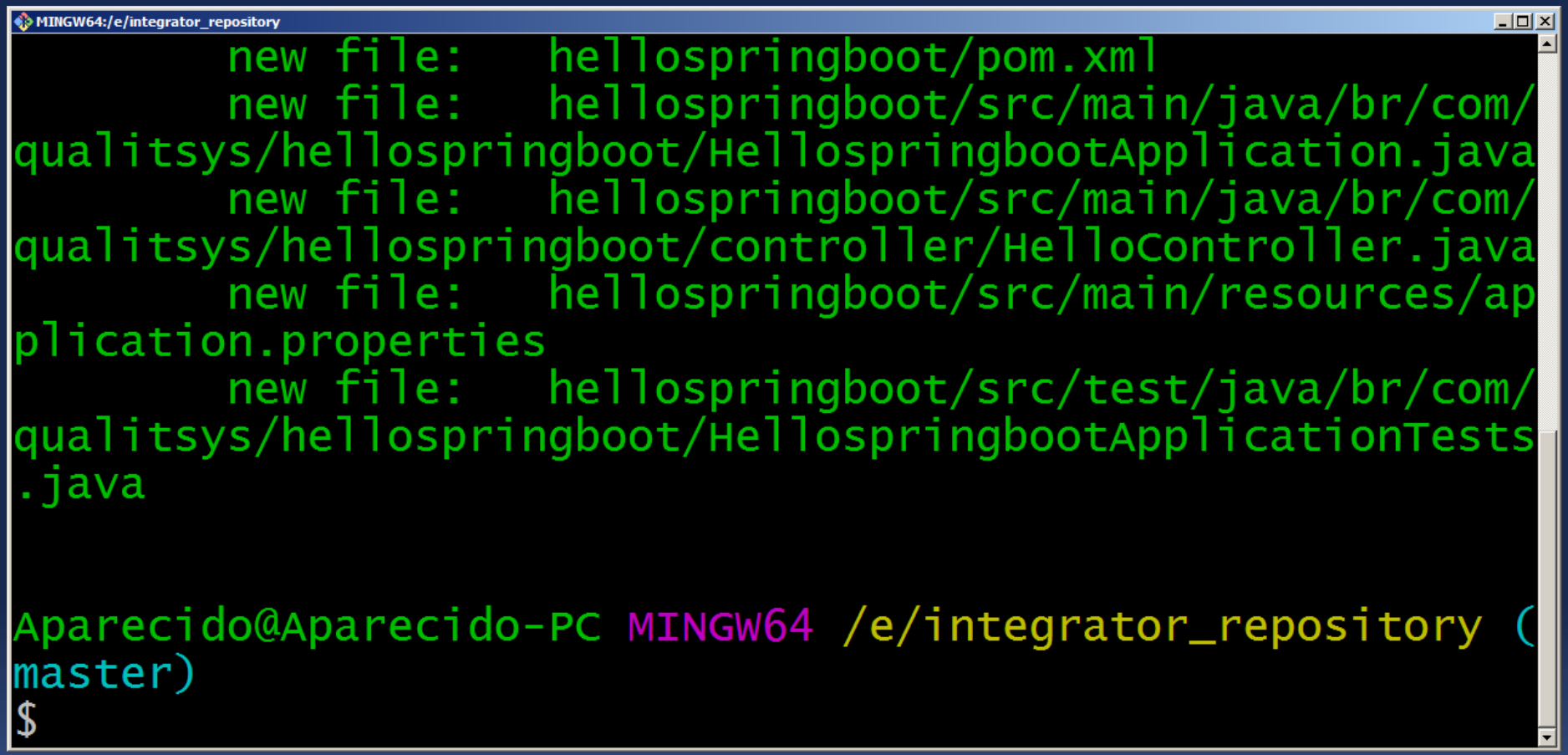
git add .

- Esse comando irá incluir o projeto (todos os arquivos) para serem monitorados pelo git

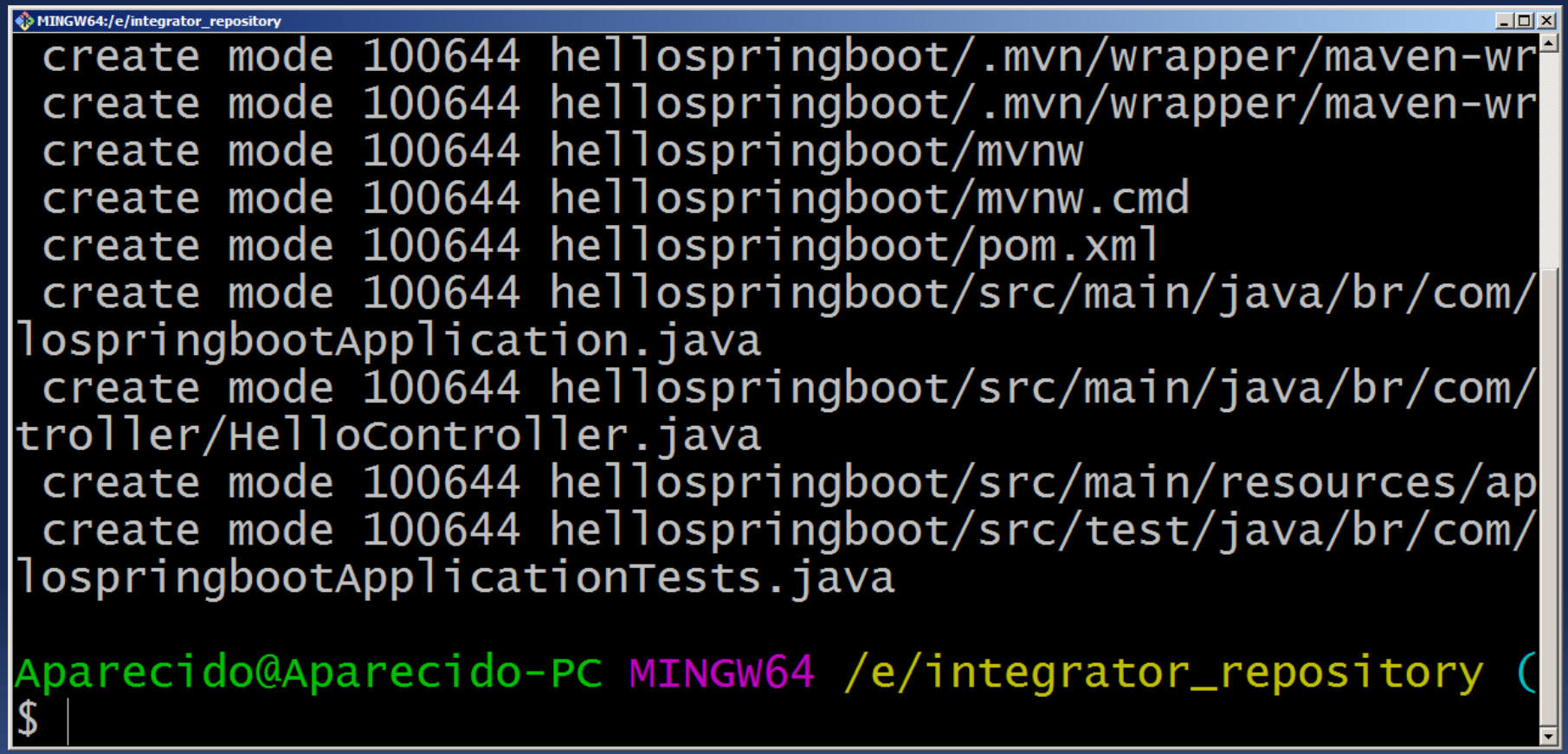


git status

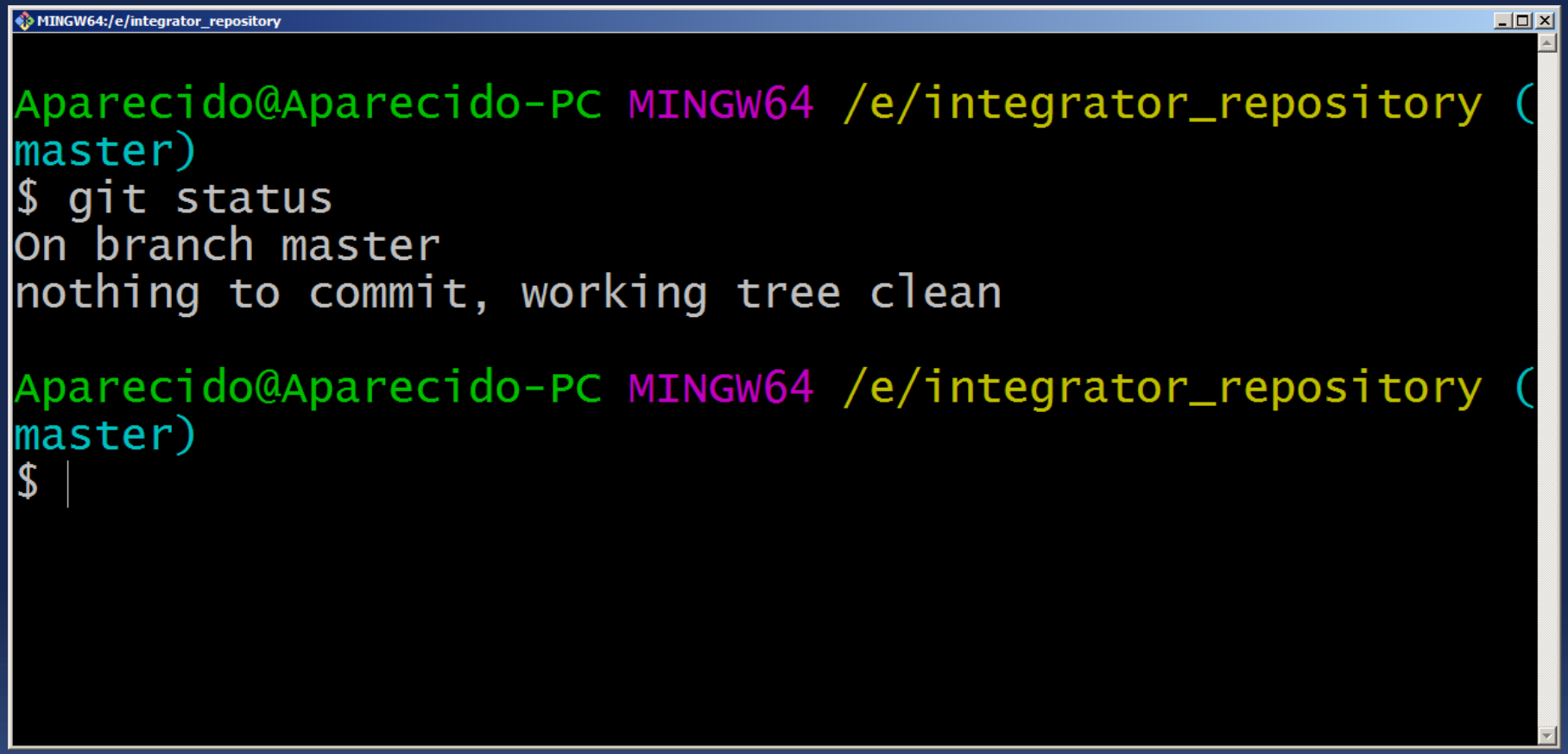
- Os arquivos com a cor **verde** indicam que ainda não foram comitados.



git commit -m "v1.0.0 - 19/07/2020"



git status


MINGW64:/e/integrator_repository
Aparecido@Aparecido-PC MINGW64 /e/integrator_repository (master)
$ git status
On branch master
nothing to commit, working tree clean
Aparecido@Aparecido-PC MINGW64 /e/integrator_repository (master)
$

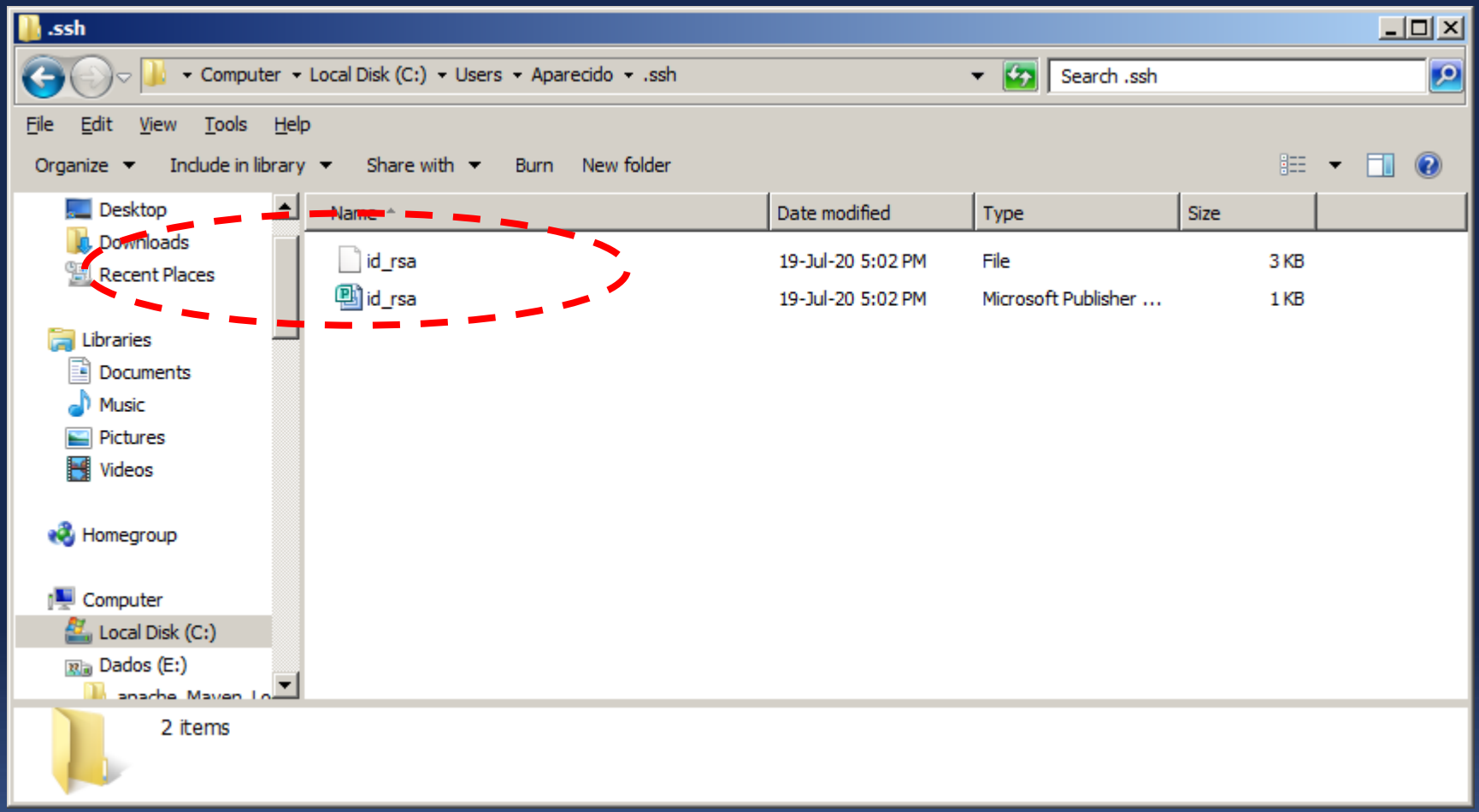
ssh-keygen (Chave de segurança)

- A chave de segurança é gerada para se acessar o github a partir do repositório local;
- As chaves são criadas em 2 arquivos na pasta do usuário: c:/users/Aparecido/.ssh/id_rsa e c:/users/Aparecido/.ssh/id_pub

```
MINGW64:/e/integrator_repository
Your identification has been saved in /c/Users/Aparecido/
.ssh/id_rsa
Your public key has been saved in /c/Users/Aparecido/.ssh
/id_rsa.pub
The key fingerprint is:
SHA256:OvFl8nQYnW/Z7l6OqnpXOPJ7qZ4lBoRGk3zHLEJp1SY Aparec
ido@Aparecido-PC
The key's randomart image is:
+---[RSA 3072]----+
|        o++.+    |
|       .*oE.*.   |
|       .ooo=o    |
|       . . o . o |
|      . S * ..+ .|
|       + *.oo... |
|      o . .ooo..o|
|       .  ..o+o+.|
|         .o.+*=..+|
+----[SHA256]-----+

Aparecido@Aparecido-PC MINGW64 /e/integrator_repository (
master)
$
```

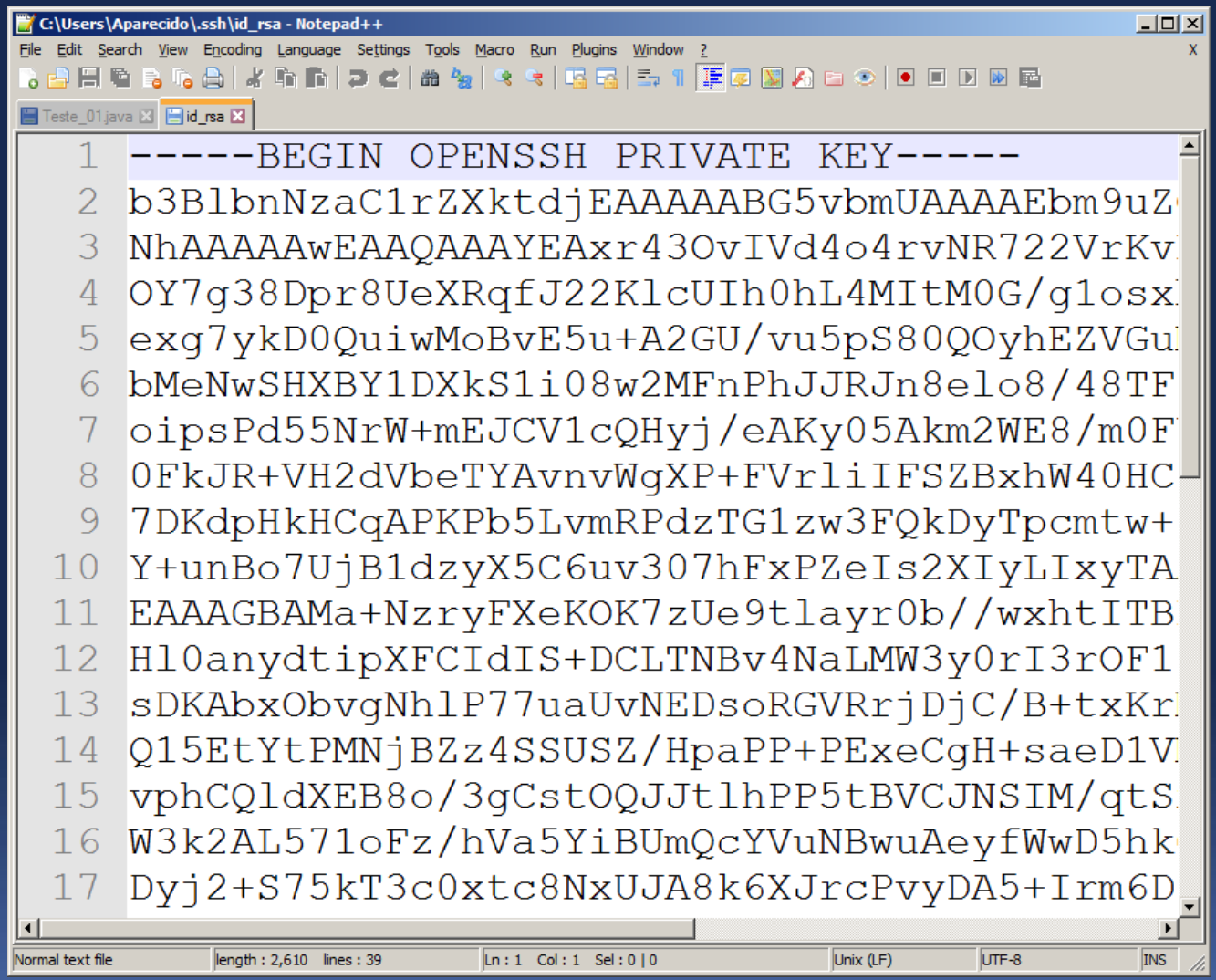
ssh-keygen (Chave de segurança)

c:/users/Aparecido/.ssh/id_rsa e c:/users/Aparecido/.ssh/id_pub



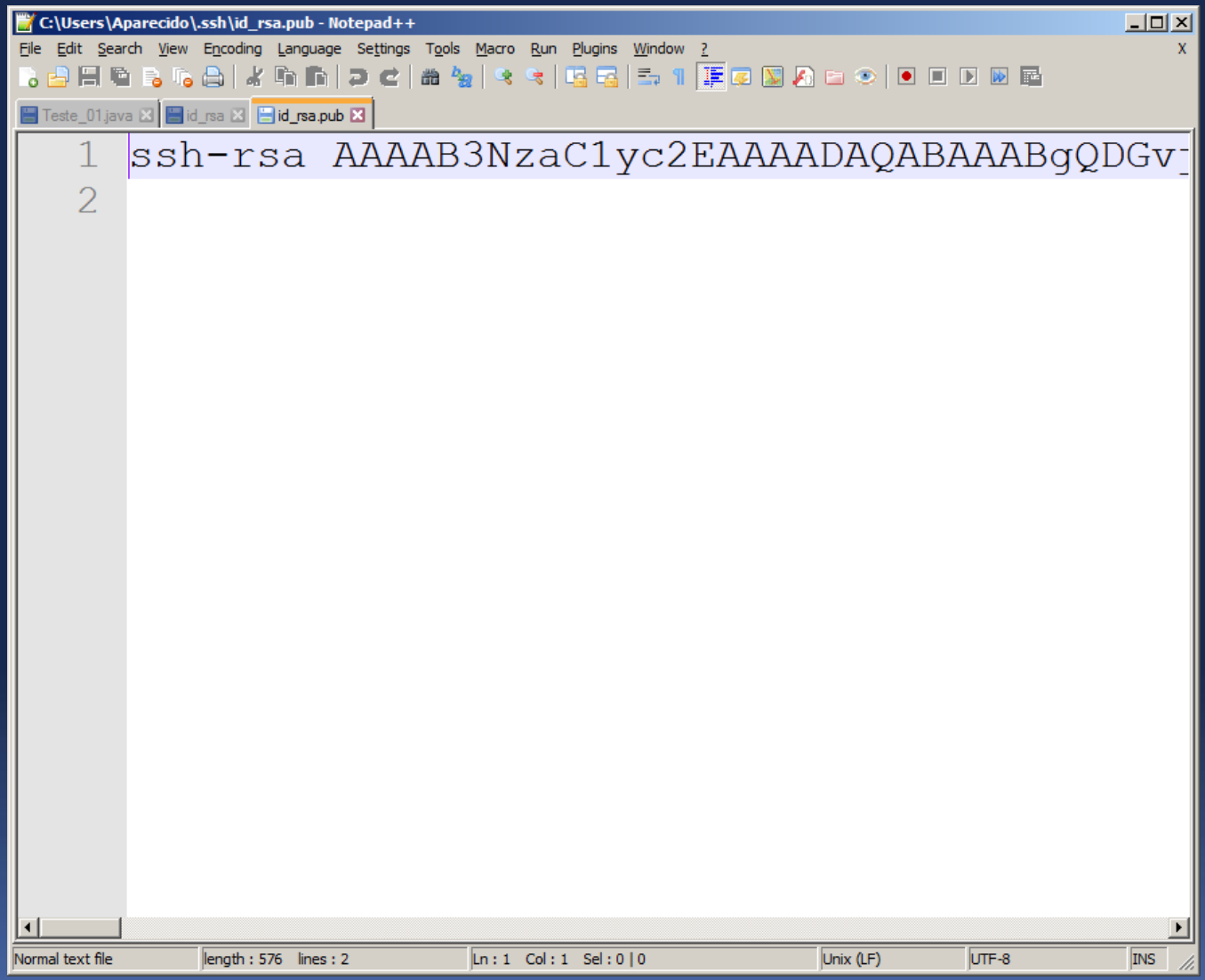
Abrindo os arquivos de segurança

c:/users/Aparecido/.ssh/id_rsa

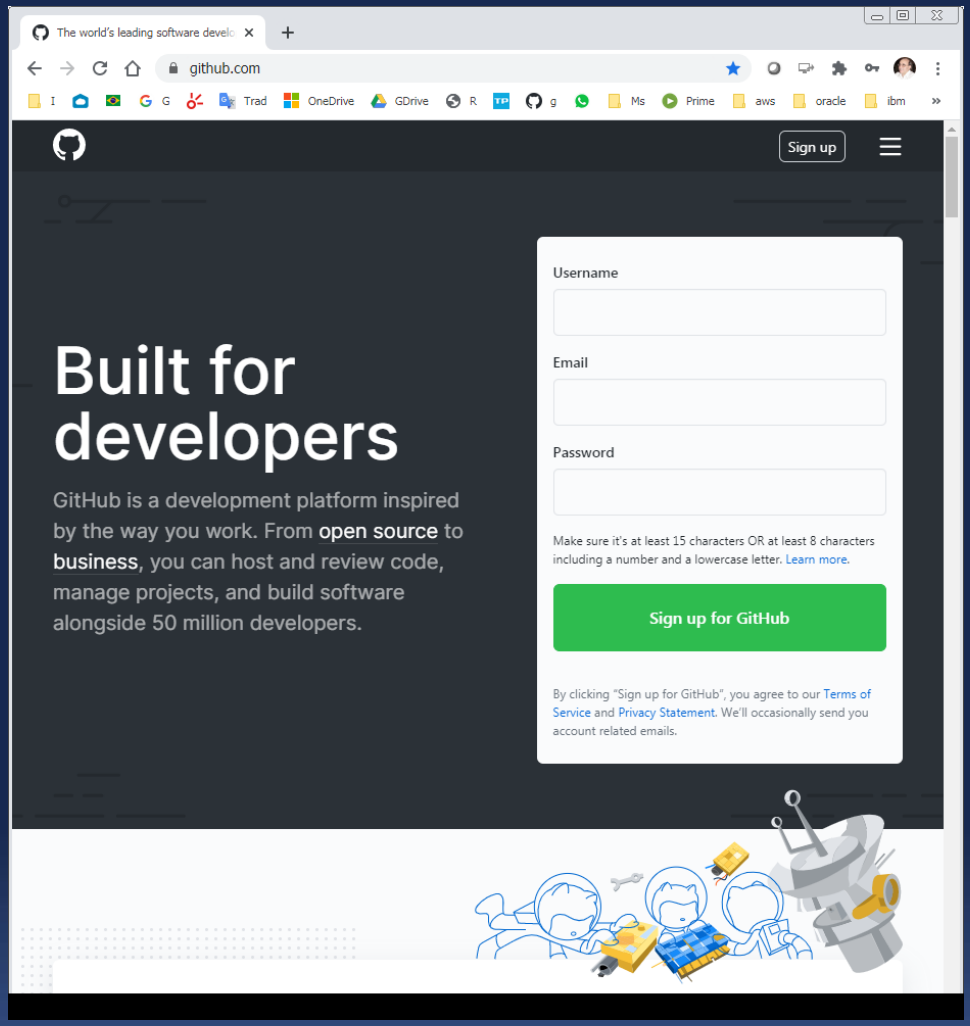


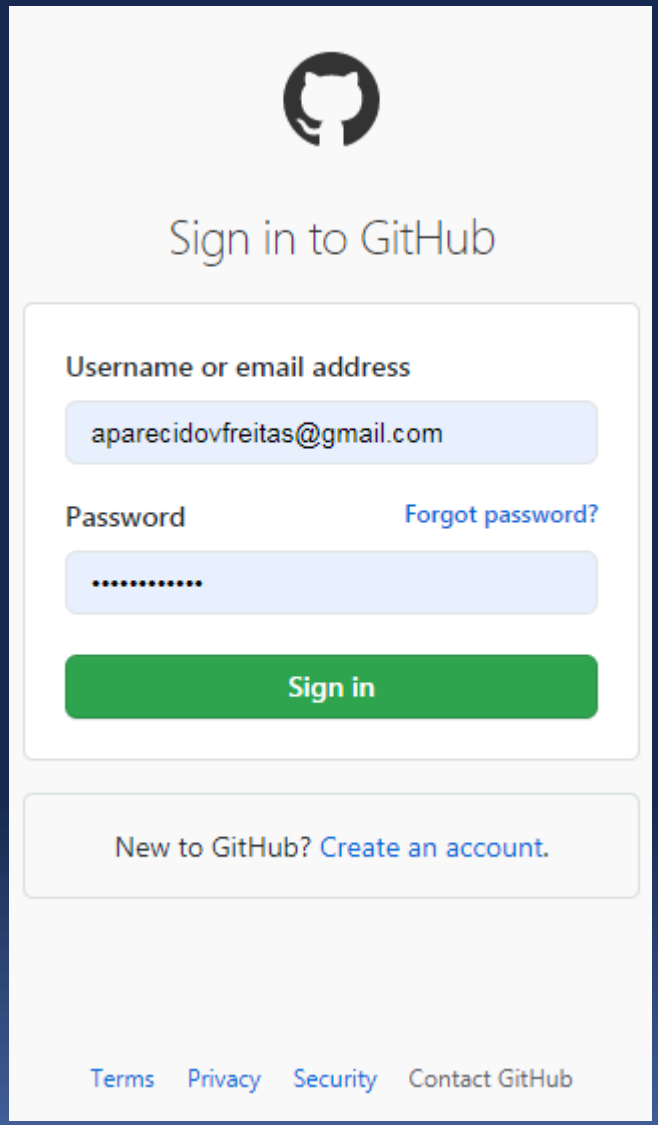
Abrindo os arquivos de segurança

c:/users/Aparecido/.ssh/id_pub

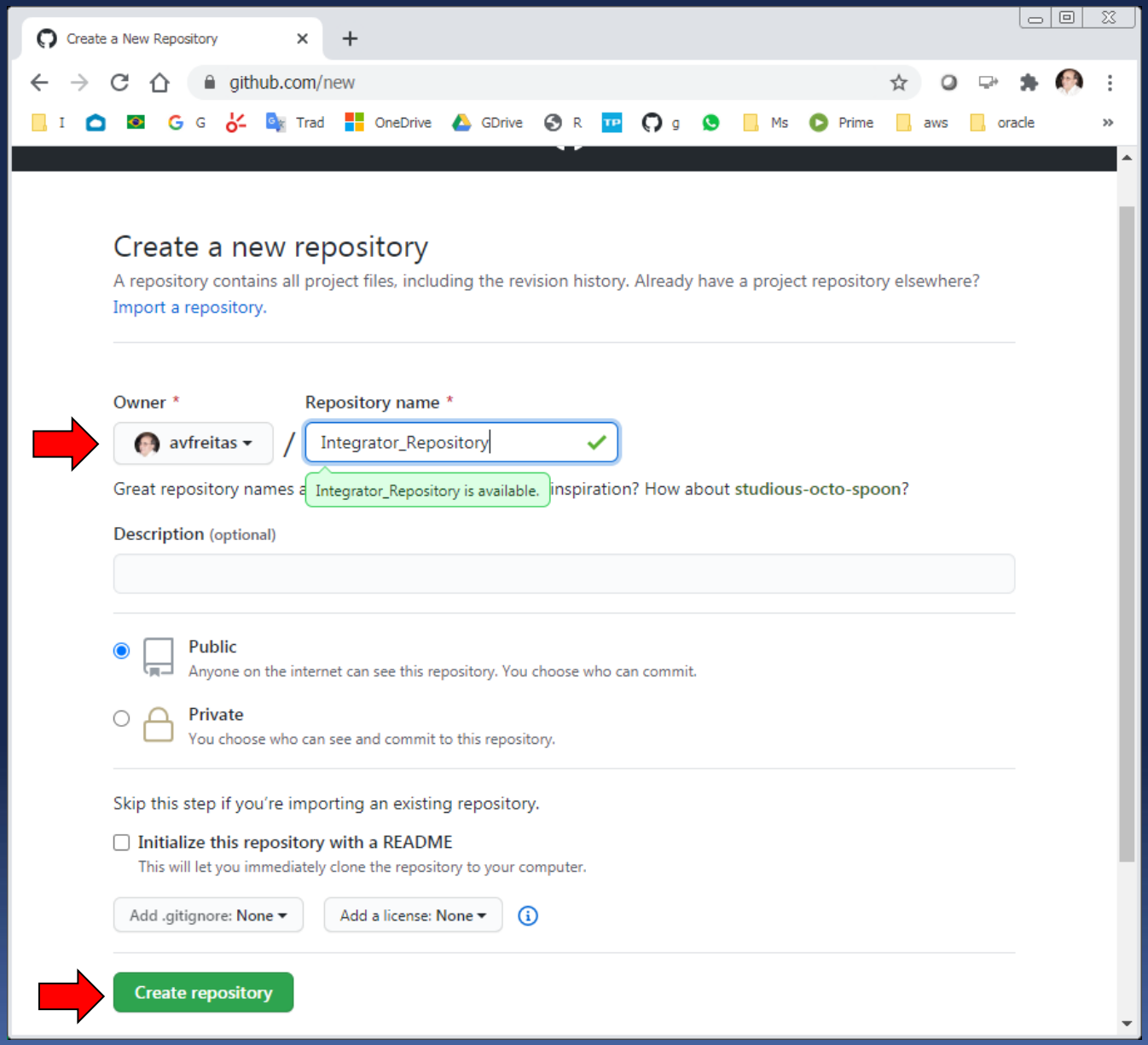


Acessando o github

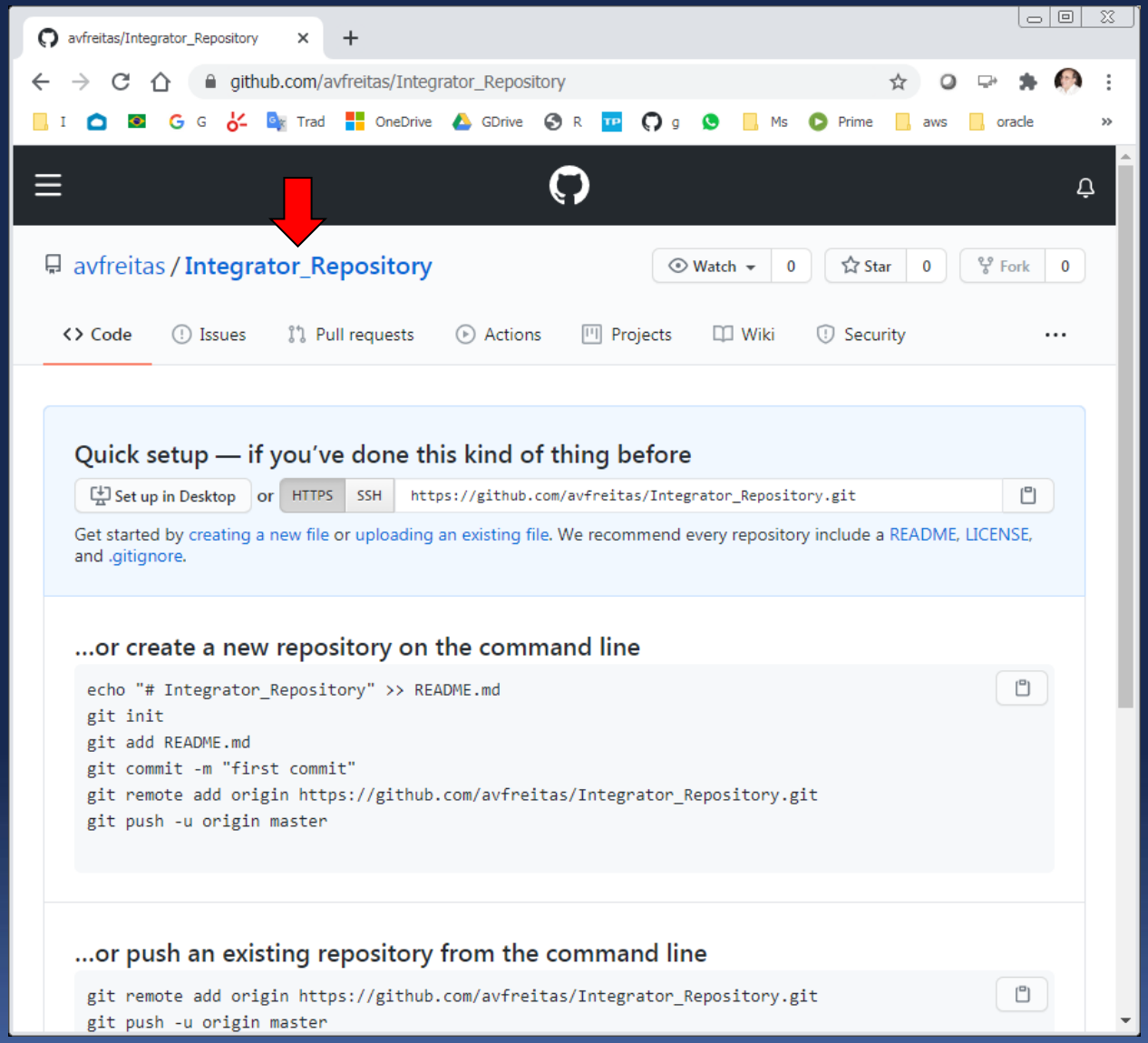




Criando novo repositório remoto



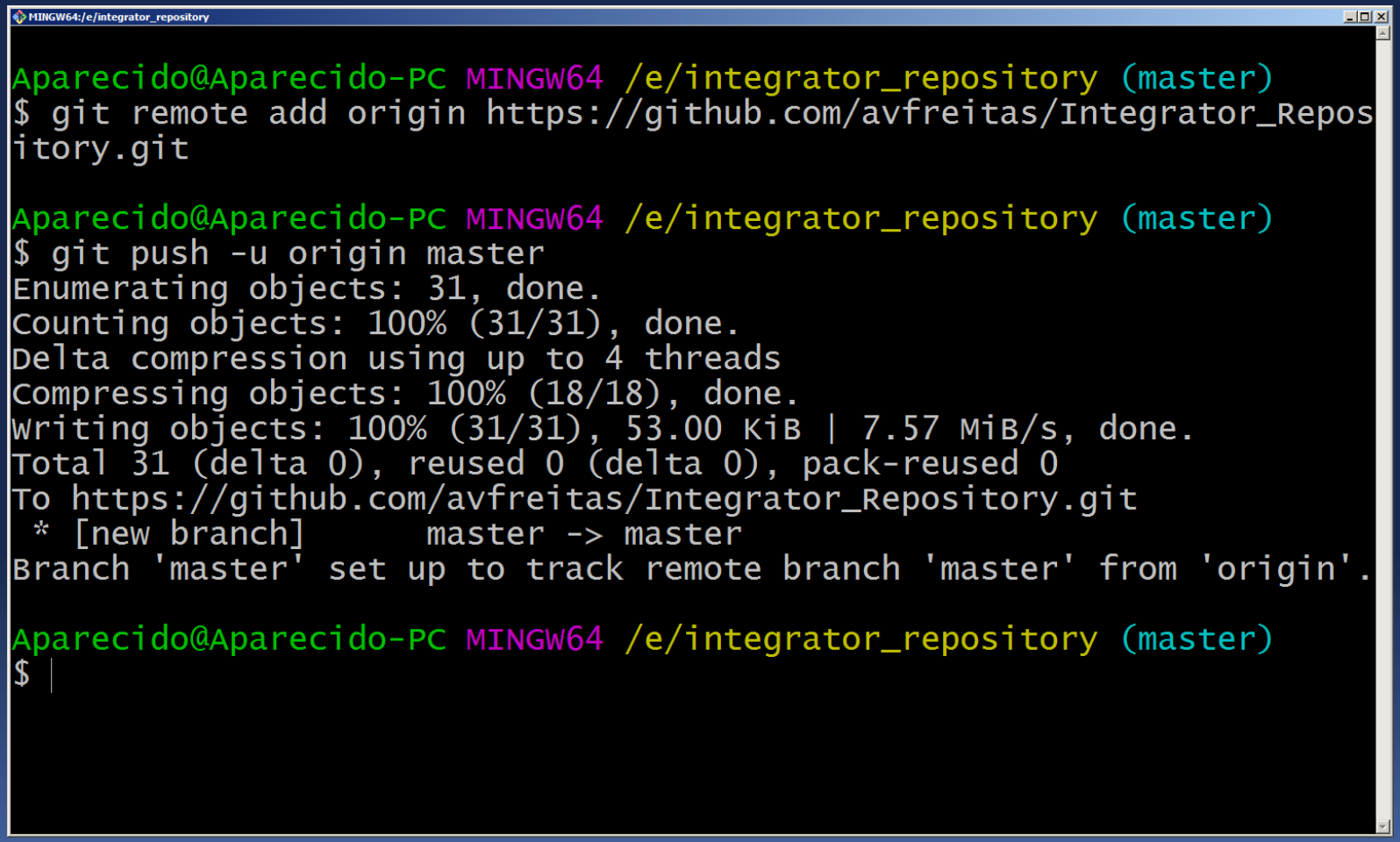
Criando novo repositório remoto



Subindo repositório local para github

Subindo repositório local github

$ git remote add origin https://github.com/avfreitas/integrator_repository
$ git push -u origin master



Projeto hellospringboot salvo no github