Nome: Filipe Serena D'avila
Telefone: (48) 99954-4905
Email: diusuorder@hotmail.com / filipe.davila@grad.ufsc.br

Desafio prático BRy Tecnologia - Estágio desenvolvedor back-end Java

Classes do projeto

main.control.CryptoController: A classe que contém todas os métodos criados com o objetivo de manipular e criar assinaturas e também autenticar documentos com elas.

main.control.FileRestController: Classe que manipula os comandos e arquivos providos por meio das requests REST. Possui os dois endpoints /signature/ e /verify/ conforme solicitado no PDF.

main.AssDigitalApplication: A classe principal da aplicação, que apenas serve para inicializar o Spring Boot que por sua vez disponibiliza os endpoints para utilização.

Timeline

Etapa 1 - Obtenção do resumo criptográfico – Arquivo: hashed doc.txt

Essa etapa foi rapidamente concluída com a utilização de um web app que faz o hash em SHA256. No arquivo resultante consta o hash SHA256 em hexadecimal conforme solicitado.

Etapa 2 - Realizar uma assinatura digital – Arquivos: certificado.txt, private key.txt, ArquivoAssinado.p7s

Essa etapa foi bem mais complicada (com vários erros de CertStore, até eu perceber que podia inicializar direto o JcaCertStore com X509Certificates de outro buffer). Primeiro, foi necessário entender o funcionamento das classes da lib BouncyCastle, bem como criar toda a estrutura de projeto e suas dependências – foi escolhido o Maven, e as dependências detalhadas no PDF foram adicionadas ao projeto. Após compreender mais ou menos o funcionamento das classes do BC, consegui extrair o certificado do alias provido e também a private key. Então, finalmente, após algumas tentativas, foi possível gerar o arquivo p7s com a assinatura digital.

Etapa 3 - Verificar a assinatura gerada – Arquivo: (Imagem abaixo)

Fiquei um bom tempo tentando corrigir um "bug", que era na verdade a ausência da declaração "Security.addProvider(new BouncyCastleProvider());" e seu respectivo import na hora de utilizar a BC.

Após corrigir este erro do provedor, foi possível pegar o certificado por meio da assinatura e em seguida, a verificação por meio do JcaSimpleSignerInfoVerifierBuilder. Na foto abaixo, confirmo a validação (retorno true) do próprio doc.txt provido utilizando a assinatura digital correspondente criada anteriormente (ArquivoAssinado.p7s).

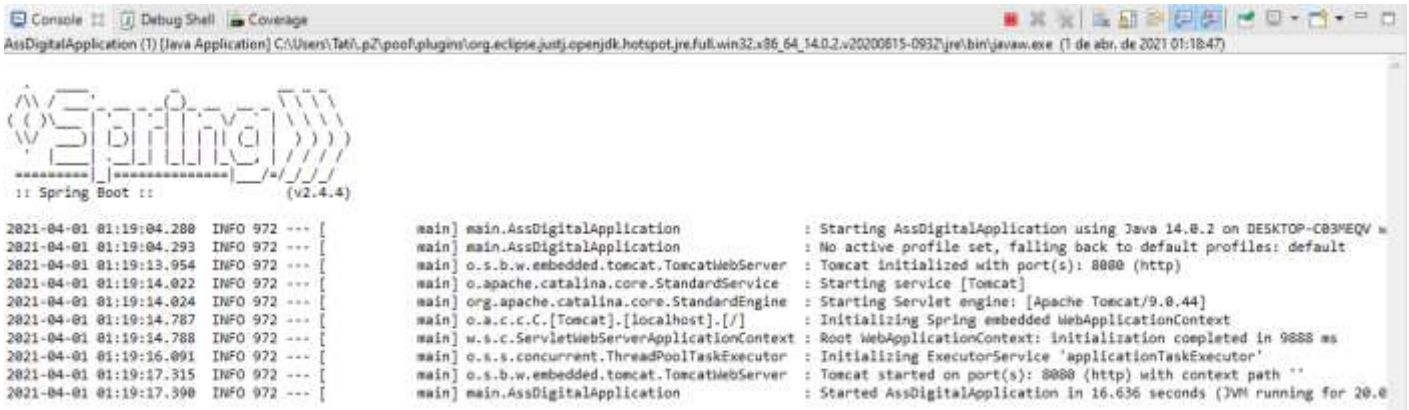
Nome: Filipe Serena D'avila
Telefone: (48) 99954-4905
Email: diusuorder@hotmail.com / filipe.davila@grad.ufsc.br

```
206             try {
207                 if (signatario.verify(
208                         new JcaSimpleSignerInfoVerifierBuilder().setProvider("BC").build(certificadoPelaAssinatura))) {
209                     sucesso = true; // unico caso onde retorna true
210                     System.out.println("Assinatura verificada com sucesso.");
211                     return sucesso;
212                 } else {
213                     System.out.println("Falha na verificacao da assinatura.");
214                     return sucesso;
215                 }
216             } catch (OperatorCreationException e) {
217                 System.out.println(
218                         "Falha na verificacao da assinatura. Verifique se o provedor de seguranca da lib BC esta incluso.");
219             } catch (CMSException e) {
220                 System.out.println(
221                         "Falha na verificacao da assinatura, verifique se esta utilizando os arquivos corretos de assinatura e texto.");
222             }
223         }
224         return sucesso; // sempre false quando chega ate aqui
225     }
```

```
Console  Debug Shell  Coverage
AssDigitalApplication (1) [Java Application] C:\Users\Tati\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_14.0.2.v20200815-0932\jre\bin\javaw.exe  (1 de abr. de 2021
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by org.bouncycastle.jcajce.provider.drbg.DRBG (file:/C:/Users/Tati/.m2/repository/org/bouncycastle/bcprov-j
WARNING: Please consider reporting this to the maintainers of org.bouncycastle.jcajce.provider.drbg.DRBG
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations
WARNING: All illegal access operations will be denied in a future release
Assinatura verificada com sucesso.
```

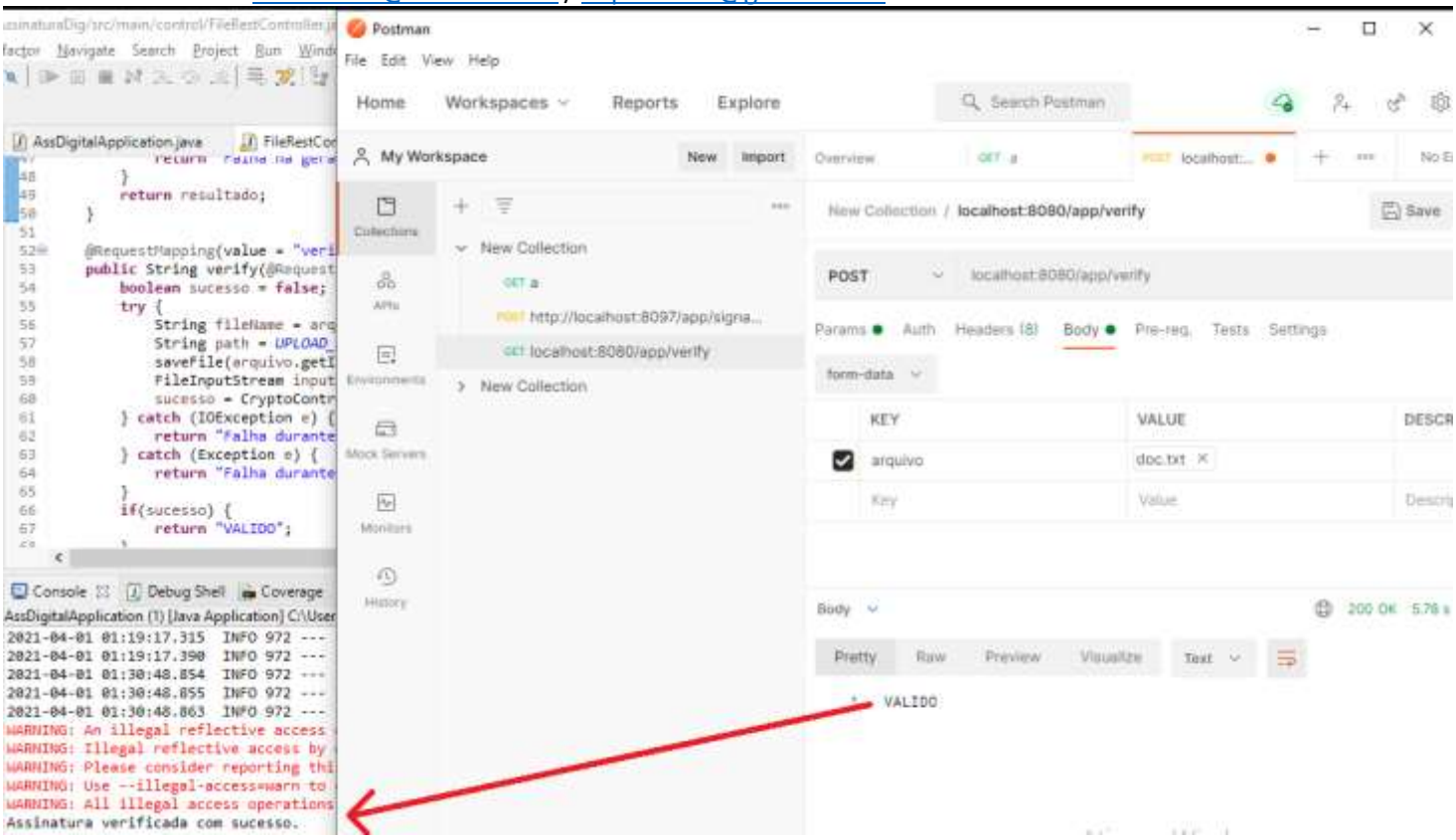

Etapa 4 - API REST – Arquivo: (Imagens abaixo)

First things first: Segue abaixo imagem comprovando o start do servidor sem erros.

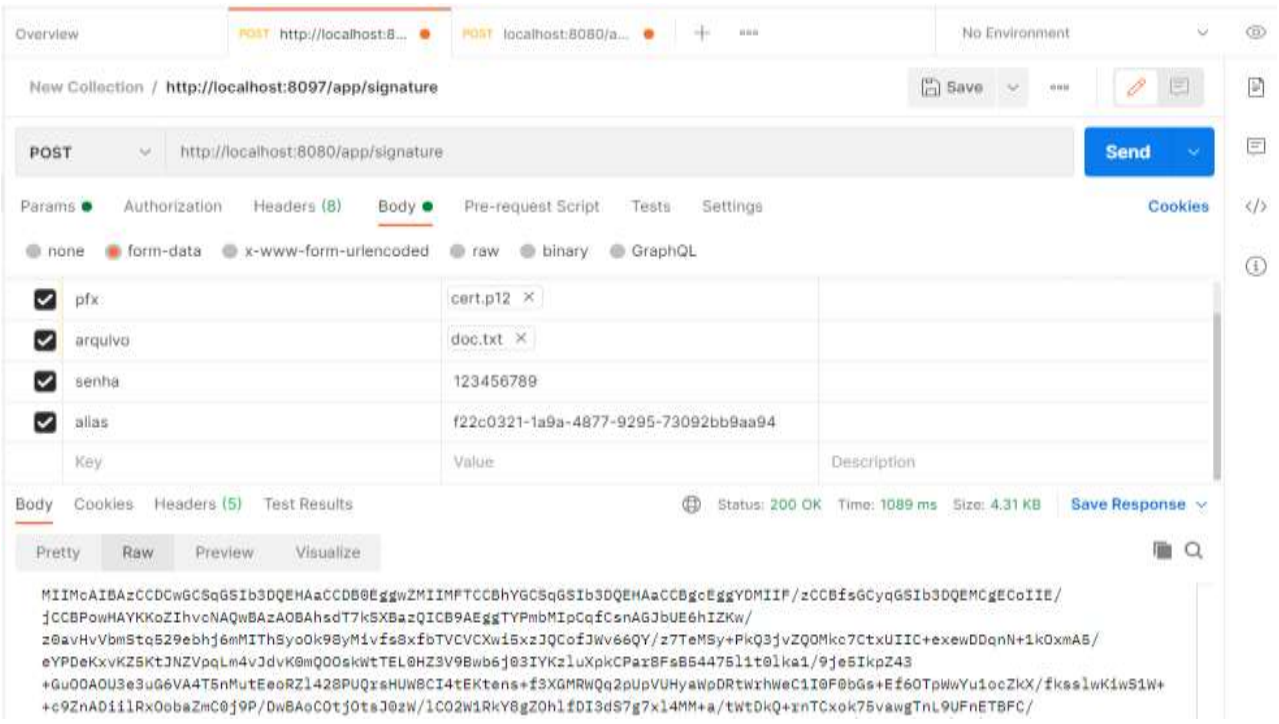


Nesse momento, eu tive que fazer uma boa pesquisa e entender primeiro como funciona a utilização da API REST no Spring Boot (marinheiro de primeira viagem, apesar de já ser cliente antigo no assunto REST por já ter o utilizado em JS / Python). Confesso que fiquei um bom tempo imaginando como que faria aquilo sem interface gráfica, como que funcionaria tudo aquilo de receber o arquivo pela requisição – mas rapidamente percebi que nada disso era necessário e que era possível fazer tudo só utilizando a API REST do Spring Boot e um par de requests com as keys corretas no Postman. Consegui então, primeiro, fazer a verificação da assinatura por meio do arquivo original, pelo método "verify". Fiz a validação do mesmo doc.txt utilizando a assinatura digital correspondente criada anteriormente (ArquivoAssinado.p7s). Abaixo, segue a imagem da resposta no Postman para a requisição abaixo:

Nome: Filipe Serena D'avila
Telefone: (48) 99954-4905
Email: diusuorder@hotmail.com / filipe.davila@grad.ufsc.br



Em seguida, depois de conseguir validar pela requisição, tinha chegado a hora de criar a assinatura também dessa forma. Segue abaixo a requisição e a resposta do procedimento de assinatura (cert.p12 é o **mesmo** arquivo de chaves provido pela BRy no teste, apenas encurtei o nome):



E com isso, encerro a participação no desafio, com todas as etapas concluídas com sucesso, agradecendo a empresa pela oportunidade de participar do mesmo. **Atenciosamente, Filipe Serena D'avila.**