BRyTecnologia

Desafio prático - estágio desenvolvedor back-end Java

Este é um roteiro das atividades do desafio para vaga de estágio Desenvolvedor Java. O desafio consiste em 5 etapas, sendo elas: obtenção do resumo criptográfico de um arquivo, geração de uma assinatura digital, verificação de uma assinatura digital, criação de uma API REST e a escrita de um relatório.

O objetivo é avaliar o candidato no uso das tecnologias indicadas e das boas práticas de programação considerando o nível esperado (júnior, pleno ou sênior). A implementação é livre, use sua criatividade!

Qualquer dúvida envie um e-mail para darlan@bry.com.br

Observações:

- É fácil encontrar código de exemplo pronto para este fim na Internet;
- Você pode se basear nestes códigos, mas espera-se que o sistema final entregue seja robusto e estável, o que normalmente não é o caso de códigos de exemplo;
- Utilize seu conhecimento e os métodos adequados para garantir/provar esta robustez e estabilidade;
- Comentários em código e tratamentos de erro são bem-vindos;
- Utilize Java 8 ou versão mais atual;
- Utilizar as bibliotecas BouncyCastle (bcprov e bcpkix) e commons-io;
- Utilizar o framework SpringBoot 2.4.x para a API REST;
- É desejável, mas não obrigatório, o uso do Maven para gestão de dependências;
- Caso não consiga concluir o desafio, entregue-o mesmo assim.
- No fim deste roteiro há alguns links úteis para ajudar

Entregáveis

- A entrega do desafio pode ser em um arquivo compactado do projeto por email ou no seu repositório GIT de preferência;
- Arquivo assinado da etapa 2;
- Relatório das etapas;

Etapa 1 - Obtenção do resumo criptográfico

Aqui a tarefa é bem simples, pois basta obter o resumo criptográfico de um documento. Note que o documento deve ser o arquivo doc.txt, que está na pasta resources/arquivos/.

Os pontos a serem verificados para essa etapa ser considerada concluída, são os seguintes:

- Obter o resumo criptográfico do documento, especificado na descrição dessa etapa, usando o algoritmo de resumo criptográfico conhecido por SHA-256;
- Anexar ao desafio um documento contendo o resultado do resumo criptográfico em hexadecimal.

Etapa 2 - Realizar uma assinatura digital

Essa etapa é um pouco mais complexa, pois será necessário implementar um método para gerar assinaturas digitais. O padrão de assinatura digital adotado será o Cryptographic Message Syntax(CMS).

Esse padrão usa a linguagem ASN.1, que é uma notação em binário, assim não será possível ler o resultado obtido sem o auxílio de alguma ferramenta. Caso tenha interesse de ver a estrutura da assinatura gerada, então, recomenda-se o uso da ferramenta ASN1 Javacript Decoder, disponível em: https://lapo.it/asn1js/.

Para realizar esta etapa, você fará uso do artefato contido em resources/pkcs12/. Primeiramente, você deve extrair a chave privada e o certificado deste arquivo PKCS12. Para isso, utilize a classe KeyStore do Java. O alias para obter o certificado é “f22c0321-1a9a-4877-9295-73092bb9aa94” e a senha para obter a chave privada é “123456789”.

Com os dados do assinante prontos, utilize a classe CMSSignedDataGenerator da BouncyCastle para preparar e gerar uma assinatura. O arquivo a ser assinado é o mesmo da primeira etapa.

Os pontos a serem verificados para essa etapa ser considerada concluída, são os seguintes:

- Gerar uma assinatura digital usando o algoritmo de resumo criptográfico SHA-256 e o algoritmo assimétrico RSA;
- Assinar o documento doc.txt;
- Gravar a assinatura em disco com a extensão ".p7s".

Etapa 3 - Verificar a assinatura gerada

A terceira etapa deste desafio consiste na verificação da assinatura que você gerou no passo dois. Para isso, utilize as classes SignerInformation e SignerInformationVerifier. Para gerar um Signer Information Verifier, utilize a classe JcaSimpleSignerInfoVerifierBuilder, todas da BouncyCastle.

Os pontos a serem verificados para essa etapa ser considerada concluída, são os seguintes:

- Verificar a assinatura gerada retornando true no resultado.

Etapa 4 - API REST

Na última etapa deste desafio, será implementado uma API Rest reutilizando o código criado nas etapas anteriores. Será necessário implementar dois endpoints, conforme descrição abaixo:

1. /signature/
    a. A requisição deve ser no formato multipart/form-data. No corpo da requisição deve conter:
        i. o arquivo a ser assinado;
        ii. o arquivo .pfx;
        iii. um parâmetro informando a senha do arquivo .pfx
    b. Deve retornar no corpo da resposta a assinatura no formato Base64.
2. /verify/
    a. A requisição deve ser no formato multipart/form-data. No corpo da requisição deve conter:
        i. um arquivo assinado
    b. Deve retornar no corpo da resposta:
        i. "VALIDO", caso a verificação da assinatura retorne true;

ii. “INVALIDO”, caso a verificação da assinatura retorne false

Para auxiliar nos testes da implementação, recomendamos o uso do software Postman, que possui uma interface simples para configurar uma requisição.

Os pontos a serem verificados para essa etapa ser considerada concluída, são os seguintes:

- Start do servidor sem erros;
- Endpoint conseguem receber dados e retornar uma resposta;
- Tratamento de erros;

Etapa 5 - Relatório das Atividades

Após concluir todas as etapas práticas, é hora de relatar os acontecimentos. Neste relatório, descrever as dificuldades encontradas durante a resolução das etapas, comente sucintamente o código implementado e, caso queira, detalhe informações que você achou relevante durante o desafio.

Obs: caso não conclua todas as etapas práticas, escreva um relatório mesmo assim (descrevendo o que foi feito e o que não foi feito), isso acaba ajudando no momento da avaliação.

Links úteis

- O que é uma assinatura digital?
    - Assinatura digital: descubra o que é e como funciona
- O que é um certificado digital?
    - Certificado digital: o que é e como funciona? Aprenda como usar
- SpringBoot
    - https://spring.io/
- Tutorial criar API Rest com SpringBoot
    - https://spring.io/guides/gs/rest-service/
- Repositório bibliotecas BouncyCastle:
    - http://repo2.maven.org/maven2/org/bouncycastle/
- Documentação BouncyCastle Provider:
    - https://jar-download.com/artifacts/org.bouncycastle/bcprov-jdk15on/1.60/documentation
- Documentação BouncyCastle PKIX:
    - https://jar-download.com/artifacts/org.bouncycastle/bcpkix-jdk15on/1.60/documentation
- Java Cryptography Architecture (JCA)
    - https://docs.oracle.com/javase/8/docs/technotes/guides/security/crypto/CryptoSpec.html
- RFC padrão assinatura CMS:
    - RFC 5652 - Cryptographic Message Syntax (CMS)