Relatório de Desenvolvimento – back-end Java

Etapa 1:

Iniciei a etapa com as leituras relacionadas a assinatura/certificado digital. Depois procurei por uma classe para gerar o resumo, escolhendo a classe MessageDigest. Utilizando esta classe transformo o conteúdo do arquivo a ser obtido o resumo criptográfico em conteúdo aceitavel pela classe e a utilizo.

Etapa 2 e 3:

Nestas etapas começei procurando sobre os métodos/classes que seriam utilizados (CMSSignedDataGenerator, KeyStore, SignerInformation, SignerInformationVerifier e JcaSimpleSignerInfoVerifierBuilder). Depois observei que o conteúdo retornado pela KeyStore não era o mesmo daquele utilizado pela classe da BouncyCastle, assim como não havia um método simples para obter o SignerInformation de um certificado. Estas transformações foram a maior dificuldade desta etapa, mas consegui resolve-las no trabalho após analisar algumas codificações funcionais já implementadas. Apos isto o restante de assinatura e validação foram implementados utilizando os métodos da BouncyCastle.

Etapa 4:

Para criação da API Rest iniciei criando um pacote de modelo, para implementar a estrutura dos modos de assinatura e verificação. Para assinatura criei uma classe que recebe os parametros e instancia os certificados com base nestes parametros. Depois de inicializada esta classe apenas assina as mensagens. Como a verificação de uma assinatura não possui nenhum método que necessite de criação de parametros, a implementei como um método estatico de uma classe Validacao.

Com estas duas classes de modelo criadas, inicializei a classe base do Spring Boot de forma a trabalhar com requisições multipart/form-data. Arrumei os métodos necessarios para utilizar /signature/ e /verify/, assim como envio de um formulario se a requisição for do tipo GET. Para receber os dados da requisição criei duas novas classes, Arquivo para relacionar os arquivos recebidos pelo método de /signature/ e ArquivoAssinado para os arquivos recebidos pelo método /verify/. Para as respostas das requisições também criei novas classes, que estão no pacote retorno. Estas classes recebem a informação relacionada ao retorno e a transformam para o modo como o retorno acontece.

Funcionamento:

A API recebe requisições POST nos endereços de /signature/ e /verify/. Para facilitar testes e utilização local ao utilizar uma requisição GET para estes endereços, um formulário contendo os campos necessários é gerado para preenchimento. O endereço /verify/ retorna VÁLIDO para arquivos pkcs7-signature assinados e INVALIDO caso contrário. O endereço /signature/ retorna a assinatura no formato Base64 se o conteúdo representado por filePFX está no formato x-pkcs12, com alias e senha corretos. Ele retorna um erro caso contrário.