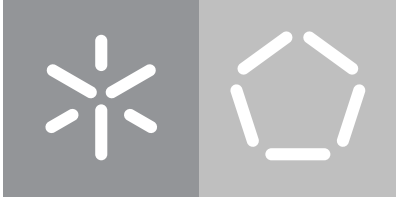
Universidade do Minho
Escola de Engenharia

Francisco Fernando Vilela Araújo

PAdES Server Signer

Janeiro, 2022



Universidade do Minho
Escola de Engenharia

Francisco Fernando Vilela Araújo

PAdES Server Signer

Dissertação de Mestrado
Mestrado em Engenharia Informática

Trabalho efetuado sob a orientação do(a)
Professor Doutor José Carlos Bacelar Almeida

Janeiro, 2022

DECLARAÇÃO DE INTEGRIDADE

Declaro ter atuado com integridade na elaboração do presente trabalho académico e confirmo que não recorri à prática de plágio nem a qualquer forma de utilização indevida ou falsificação de informações ou resultados em nenhuma das etapas conducente à sua elaboração.

Mais declaro que conheço e que respeitei o Código de Conduta Ética da Universidade do Minho.

_____, _____

(Localização) (Data)

_____

(Francisco Fernando Vilela Araújo)

Agradecimentos

Em primeiro lugar, gostaria de agradecer ao Doutor José Pina Miranda por toda ajuda e por toda a disponibilidade que teve ao longo da realização deste projeto. Gostaria, igualmente, de agradecer ao Professor Doutor José Bacelar Almeida por ter aceite ser meu orientador e me ter ajudado sempre que foi necessário.

Aos meus pais, por me terem dado tudo o que era necessário para eu ter uma percurso académico de qualidade.

À Patrícia Ferraz por toda a ajuda e por ter sido uma das pessoas mais importantes na minha vida académica.

Aos meus amigos, que realizaram esta caminhada na universidade comigo, quero agradecer por todo o auxílio e força que me deram para atravessar as adversidades encontradas.

Resumo

Em todas as organizações, públicas ou privadas, as soluções de desmaterialização são cada vez mais usadas, começando a existir pressão para a assinatura eletrónica (em formato PAdES) dos documentos PDF. Com a introdução de tecnologias de assinatura qualificada remota, cada vez menos se justifica que os documentos sejam assinados de forma manuscrita e depois digitalizados para serem integrados no sistema documental da organização, sendo natural a evolução para a integração de serviços de assinatura no próprio sistema documental da organização.

Neste sentido, esta dissertação de mestrado tem como objetivo desenvolver uma plataforma (que designaremos por "PAdES Server Signer") que permita efetuar uma assinatura eletrónica (em formato PAdES) de documentos PDF, com base nos mecanismos de assinatura eletrónica: CMD (Chave Móvel Digital) e CC (Cartão de Cidadão). A plataforma rege-se pelas rigorosas especificações do Regulamento eIDAS da União Europeia para assinaturas eletrónicas. Perante o aumento da utilização das assinaturas realizadas remotamente, visa-se garantir que a plataforma esteja em conformidade com a especificação do *Cloud Signature Consurtium* para a realização de assinatura baseadas em chave privada e *hardware* criptográfico remoto, utilizado sob controlo do titular da mesma. Em paralelo, foi realizada uma Aplicação Web que realize a interação com o utilizador, permitindo que este possa assinar um documento PDF, através do *PAdES Server Signer*.

Palavras-chave: Assinaturas Digitais, PDF, PAdES.

Abstract

In all public or private organizations, the dematerialization solutions are even more a requisition. Pressure is beginning to exist for the electronic signatures for PDF documents (in PAdES format). With the introduction of qualified remote signature technologies, less and less reasons are required for documents to be signed by hand and then scanned to be integrated into the organization's document system.

In this master's thesis was developed a platform (which we will call "PAdES Server Signer") that allows the signature of PDF documents (in PAdES format), based on CMD (Chave Móvel Digital), CC (Cartão de cidadão). This signatures must complie with the strict specifications of the European Union's eIDAS Regulation for electronic signatures. With the increase in the use of remote signatures, this platform is guided by the Cloud Signature Consurtium standard for the creation of digital signatures where the signing key is held "in the cloud". At the same time, was developed a Web Application for the user to sign a PDF document using the PAdES Server Signer application.

Keywords: Digital Signatures, PDF, PAdES.

Índice

Capítulo 1

Introdução

1.1 Contextualização e Motivação

Em todas as organizações, públicas ou privadas, as soluções de desmaterialização são cada vez mais usadas. Deste modo, salienta-se a necessidade crescente de recorrer a uma assinatura eletrónica dos documentos PDF, de modo a garantir a integridade do documento, autenticação e não repúdio da assinatura.

A introdução de tecnologias de assinatura qualificada remota, que se caracterizam por ser muito mais práticas e eficazes, conduzem ao abandono do procedimento anteriormente utilizado, o qual não garante a integridade nem a autenticidade do documento. Assim, deixa de ser necessário que os documentos sejam assinados de forma manuscrita e, posteriormente, scanados para serem integrados no sistema documental da organização. Neste sentido, espera-se uma natural evolução para a integração de serviços de assinatura no próprio sistema documental da organização. A utilização de assinaturas eletrónicas remove a necessidade da utilização do papel,levando a uma maior facilidade de processos, já que elimina o tempo, custos e riscos da utilização do formato em papel.

Nesse sentido, a União Europeia tem vindo a criar regulamentação, ferramentas e serviços para a realização de Assinaturas Eletrónicas que visam uma maior segurança e confiança para os cidadãos utilizarem estes tipos de assinaturas. Além disso, procuram promover uma maior utilização destes serviços, ajudando as empresas e as administrações públicas a acelerar os processos de criação e verificação de assinaturas, que são legalmente válidas em todos os estados membros da União Europeia.

A utilização das assinaturas eletrónicas facilita a experiência do utilizador, com tempos de aprovação de assinatura mais pequenos e uma realização de assinatura mais rápida, sem necessidade de utilização do formato em papel. Este tipo de assinaturas garante a integridade do documento, reduzindo assim o risco de duplicação ou alteração do mesmo.

O principal Regulamento da União Europeia em relação às assinaturas electrónicas é o Regulamento

n.°910/2014 [18] também conhecido como Regulamento eIDAS, que visa "...reforçar a confiança nas transações eletrónicas no mercado interno criando uma base comum para a realização de interações eletrónicas em condições seguras entre os cidadãos, as empresas e as autoridades públicas, aumentando assim a eficácia dos serviços públicos e privados em linha, os negócios eletrónicos e o comércio eletrónico na União."

Nesse seguimento, o European Telecommunication Standards Institute (ETSI), publicou *standarts*, para o tipo de assinaturas PAdES [14][19]. Estes são baseados no conceito de assinatura digital do PDF [2] e descreve um conjunto de perfis que estão de acordo com o Regulamento eIDAS da União Europeia que visa uma maior segurança nas assinatura eletrónicas.

Nos últimos anos, verifica-se que as assinaturas electrónicas que eram baseadas em chave privada e *chip* criptográfico dos *tokens* e *smartcards*, têm vindo a ser substituídas substituídas por assinaturas em *Cloud* (baseadas em chave privada e *hardware* criptográfico remoto, utilizado sob controlo do titular da mesma). Atendendo a esta informação, apresentam-se as seguintes razões: oferta de uma maior simplicidade e permissão de uma maior utilização de dispositivos móveis, os quais nem sempre têm meios de conectividade para os tokens.

O aumento da utilização da assinatura em *Cloud* contribuiu para a criação do Cloud Signature Consurtium (CSC). Este grupo de organizações industriais e académicas procura padronizar a arquitetura das plataformas de assinatura em *Cloud*, com vista a que todas partilhem os mesmos protocolos na comunicação, bem como as mesmas estruturas de dados para diferentes serviços de assinatura digital. A utilização do standard disponibilizado pelo CSC vai de encontro aos os requisitos impostos pelo regulamento eIDAS da União Europeia, responsável por estabelecer as regras para identificação eletrónica na União Europeia.

Numa perspetiva nacional, existem dois principais mecanismos capazes de assinar eletronicamente documentos, com valor legal, fornecidos pelo Estado:

- Chave Móvel Digital (CMD) - trata-se de um meio seguro de assinatura electrónica qualificada que permite que um cidadão assine digitalmente, em formato PDF, recorrendo a uma palavra-passe e um respetivo código de segurança, previamente definidos;
- Cartão de Cidadão (CC);

No âmbito desta dissertação, cujo tema foi proposto pela empresa Devise Futures Lda., pretende-se disponibilizar uma plataforma de assinatura em cloud, de acordo com o standard CSC, e que permita assinar com o CMD e CC, i.e., uma plataforma que disponibiize um interface CSC a aplicações cliente utilizando o CMD e CC para assinatura,

No âmbito desta dissertação, cujo tema foi proposto pela empresa Devise Futures Lda., pretende-se disponibilizar uma plataforma de assinatura em *Cloud*, de acordo com o standard CSC, e que permita assinar com o CMD e CC, i.e., uma plataforma que disponibilize um interface CSC a aplicações cliente utilizando o glscmd e CC para assinatura, em *cloud*, utilizando as tecnologias e processos necessários

para estar de acordo com as normas da União Europeia. O utilizador pode selecionar o tipo de assinatura PAdES em que pretende assinar, formato e local onde deseja assinar.

1.2 Questão de investigação

Em primeiro lugar, é necessário realizar uma questão de investigação que especifique a problemática a resolver neste projeto.

A dissertação pretende responder à seguinte pergunta:"É possível desenvolver uma plataforma de assinaturas electrónicas (que designaremos por '*PAdES Server Signer*'), e acordo com o standard CSC, que permita realizar assinaturas electrónicas, nos vários formatos PAdES, através do Cartão de Cidadão de da Chave Móvel Digital, de acordo com o regulamento eIDAS?"

1.3 Objetivos

Perante estas situações e pela crescente utilização de assinaturas eletrónicas no formato PAdES, esta dissertação terá os seguintes objetivos:

1. Desenvolver uma arquitetura capaz de efetuar a assinatura (em formato PAdES) de documentos PDF no "*PAdES Server Signer*", a partir do *web browser* (assim como de uma aplicação *desktop*) do utilizador, com as seguintes alternativas:

    a) assinatura com CMD , CC;

    b) com/sem assinatura visível;

    c) com local prévio para assinatura visível ou com identificação de local para assinatura através do *browser*.

2. Garantir a conformidade com a utilização da especificação do CSC na arquitetura do ponto anterior;

3. Avaliar a necessidade de *plugin* para assinatura com CC, na arquitetura do ponto 1;

4. Avaliar tecnologia, componentes/plataformas e ferramentas *open source* que possam ser utilizadas para a implementação do ponto 1;

5. Implementar (e integrar) as componentes de software necessárias para a concretização da arquitetura idealizada no ponto 1;

6. Utilizar ferramentas que permitam aquilatar da qualidade do código desenvolvido/utilizado, no que respeita vários factores, como por exemplo: *Code Coverage*, *Abstract Interpretation*, *Compiler Warnings*, *Coding Standards*, *Code Duplication*, *Security*, *Dead Code*.

1.4 Organização do documento

A seguinte dissertação está dividida em 6 capítulos. Primeiramente, apresenta-se este capítulo que diz respeito à contextualização do problema, sendo também relatada a Questão de Investigação e os Objetivos do projeto. O segundo capítulo diz respeito ao Estado de Arte, onde é verificado o conhecimento já existente acerca da matéria estudada. O terceiro capitulo contém a descrição de uma solução para a realização do problema apresentado. O quarto capítulo aborda as tecnologias utilizadas que ajudaram à codificação da plataforma de assinatura. Segue-se o capítulo cinco que contém as diversas maneiras que foram implementadas para testar a aplicação e aumentar a sua segurança. Por fim, no último capítulo encontra-se a conclusão, na qual se compara com os objetivos que foram traçados para este projeto e os trabalhos futuros.

Capítulo 2

Estado da arte

O Estado da arte é uma descrição do conhecimento já existente sobre uma determinada área ou tema, tendo como objetivo uma pesquisa exaustiva sobre os trabalhos já publicados acerca do caso de estudo.

Numa primeira instância, é efetuada uma investigação sobre os regulamentos para a assinatura digital tendo em foco o Regulamento eIDAS da União Europeia, que tem como um dos objetivos a realização de assinaturas eletrónicas seguras na União Europeia. Segui-se uma análise detalhada à estrutura do PDF para a realização de assinaturas eletrónicas, assim como, às tecnologias que permitem a realização de assinaturas eletrónicas no PDF em conformidade com o eIDAS.

Por fim, averiguam-se os documentos relativos aos serviços de assinatura em Portugal, sendo alvo de caso de estudo o CC e CMD.

2.1 Regulamento eIDAS

No ano de 2014, o Parlamento Europeu e o Conselho da União Europeia aprovaram o Regulamento EU n.º 910/2014 [18] relativo à identificação eletrónica e aos serviços de confiança para as transições eletrónicas no mercado interno. Também conhecido como o regulamento *eIDAS* "electronic IDentification, Authentication and trust Services", este determina uma base europeia para uma interação eletrónica segura, como assinaturas e transações eletrónicas, atribuindo uma maior confiança e segurança nas transações *online* na União Europeia. Tais características promovem uma maior utilização destes serviços *online* por parte dos cidadãos, operadores económicos e administração pública.[1]

O eIDAS estabelece um conjunto alargado de serviços de confiança, como também o reconhecimento mútuo transfronteiriço dos meios de Identificação Eletrónica (eID). Na data de 29 de setembro de 2018, ficou definido que um cidadão da UE com um cartão eID (meio de identificação eletrónica de acordo com

[1] https://www.gns.gov.pt/assinatura-eletrónica.aspx

o regulamento eIDAS), pode aceder a qualquer serviço público *online* a partir de um qualquer Estado-Membro da UE.[2]

O Regulamento criou padrões para assinaturas eletrónicas, selos eletrónicos, selos temporais, serviços de envio registado eletrónico e autenticação de *websites* e outros mecanismos de autenticação, assegurando que estes funcionam além fronteiras, possuindo o mesmo poder legal que os mecanismos em papel.[3]

2.1.1 Níveis de assinaturas eletrónicas no eIDAS

De acordo com o eIDAS, uma assinatura eletrónica consiste "(...) em dados sob forma eletrónica, ligados ou logicamente associados a outros dados eletrónicos, e que sejam utilizados como método de autenticação". Por outro lado, uma assinatura digital, refere-se ao conceito matemático e criptográfico que é amplamente utilizado para realizar instâncias práticas da assinatura digital. A ETSI TR 119 100 [7] define como os dados anexados ou uma transformação criptográfica dos dados que permitem que ao dono desses, provar a fonte e a integridade dos dados e os proteja contra uma falsificação [18].

O eIDAS definiu três tipos de assinaturas eletrónicas: as 'Simple' Eletronic Signatures (Assinatura Eletrónica), as mais simples, tratam-se de todas as assinaturas que não são avançadas nem qualificadas; as Assinatura Eletrónica Avançada (AdES) e as Assinatura Eletrónica Qualificada (QES). Os requisitos de cada nível de assinatura são realizados partindo dos requisitos do nível abaixo dele. A grande diferença entre as três assinaturas encontra-se no nível de segurança de cada assinatura e na complexidade do sistema de verificação da identidade do assinante. À medida que o nível de assinatura progride, os requisitos para a verificação da identidade do assinante e para a autenticidade dos documentos tornam-se mais rigorosos [18].[4]

De uma modo global, uma assinatura corresponde a dados em forma eletrónica anexados a um documento. Não existe nenhuma lista de requisitos estabelecida pelo regulamento eIDAS para este tipo de assinaturas. Esta assinatura é a mais simples o regulamento eIDAS, no artigo 25° [18], indica que, não lhe pode ser negado efeitos legais nem admissibilidade enquanto prova em processo judicial pelo simples facto de se apresentar em formato eletrónico.

A Assinatura Eletrónica Avançada (AdES) é uma assinatura que tem um maior nível de segurança. Uma assinatura eletrónica é considerada avançada se obdecer aos seguintes requisitos, no artigo 26° do regulamento eIDAS [18].

1. Estar associada de modo único ao signatário;
2. Permitir identificar o signatário;
3. Ser criada utilizando dados para a criação de uma assinatura eletrónica que o signatário pode, com um elevado nível de confiança, utilizar sob o seu controlo exclusivo; e

[2] https://www.gns.gov.pt/assinatura-eletrónica.aspx
[3] https://www.gns.gov.pt/assinatura-eletrónica.aspx
[4] https://www.adobe.com/hk_en/sign/compliance/eidas.html

4. Estar ligada aos dados por ela assinados de tal modo que seja detetável qualquer alteração posterior dos dados.

Se estes requisitos forem garantidos, então a assinatura irá garantir o não-repudio do signatário e integridade dos dados. Esta assinatura fornece uma segurança completa e de alto nível dos documentos. A integridade do produto assinado é, portanto, a sua principal característica, garante que alteração ao documento é detetada, bem como a identidade do assinante[18].

A Assinatura Eletrónica Qualificada (QES) é uma assinatura eletrónica avançada criada por um dispositivo qualificado de criação de assinaturas eletrónicas e que se baseie num certificado qualificado de assinatura eletrónica. Esta é a assinatura eletrónica com o maior nível de segurança e "tem um efeito legal equivalente ao de uma assinatura manuscrita"(conforme artigo 25º do regulamento eIDAS) [18]. As suas restrições de como a identidade do assinante é verificada e de que modo a chave de assinatura é guardada, estão reguladas. Por fim, sublinha-se que é reconhecida em todos os estados membros da União Europeia.

Um certificado qualificado de assinatura eletrónica tem as seguintes características:[5]

1. uma indicação de que é emitido como certificado qualificado;
2. a identificação do prestador de serviços de certificação;
3. o nome do signatário;
4. a possibilidade de incluir um elemento adicional específico de autenticação, como a data de nascimento, do signatário (segundo os objetivos visados com a emissão do certificado);
5. os dados de verificação de assinaturas: estes devem corresponder aos dados de criação de assinaturas que estejam sob o controlo do signatário;
6. as datas de início e de fim do prazo de validade do certificado;
7. o código de identidade do certificado;
8. a assinatura eletrónica avançada do prestador de serviços de certificação que o emite.

A emissão do certificado qualificado requer que o assinante utilize um prestador qualificado de serviços de confiança que esteja identificado na European Union Trusted Lists (EUTL). A este respeito, convém esclarecer que a EUTL é uma lista pública com mais de duzentos Trust Service Providers (TSPs) ativos que estão credenciados para fornecer serviços de confiança segundo o regulamento eIDAS. Cada Estado Membro supervisiona os prestadores destes serviços no seu país. Quando um TSPs é aprovado num país, os seus serviços podem ser utilizados noutro país da União Europeia com o mesmo nível de conformidade [1].

[5] https://eur-lex.europa.eu/legal-content/PT/TXT/?uri=LEGISSUM:l24118

De seguida, apresentam-se os serviços qualificados disponibilizados por prestadores qualificados de serviços de confiança portugueses, à data de escrita desta dissertação e que tem relevância para a mesma:

Prestadores de Serviços	Certificados Qualificados para Assinaturas Eletrónicas	Certificados para Selos Temporais	Certificados Qualificados para Selos Eletrónicos
ACIN iCloud Solutions, Lda	✓	✓	✓
AMA - AGÊNCIA PARA A MODERNIZAÇÃO ADMINISTRATIVA I. P.	✓		
CEGER - Centro de Gestão da Rede Informática do Governo	✓	✓	✓
DigitalSign - Certificadora Digital	✓	✓	✓
Instituto dos Registos e do Notariado I.P.	✓	✓	
MULTICERT - Serviços de Certificação Electrónica S.A.	✓	✓	✓

Figura 2.1: Lista de TSPs em Portugal Fonte:[23]

2.1.2 Tipos de perfis de assinatura

De acordo com aos requisitos do regulamento eIDAS, o ETSI desenvolveu um conjunto de perfis base que contêm um conjunto de padrões apropriados para realizar assinaturas eletrónicas avançadas segundo o eIDAS, que se chamam os "Ades Baseline Profiles"[11]:

1. XML Advanced Electronic Signatures (XAdES) que é um conjunto de extensões à assinatura no formato XML[8][9];
2. PDF Advanced Electronic Signatures (PAdES), que é um conjunto de restrições e extensões ao PDF[14][19].
3. CMS Advanced Electronic Signatures (CAdES), sendo este um conjunto de extensões aos dados assinados com Cryptographic Message Syntax (CMS)[6][20].

4. *ASiC Baseline Profile*[4][5].

5. JSON Web Electronic Signatures (JAdES)[12], que é especificação para *JSON Web Signatures*.

2.2 Assinatura no PDF

O PDF é um formato de arquivo desenvolvido pela Adobe Systems e padronizado pela International Organization of Standardization (ISO) no ISO 3200-1 [2] para exibir e compartilhar documentos com segurança, independentemente de software, hardware ou sistema operativo. Os documentos PDF contêm texto, imagens, *links* e botões, campos de formulário, áudio, vídeo e lógica de negócios. Existe também a possibilidade de assinar eletronicamente os PDF, tendo este um estrutura delineada para isso [2].

2.2.1 Estrutura de dados da assinatura PDF

Os documentos PDF são responsáveis por guardar a assinatura e os dados no seu conteúdo, depois de este ser assinado. O ISO 3200-1[2] suporta 3 tipos de atividades: adicionar uma assinatura digital ao PDF, inserir um campo onde se irá colocar as assinaturas no futuro e verificar a validade das assinaturas [2, 19].

A assinatura é um objeto binário utilizando CMS[15] ou formatos relacionados(PKCS 7[16] ou CAdES)[6][20]. O formato da assinatura depende do perfil PAdES em que esta foi desenvolvida.(ver secção 2.3.2)

A estrutura de dados que está incumbida de guardar as informações relativas à assinatura do PDF é designada por *signature dictionary* [19]. Como revela a figura 2.2, na sua composição constam, principalmente, os seguintes elementos:

1. ***Byte-Range***: Dados para a distinção entre o bloco de dados assinado e o não assinado.

2. ***Contents***:Contém o valor da assinatura do documento. Nas assinaturas no PDF do tipo PAdES, os dados devem ser guardados no formato binário DER codificado no formato CAdES definido no ETSI EN 319 122-1[20].

Antes de realizar a assinatura é necessário criar um *digest* sobre os dados que estão no PDF. Os bytes que irão ser assinados correspondem a todos os dados, à exceção do campo onde estão os dados da assinatura do PDF. Este está identificado na figura 2.2 como "PDF Signature", o qual é correspondente ao campo *Contents*. O campo *ByteRange* está incluído nos bytes que irão ser assinados. Por isso, antes da realização da assinatura, é necessário que previamente se tenha alocado o tamanho para o campo *Contents*. Este será preenchido com o valor hexadecimal de 0x00 e, só posteriormente, completado com o seu valor. Assim, é assegurado que só os dados correspondestes à assinatura não são assinados [19].

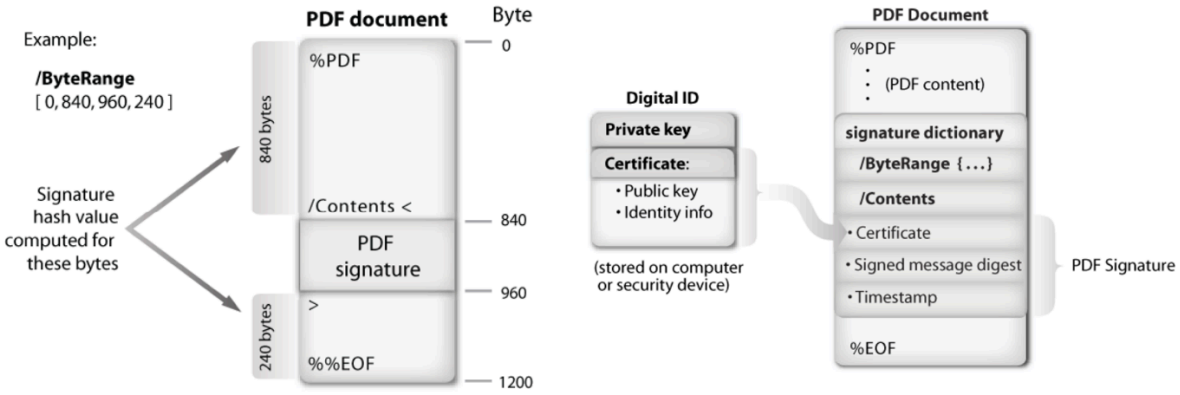


Figura 2.2: Estrutura de dados da assinatura no PDF Fonte:[19]

O PDF tem um campo chamado "MDP", que se trata de um dicionário do PDF que permite que o documento seja modificado em algumas partes e, mesmo assim, a assinatura do PDF é válida. Este dicionário pode conter 3 valores [2]:

1. Nenhuma mudança no documento pode ser realizada. Caso isso aconteça, a assinatura não é válida;
2. São permitidas alterações no preenchimentos de formulários, na realização de assinaturas e na modificação dos modelos das páginas;
3. É permitido tudo o que foi dito no ponto dois e, além disso, é possível criar, apagar e modificar anotações.

Enquanto noutras assinaturas digitais baseadas em CMS[15] é possível, para o mesmo byte range, conseguir várias assinaturas diferentes, no ISO 32000-1 [2] não é permitido. Este apenas permite adicionar um campo de assinatura de cada vez. A alternativa para realizar várias assinaturas no documento, consiste no seguinte: depois do primeiro individuo assinar o documento, o segundo individuo que quiser assinar, deve fazê-lo por cima do primeiro. Na figura 2.3 apresentada, exemplifica-se esta alternativa. O novo assinante pode também adicionar dados ao documento. No final, deve assinar os dados que adicionou, assim como o resto do documento.

No processo de verificação das várias assinaturas, cada assinatura é validada individualmente. Deste modo, no exemplo da figura 2.3, constata-se que se a assinatura um e dois forem válidas, então o documento que as contém será válido. Caso a terceira assinatura não seja válida, apenas o bloco que corresponde às alterações desta última versão será inválido. Por fim, embora as assinaturas em paralelo não sejam possíveis, existe o meio anteriormente referido para se conseguir ultrapassar esse obstáculo.

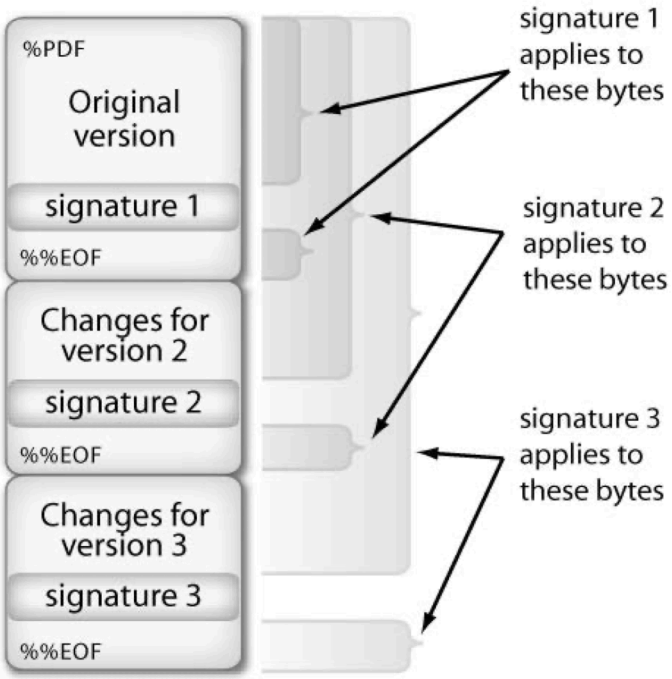


Figura 2.3: Estrutura de assinaturas em paralelo no PDF Fonte:[19]

2.3 PAdES

O padrão do PAdES - realizado e publicado pelo ETSI - concretizou-se em conformidade com a regulamento eIDAS - (*Electronic Identification*, *Authentication and Trust Services*) - da União Europeia. As assinaturas PAdES são baseadas em assinaturas PDF especificadas no ISO 32000-1[2], com uma codificação de assinatura digital diferente para suportar o tipo de assinatura com um formato equivalente ao CAdES especificado no ETSI EN 319 122-1 [20]. O documento que especifica as assinaturas digitais em formato PAdES é o ETSI EN 319 142-1[14] de 2016, o qual substitui os formatos dos perfis definidos no ETSI TS 103 172 de 2013 [14]. As assinaturas PAdES especificadas no documento ETSI EN 319 122-1 [20] visam suportar Assinaturas Eletrónicas, Assinaturas Eletrónicas Avançadas, Assinaturas Eletrónicas Qualificadas, Selos Eletrónicas, Selos Eletrónicos Avançados e Selos Eletrónicos Qualificados em concordância com o regulamento eIDAS [18].

No ETSI EN 319 142-1[14] são definidos 4 níveis de de perfis para realizar a assinatura do PDF no formato PAdES. Estes perfis vão aumentando a complexidade. Cada perfil implementa o perfil anterior e incrementa novos requisitos, acrescentando uma maior segurança à assinatura e providenciando validade para um período de tempo mais longo.

2.3.1 Atributos PAdES

No ETSI EN 319 142-1[14] são definidos os atributos que vão ser inseridos no documento PDF para criar uma assinatura PAdES, utilizando os atributos que estão definidos no ISO 32000-1[2] do PDF definidos na

secção 2.2.1, os atributos CAdES, assim como define novos atributos para criar uma assinatura PAdES.

Primeiramente, como é possível verificar na figura 2.2 o PDF contém uma estrutura de dados apelidada de *Signature Dictionary*, que é o local no PDF onde são guardadas os dados referentes á assinatura eletrónica. Nesta estrutura está um campo com a chave Contents, que deverá ter o valor do objeto SignedData que é especificado na clausula 5 do ETSI EN 319 122-1 [14](CAdES), tendo de estar codificado no formato DER[24].

Em relação aos restantes campos do *Signature Dictionary*, estes são utilizadas como estão definidas no ISO 32000-1 [2]. A este respeito, os atributos que são utilizados para realizar a assinatura PAdES são: M, Contents, Filter, SubFilter, ByteRange, Location, Name, ContactInfo e Reason.

A assinatura do tipo PAdES permite que as assinaturas mais avançadas sejam válidas por um longo período de tempo. Para tal, é necessário inserir no PDF os dados para validação da assinatura digital que este contém. Importa especificar que estes dados consistem na cadeia de certificação até ao Certification Authority (CA), na lista de certificados revogados (Certificate Revocation List (CRLs)) e no protocolo de status de certificado *online* (Online Certificate Status Protocol (OCSP)), conforme secção 2.3.2.

2.3.1.1 Dados para a validação da assinatura

Para colocar estes dados no PDF a ETSI definiu uma extensão ao ISO 32000-1[2] do PDF, a qual consiste num dicionário chamado *Document Security Store* (DSS). Este irá conter os dados necessários para proceder à validação do assinatura. Este campo de dados irá ser inserido na tabela 28 do ISO 32000-1[2] do PDF [2]. O Document Security Store contém os seguintes campos:

1. ***VRI***: Inclui os dados de validação para uma assinatura especifica que está no documento. Sendo um dicionário e a chave para cada elemento, trata-se do digest da assinatura ao qual os dados correspondem. Para a assinatura específica, o dicionário VRI vai conter todos os certificados, OCSPs e CRLs que vão permitir a validação da assinatura correspondente.

2. ***Certs***: Contém os certificados que vão ser usados para realizar a validação de todas as assinaturas que estão no documento.

3. ***OCSPs***: Contém os OCSP que vão ser usados para realizar a validação de todas as assinaturas que estão no documento.

4. ***CRLs***: Contém os CRLs que vão ser usados para realizar a validação de todas as assinaturas que estão no documento.

2.3.1.2 Time-stamping

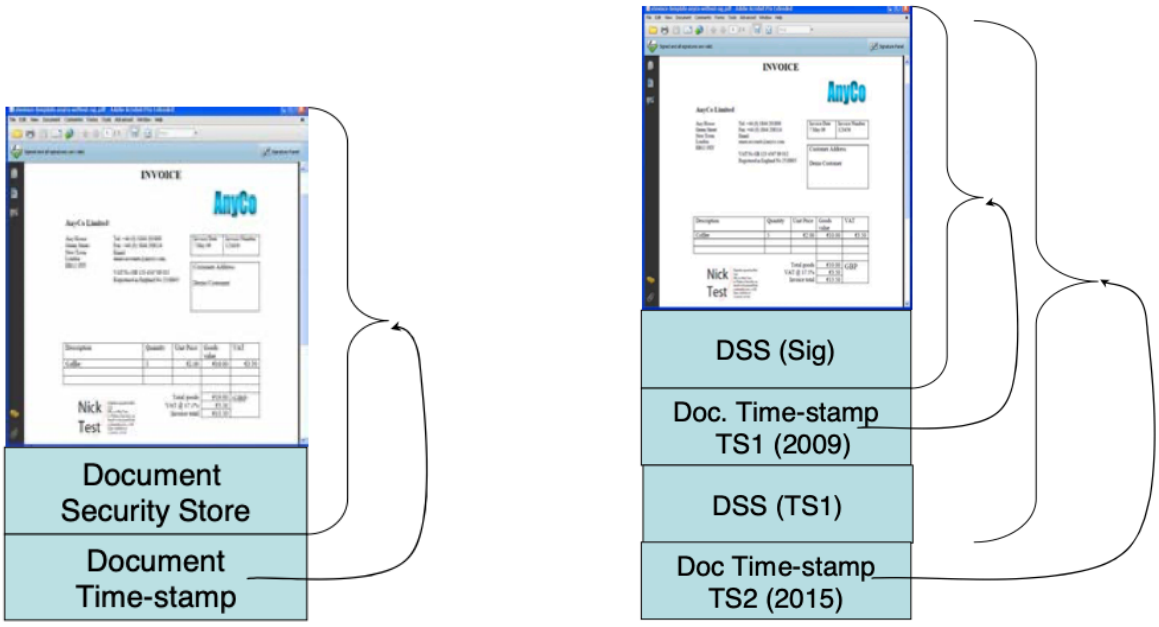
No PAdES é possível definir três Time-stamps. Primeiramente são definidos o signature-time-stamp que é um *Time-stamp* somente sobre a assinatura do documento. De seguida existe o content-time-stamp que é um *Time-stamp* sobre todos os dados do *SignedData* do campo *Contents* antes de este ser assinado.

Estes dois atributos encontram-se no dicionário *SignedData* definido no ETSI EN 319 122-1 [14], como já referido.

Por outro lado, definiu-se que seria possível realizar um Time-stamp sobre o documento, definido no ETSI EN 319 142-1[14] como *Document Time-stamp* e apresentado na figura 2.4. Para tal, foi criado um dicionário para o Time-*stamp* do documento, que é igual ao *Signature Dictionary* do PDF ISO 32000-1, embora apresente pequenas alterações nos campos SubFilter, Contents e V.

O Time-stamp do documento é adequado para aumentar a longevidade do documento assinado, bem como para fazer com que o documento seja válido para assinaturas do tipo Long Term Validation (LTV). Na figura 2.4a que se segue, pode observar-se isso. Como vemos, sobre o Document Security Store da assinatura é realizado um Time-stamp do documento.

Além disso, o documento pode ver a sua longevidade estendida se for realizado um DSS sobre o primeiro *Time-stamp* do documento e realizado um segundo *Time-stamp*. Isto pode ser feito, sempre que for necessário, criando-se um *Time-stamp* 3 , um *Time stamp* 4, e assim sucessivamente, como é exemplo a figura 2.4b.



(a) Assinatura com Time-Satmp

(b) Várias assinaturas do documento

Figura 2.4: Demonstração da assinatura do PDF com o *Document Security Store* e *Document Time-Stamp* Fonte:[14]

2.3.2 Niveis de assinatura PAdES

A realização de uma assinatura no PDF que esteja no formato PAdES definido pela ETSI, PAdES assenta nos níveis de assinatura base especificados no ETSI EN 319 142-1[14]. Este implementa 4 níveis, sendo que cada um destes extende o anterior. Em relação aos 4 níveis, apresenta-se:

1. **PAdES-B** :é o nível usado para assinaturas a curto prazo, inclui a assinatura do documento e o certificado.

2. **PAdES-T**: adionado um *time-stamp* que prova que a assinatura foi realizado numa certa data e hora.

3. **PAdES-LT**: este nível integra todos os ficheiros necessários para a validação da assinatura do documento, adicionando um dicionário VRI ao DSS. Por exemplo: os certificados da cadeia de certificação desde o do utilizador até ao do usuário, isto é, todos os certificados utilizados na validação da assinatura, como também os CRLs(Certificate revocation list) e as respostas OCSP. Este processo permite que o documento seja validado por grandes períodos de tempo, ainda que o certificado CA já não esteja disponível.

4. **PAdES-LTA**: AO *LT*, adiciona ao DSS um time-stamp do documento e os dados VRI para o TSA(Time-stamp Authority).

O PAdES, em particular o PAdES-LTA, apresenta como grande vantagem a capacidade de assinar documentos que permanecerão válidos durante muitos anos, preservando-os de futuros avanços tecnológicos. O capitulo 2.3.1.2 tem a descrição do processo para a realização da assinatura com *Time-stamp*.

2.4 Cloud Signature Consortium

O CSC é constituído por um grupo de organizações académicas e industriais que se comprometeu a criar *standards* para assinaturas digitais em *cloud*, para aplicações web e mobile, mediante o cumprimento dos mais exigentes regulamentos de assinatura do mundo. Procura, como principal objetivo, fornecer uma especificação técnica comum para ser usada no mercado global.

Este projeto do CSC foi produzido de modo a satisfazer aos requisitos complexos e rigorosos pelo qual se rege o Regulamento eIDAS para assinaturas digitais seguras. A crescente procura por soluções digitais seguras fomenta o desejo de expandir o seu impacto, esperando que este assuma uma escala global e não se restrinja, apenas,à União Europeia.

Para esta dissertação, recorreu-se à versão mais recente 1.0.4.0 (2019-06), do *standard* proposto pelo CSC, cujo título é: "Architetures and protocols for remote siganture appliations"[3]. Este documento pretende definir uma arquitetura, as tecnologias a utilizar, os protocolos de comunicação, a estrutura de dados e os requisitos técnicos necessários para criar uma API padrão de assinaturas na Cloud em concordância com a Regulamento eIDAS, suportando a criação de assinaturas avançadas e qualificadas na Cloud. Está apenas focado em arquiteturas, nas quais a chave de assinatura está remota, ou seja, na Cloud, pelo que não aborda situações em que a chave de assinatura se encontra na posse do assinante (por exemplo, em *smartcards*).

A versão estudada do documento2.4, foca-se na comunicação entre uma Aplicação de assinatura(*Signature Application* na figura 2.5) e um Remote Signing Service Provider (RSSP na figura 2.5). A aplicação de assinatura é qualquer aplicação do cliente ou serviço que pretenda realizar uma assinatura em cloud através do RSSP. O *Remote Signing Service Provider* é um serviço que gere as credenciais dos utilizadores e permite que estes criem assinaturas em *cloud*. O RSSP tipicamente providencia um RCSD e também um serviço de autenticação, este serviço gere as credenciais do utilizador que pode ser acedido pela Internet. Neste tipo de serviço as chaves de assinatura e o cadeia de certificação tem de ser criados antes das operações de assinatura começarem.

A realização de uma assinatura electrónica remota ocorre perante os seguintes cenários:

- A Aplicação de Assinatura que recebe o documento do utilizador para assinar, e se for necessário a cadeia de certificação, *revocation information* e *time-stamps* do respetivo trust service provider. Esta faz um pedido ao Remote Signing Service Provider com *Hash* do documento para este realizar a assinatura.

- O serviço de autorização pode ser através da Aplicação de Assinatura, o utilizador passa-lhe as credenciais, ou pode redirecionar para um servidor externo de autorização *OAuth* 2.0. Na maior parte dos serviços o serviço de autenticação é realizado pelo Remote Signing Service Provider.

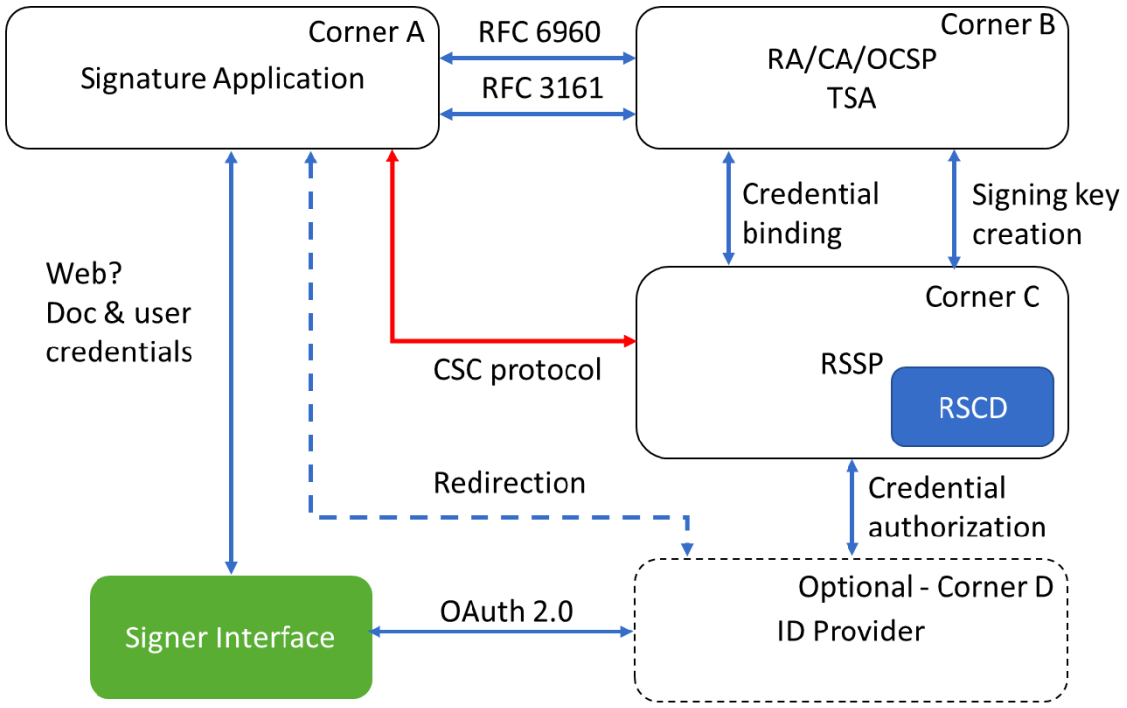


Figura 2.5: Arquitetura genérica do *standart* do CSC Fonte: [3]

2.4.1 API Protocolo CSC

As aplicações e serviços Web utilizam Application Programming Interfaces (APIs) para falar umas com as outras. Num contexto Web uma API é um conjunto de instruções e padrões para aceder uma aplicação ou serviço através da Web.

A API definida no documento [3] define o protocolo para se realizar a comunicação com o RSSP, permitindo a uma Aplicação de assinatura comunicar com este serviço remoto através da Internet. Este protocolo está definido na figura 2.5 como "CSC protocol", sendo tratado como protocolo CSC neste documento.

1. **Formato e Sintaxe**

   Os serviços Web utilizam APIs que são baseadas em padrões e protocolos como o HTTP e JSON. As funções utilizadas pelo RSSP utilizam o protocolo de comunicação HTTP RPC, com *payload* de pedido e resposta em formato JavaScript Object Notation (JSON), por essa razão todos os pedidos têm de conter no cabeçalho do HTTP o seguinte valor *Content-Type: application/json*. Todos os serviços devem utilizar a versão HTTP 1.1 ou superior.

2. **URI**

   Para aceder ao RSSP é utilizado o formato URI com o protocolo HTTP.

   O URI deve conter a versão da implementação do RSSP, no caso da especificação [3], o numero da versão deve ser **v1**. De seguida está um exemplo de como um URI deve estar definido:

   `https://service.domain.org/csc/v1/`.

   O **service.domain.org** no URI anterior é apenas um exemplo, este deve ser substituido pelo *hostname* onde o RSSP está a operar. Todas as operações que são definidas pela API descrita no documento[3] devem ser concatenadas a este URI base. Os unicos metódos que não devem seguir este URI padrão são os do OAUTH 2.0 (caso seja utilizado).

3. **Integridade e confidencialidade**

   O RSSP deve garantir sempre a integridade e confidencialidade das comunicações. De modo a assegurar essa condição, deve implementar-se o Transport Layer Security (TLS) a partir da versão 1.2. Pode, igualmente, recorrer-se a outro tipo de método, a saber: VPN.

4. **Parâmetro clientData**

   Quase todos os métodos do RSSP contêm um parâmetro extra de entrada designado por *clientData*, podendo este conter dados adicionais.

2.4.2 Autenticação e Autorização

O protocolo CSC permite definir dois tipos de autenticação e autorização. Primeiramente, temos a autorização e autenticação do serviço RSSP, que são as credenciais para proteger o serviço de acesso não

autorizado. De seguida temos a autorização das credenciais para a realização da assinatura em que o titular da chave de assinatura tem que dar autorização para a assinatura se realizar.

1. **Service authorization and authentication**

   Para aceder à API é obrigatório estar autenticado. Existem duas maneiras de realizar a autenticação:

   a) Esta pode ser concretizada pelo "RSSP", em que a Aplicação de Assinatura passa as credencias do utilizador directamente para RSSP, e este autentica o utilizador. A autenticação desta maneira deve ser realizada através do tipo: *HTTP Basic* ou *HTTP Digest*.

   b) No outro caso, a autenticação pode ser realizada pela utilização do OAuth 2.0, em que a Aplicação de Assinatura não vai colecionar os dados de autenticação do utilizador e vai redirecionar o utilizador para o serviço que o vai autenticar, representado na figura 2.5 como "Corner D".

   Quando o utilizador é autenticado como sucesso num dos dois serviços apresentados anteriormente o "RSSP"irá enviar um "acess token"para a Aplicação de Assinatura que irá garantir autorização ao utilizador para realizar o pedido à API. A partir deste momento todos os pedidos á API devem conter o *token*, no *Authorization header* do protocolo HTTP.

2. **Credential Autorization**

   Para assinar um documento remotamente é necessário uma autorização, por parte do utilizador, que deve ter um método para controlar a chave privada para realizar a assinatura. Uma autorização de dois fatores controlada pelo utilizador, poderá ser um bom exemplo. Para tal, o padrão CSC definiu três maneiras para colecionar os dados:

   a) ***Explicit authorization***: neste caso, quem coleciona os dados para enviar para o utilizador é a Aplicação de Assinatura, que recebe os dados de autenticação do utilizador, como Pin e One-time password (OTP), e envia para o "RSSP"dar a autorização da assinatura do documento.

   b) ***Implicit authorization***: neste caso, o RSSP coleciona os dados diretamente do utilizador sem utilizar a Aplicação de Assinatura como intermediário.

   c) ***OAuth 2.0 authorization***: é uma*framework* que permite ao utilizador autorizar aplicações a aceder aos seus recursos em seu nome de forma segura, sem ser necessário partilhar os seus dados de acesso.

2.4.3 Criação da assinatura remota

Cada vez que é preciso realizar uma assinatura remota, o utilizador necessita de se autenticar. A API CSC permite realizar uma ou mais assinaturas por sessão de assinatura. Assim, é permitido:

1. A assinatura remota de uma *Hash* apenas.
2. A assinatura de várias *Hahs* na mesma operação de assinatura.
3. A assintura de várias *Hahs* utilizadas em várias operações de assinatura, na mesma sessão.

2.4.4 *Status Code* e Mensagens de erro

A especificação define uma lista dos *status code* do protocolo HTTP que se pode utilizar numa aplicação de assinatura remota, conforme a seguinte tabela:

Standard Status Code	Description
200 OK	Response to a successful API method request.
204 No Content	Response to a successful API method request in case no content is returned.
302 Found	Response used to redirect the user to an OAuth 2.0 authorization endpoint.
400 Bad Request	Returned due to unsupported, invalid or missing required parameters.
401 Unauthorized	Returned when a bad or expired authorization token is used.
429 Too Many Requests	Returned when a request is rejected due to rate limiting.
500 Internal Server Error	Returned when the server encounters an unexpected condition.
501 Not Implemented	Returned when an unimplemented method is requested.
503 Service Unavailable	Returned when the server is currently unable to handle the request due to temporary overloading or maintenance conditions.

Tabela 2.1: Lista de *status code* permitidos no protocolo CSC Fonte:2.5

Em relação à mensagens de erro, sempre que existe um erro na aplicação deve recorrer-se à especificação, onde se encontram definidas as mensagens de erro, caso não seja possível utilizar estas mensagens padrão é possível utilizar outras mensagens que não estejam predefinidas na tabela 2.2. Neste caso, a aplicação deverá devolver um JSON apenas com dois objetos: o "error"com a mensagem geral de erro e o "error_description", que deve conter uma mensagem mais especifica do erro.

Error	Error Description
invalid_request	The request is missing a required parameter, includes an invalid parameter value, includes a parameter more than once, or is otherwise malformed.
unauthorized_client	The client is not authorized to use this method.
access_denied	The user, authorization server or remote service denied the request.
unsupported_response_type	The authorization server does not support obtaining an authorization code using this method.
invalid_scope	The requested scope is invalid, unknown, or malformed.
server_error	The authorization server encountered an unexpected condition that prevented it from fulfilling the request.
temporarily_unavailable	The authorization server is currently unable to handle the request due to a temporary overloading or maintenance of the server.
expired_token	The access or refresh token is expired or has been revoked.
invalid_token	The token provided is not a valid OAuth access or refresh token.

Tabela 2.2: Lista de mensagens de erro permitidas no protocolo CSC Fonte:2.4

2.4.5 Métodos da API

Neste ponto, importa referir os métodos que um serviço de assinatura deve fornecer, os quais se encontram expostos na seguinte tabela. Destaca-se que o método /info é obrigatório, enquanto que os restantes métodos são opcionais.

Todos os Métodos que o CSC permite	
Método	**Descrição**
info	Trata-se de um método obrigatório, responsável por concretizar o retorno da informação com todos os métodos que a RSSP contem.
auth/login	Este método tem a responsabilidade de realizar a autenticação no RSSP. A aplicação de assinatura deve enviar para o RSSP os dados de autenticação do utilizador(userID e password). A autenticação pode ser feita através de *HTTP Basic Authentication* ou *HTTP Digest Authentication*. O *RSSP* tem de retornar uma *acess token*, para ser utilizado pelo utilizador em todos os pedidos subsequentes que realizar. O *acess token* tem de estar sempre no campo "Authetication Bearer" do cabeçalho HTTP. Se a aplicação permitir o uso de um *refresh token* para estender a autenticação realizada com sucesso, este permite a sua utilização em sessões seguintes sem ter de se autenticar novamente.
auth/revoke	Serve para anular o *acess token* ou o *refresh token*, que foi obtido.
credentials/list	Retorna a lista de credenciais(credentialID) que o utilizador tem naquele RSSP. Cada utilizador pode ter uma ou várias credenciais no RSSP. Isto é, cada credencial corresponde ás várias chaves de assinatura que o utilizador pode utilizar.
credentials/info	Neste método, o utilizador envia o *credentialID* recebido no *credentials/list* e recebe a informação acerca do modo como a assinatura vai ser realizada, o certificado de chave pública associado e, ainda, a sua cadeia de certificação. Pode, do mesmo modo, conter informação sobre o mecanismo de autorização que permite o acesso às credenciais para a assinatura remota. A descrição do PIN ou OTP, se este for utilizado vem incorporada neste pedido.
credentials/authorize	Neste ponto, é autorizado o acesso à realização da assinatura. São também enviados os mecanismos de autorização associados (PIN ou OTP). É enviada a *hash* para assinar e o número de assinaturas. Este método retorna um Signature Activation Data (SAD), o qual é requerido quando é chamado o método signatures/signHash, descrito posteriormente.
credentials/ extendTransaction	Aumenta a validade do período de transição quando são retornadas várias assinaturas através do método signatures/signHash. O SAD pode perder a sua validade, retornando um SAD novo.
credentials/sendOTP	Este método gera um OTP, entregue ao utilizador por um mecanismo de comunicação seguro.

signatures/signHash	Calcula a assinatura digital remota para uma ou múltiplos valores da *hash*. Retornando as assinaturas realizadas.
signatures/timestamp	Gera um *token* de *time-stamp* para o respetivo *hash*, o qual tem de ser gerado por uma *Time Stamping Authority*.
oauth2/authorize	Inicia o processo de autenticação através do *OAuth 2.0*
oauth2/token	Obter o *acess token* ou *refresh token* do *OAuth 2.0*
oauth2/revoke	Serve para anular as credenciais do *OAuth 2.0*.

2.5 Chave Móvel Digital

A Chave Móvel Digital (CMD)[6] é um meio de autenticação que permite a associação entre um número de telemóvel e o número de identificação civil (NIC) para um cidadão português. No caso de se tratar de um cidadão estrangeiro, é possível recorrer-se ao número de passaporte ou título de residência. Para além disso, possibilita, ainda, a assinatura digital qualificada de documentos, através de uma password previamente escolhida e o respetivo código de segurança.

2.5.1 Assinatura com Chave Móvel Digital

Para realizar a assinatura com a chave móvel digital o utilizador necessita:

1. ter a Chave Móvel Digital ativada;
2. ter a assinatura digital da CMD ativada;
3. código PIN de assinatura da CMD (que pode ser diferente do código PIN da CMD).

Após a realização dos três processos enumerados, pode-se, então, proceder à assinatura a partir da CMD. Na realização da assinatura, o utilizador terá apenas de concretizar dois passos para autenticação, a saber: inserir os dados para autenticação, o número de telemóvel e o PIN e, de seguida, inserir o OTP que recebeu no telemóvel.

O utilizador que pretenda realizar uma assinatura com a chave móvel digital pode realizá-la através do site da autenticação.gov, já que este permite que o utilizador assine documentos com o formato PDF ou através da aplicação Autenticação.Gov.

A assinatura através do site é um processo simples. Para tal, o utilizador começa por escolher o documento PDF que pretende assinar. De seguida, caso queira que a assinatura fique visível no documento, deverá selecionar o local. Mais tarde, avança-se para a parte da autenticação, como foi descrito anteriormente.

Se o utilizador preferir realizar a assinatura através da aplicação desktop, o processo é similar ao que é efetuado através do website. Contudo, neste caso, é possível adicionar um selo temporal.

[6] https://www.autenticacao.gov.pt/a-chave-movel-digital

Relativamente às assinaturas do tipo PAdES, é importante esclarecer que tanto a assinatura no website como a assinatura sem selo temporal, concretizada através da aplicação, não representam nenhum nível do tipo PAdES. Por outro lado, a assinatura realizada através da aplicação pode conter selo temporal e validação a longo parzo, por isso, é uma assinatura do tipo PAdES com o perfil PAdES-B, PAdES-T ou PAdES-LTA, referido neste documento na secção 2.3.2.

Figura 2.6: Assinatura CMD através da aplicação Desktop Autenticação.Gov

2.5.2 Especificação

Podem ser realizadas chamadas para uma API da chave móvel digital. A este respeito, interessa destacar que esta especifica as operações para a realização da assinatura de uma ou várias hash's, assim como para a obtenção da cadeia de certificados. Nesta secção, serão especificadas as operações que são disponibilizadas pelo CMD. Para que os sistemas externos efetuem as assinaturas, os métodos disponibilizados são os seguintes:

1. **GetCertificate**: método no qual se insere o identificador da aplicação e do utilizador e se obtém como retorno o certificado do cidadão em *String*.

GetCertificateRequest			
Parâmetros	**Tipo**	**Obrigatório?**	**Descrição**
ApplicationId	byte[]	Sim	Identificador da aplicação que efetua o request
UserId	String (cifrado)	Sim	Identificador da conta do utilizador (ex.: N° de Telemóvel: "+351 966666666")

GetCertificateResponse			
Parâmetros	**Tipo**	**Obrigatório?**	**Descrição**
certificate	string	Sim	Cadeia de certificação da chave de assinatura

2. **SCMDSign** : método que se utiliza sempre que se pretende assinar um documento. Os parâmetros descritos na tabela, que a seguir se apresenta, correspondem aos de entrada. Esta operação tem como parâmetro de saída o SignStatus, que contém o *ProcessId* em String, o qual será posteriormente utilizado no método *ValidateOtp*.

SignRequest			
Parâmetros	**Tipo**	**Obrigatório?**	**Descrição**
ApplicationId	byte[]	Sim	Identificador da aplicação que efetua o request
DocName	string	Não	Nome do Documento ou identificador para permitir ao Cidadão identificar o ato que vai originar a assinatura
Hash	byte[]	Sim	Hash da informação sobre a qual vai ser gerada a assinatura
Pin	String(cifrado)	Sim	Código PIN do utilizador
UserId	String (cifrado)	Sim	Identificador da conta do utilizador (ex.: N° de Telemóvel: "+351 966666666")

3. **SCMDIMultipleSign**: Este método é utilizado sempre que se pretende assinar vários documentos. Apresenta como parâmetro de entrada duas estruturas: a primeira é composta pelo ApplicationId, Pin e UserId, que correspondem aos parâmetros expostos na estrutura anterior do *SCMDSign*; a segunda corresponde a uma lista da estrutura *HashStructure*, descrita na tabela a seguir exposta. Os parâmetros de resposta são os mesmos da *SCMDSign*.

HashStructure			
Parâmetros	**Tipo**	**Obrigatório?**	**Descrição**
Hash	byte[]	Sim	Hash da informação sobre a qual vai ser gerada a assinatura
Name	string	Sim	Nome do documento
id	String	Sim	identificador do documento

4. **ValidateOtp**:valida o código que foi enviado para o utilizador e devolve o hash assinado, quer seja para um só documento ou para uma lista deles. Recebe, além do código do utilizador, o processId e o applicationId.

ValidateOptRequest			
Parâmetros	**Tipo**	**Obrigatório?**	**Descrição**
Code	string	Sim	Otp enviado para o telemóvel do utilizador
processId	string	Sim	processId que foi recebido no SCMDSign ou SCMDIMultipleSign.
applicationId	byte[]	Sim	Identificador dado pela AMA

ValidateOtpResponse			
Parâmetros	**Tipo**	**Obrigatório?**	**Descrição**
Signature	byte[]	Sim	Assinatura em caso do pedido ser realizado com SCMDSign
ArrayOfHashStructure	List<HashStructure>	Sim	Resposta em caso do pedido SCMDIMultipleSign

2.6 Cartão de Cidadão

O Cartão de Cidadão (CC)[7] permite ao titular realizar uma assinatura digital qualificada, que pode ser verificada através dos meios indicados (CRL e OCSP) no certificado digital pessoal do cidadão. Para assinar com o cartão de cidadão, este precisa de estar conectado ao dispositivo que o assinante pretende utilizar. Além disso, deve ter instalado o *plugin* do cartão de cidadão[8]. Importa frisar que, ao contrário da CMD, o CC é um *smart card* e requer-se que esteja na posse do assinante.

2.6.1 Assinatura Cartão de Cidadão

Para que o utilizador realize a assinatura com o cartão de cidadão são necessários os seguintes requisitos:

1. ter o certificado de assinatura digital do cartão ativada;
2. leitor de cartões smartcard;
3. código PIN de assinatura.

Para realizar a assinatura com o cartão de cidadão a AMA apenas fornece a aplicação Autenticação.Gov. Por isso, o utilizador tem de conectar o smartcard ao computador e, depois disso, inicia-se o processo. Assim, o utilizador pode escolher o ficheiro que pretende assinar. Caso pretenda que a assinatura fique visível, deve definir o local onde esta se deve encontrar no PDF. Após isso, deve inserir o PIN da assinatura digital do Cartão de Cidadão e, por fim, assinar.

Em relação às assinaturas do tipo PAdES no cartão de cidadão apenas é possível realizar a assinatura PAdES-B, PAdES-T e PAdES-LTA a partir da aplicação Auntenticacao.gov. Não é possível assinar com o cartão de cidadão através de Websites.

2.6.2 Plugin

Relativamente à utilização em websites, é fornecido um Plugin para instalar no computador do utilizador, que permite que se comunique com o cartão de cidadão através de uma página Web. No entanto, apenas é possível realizar o procedimento de autenticação com o Cartão de Cidadão. A utilização deste Plugin é compatível com os seguintes navegadores: Google Chrome, Mozilla Firefox, Safari, Opera, Microsoft Edge e Internet Explorer.

O Plugin que está instalado no computador do utilizador funciona como um intermediário entre o Website e o cartão e só funciona com pedidos do localhost.

[7] https://www.autenticacao.gov.pt/web/guest/o-cartao-de-cidadao
[8] https://autenticacao.gov.pt/fa/ajuda/autenticacaogovpt.aspx

2.6.3 Selo Temporal

O objetivo de um selo temporal é garantir que um documento existia num determinado momento do tempo.

Para a realização de um selo temporal é necessário fazer um pedido a um Servidor de Validação Cronológica. Deste modo, este irá recebê-lo com os dados que se pretende assinar, gerando um documento assinado com a data e horas atuais legais. Assim, este selo é a garantia que o documento existia num determinado momento. Além de certificar a data/hora da assinatura, este selo assegura, ainda, a integridade e não repúdio do conteúdo.

O Instituto dos Registos e do Notariado I.P. fornece o Serviço de Validação Cronológica do Cartão de Cidadão.

O URL para realizar o pedido para a elaboração do selo temporal é o seguinte:
`http://ts.cartaodecidadao.pt/tsa/server`.

Capítulo 3

Solução PADES SERVER SIGNER

Neste capitulo é apresentada a solução para o serviço *PAdES Server Signer*, que tem como objetivo padronizar, de acordo com o CSC, o modo de comunicação dos serviços de assinatura(CC e CMD) e da aplicação de assinatura do lado do cliente, com a finalidade de realizar uma assinatura digital nestes serviços. Para além disso, também permite realizar a assinatura de um documento PDF no formato PAdES. A solução desenvolvida também contém uma Aplicação Web com uma interface para o utilizador poder realizar a assinatura no formato PAdES. Neste capitulo, primeiramente é apresentada a arquitetura geral da do PAdES Server Signer, depois sendo apresentado a proposta de solução para a CMD e CC, e por fim a Aplicação *Web*.

3.1 Arquitetura

Atendendo ao esquema da figura 3.1, verifica-se que o *PAdES Server Signer* se divide, essencialmente, nos dois serviços de assinatura que são o CC e a CMD. Para cada um destes serviços, existem dois tipos de funcionalidades com finalidades diferentes. Assim, existe um processo que tem como objetivo assinar uma ou mais Hash's, seguindo o protocolo de comunicação definido e padronizado pelo CSC, explicados na secção 2.4. Além deste, existe um outro processo onde estão os métodos que têm como objetivo assinar o documento PDF no formato PAdES, segundo um dos quatro níveis definidos na secção 2.3.2, métodos esses que extendem a especificação CSC.

Em relação à assinatura através da CMD, esta é realizada em *Cloud*. Desta forma, o *PAdES Server Signer* necessitará de comunicar com o servidor de assinatura da Chave Móvel Digital para realizar assinaturas neste serviço, como descrito na figura 3.1. O *PAdES Server Signer* funciona assim como uma nterface CSC para o CMD. Os métodos para esta comunicação estão definidos na secção 2.5.2.

Pelo contrário, o CC está na posse do assinante, ou seja, está conectado ao dispositivo do utilizador. Por esse motivo, para realizar a assinatura através do CC é necessário que o utilizador tenha um *Plugin*

instalado no seu dispositivo que permita que uma aplicação de assinatura do tipo *Desktop* ou *Web* consiga comunicar com o CC. Este *Plugin*, para além de receber pedidos de aplicação de assinatura, também comunica com o servidor *PAdES Server Signer* para trocar informações sobre a assinatura.

Como existe necessidade de guardar os dados para ter acesso a eles nos diferentes métodos que a aplicação permite, foi essencial ter uma Base de Dados para essa finalidade.

Dado que as assinaturas com o nível PAdES-T, PAdES-LT e PAdES-LTA necessitam de um time-stamp, é preciso que o servidor comunique com um servidor time-stamp, para realizar estes níveis de assinaturas.

Por último, foi também realizada um Aplicação Web com uma interface para o utilizador poder realizar a assinatura no formato PAdES, tanto na CMD, como no CC. Esta aplicação visa estabelecer a comunicação entre o utilizador e o servidor PAdES Server Signer.

É de notar que no esquema da arquitetura geral está representado como Aplicação de Assinatura, pois qualquer aplicação, seja ela Web, Desktop ou Mobile, pode comunicar com o servidor PAdES e realizar a assinatura, sendo esta nomenclatura mais geral.

É importante destacar que na figura 3.1, a aplicação de assinatura só necessita de comunicar com Plugin e ter o CC conectado ao dispositivo, para realizar a assinatura através do CC. Nos restantes casos, é realizada a assinatura a partir da CMD no formato PAdES.

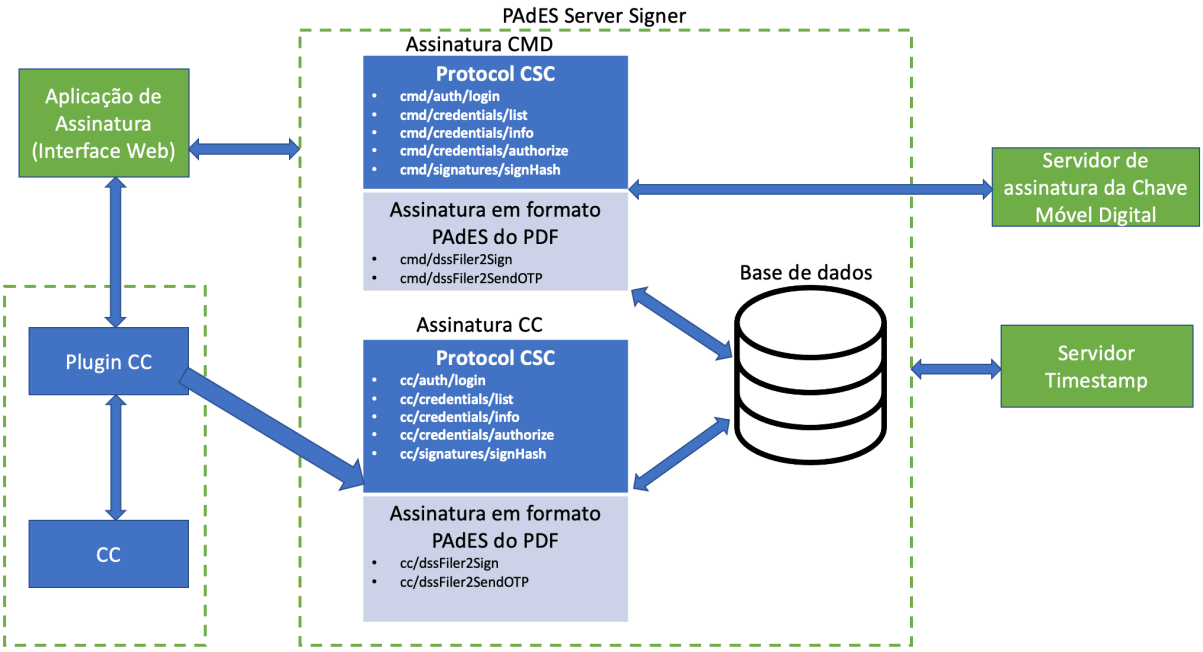


Figura 3.1: Esquema base da arquitetura do sistema

3.2 CMD

Neste capitulo, serão apresentados os métodos da API necessários para realizar a assinatura com a CMD. Este processo pode ser concretizado através de dois caminhos, os quais estão descritos nas figuras 3.2 e 3.3. Nas figuras, pode verificar-se que os dois primeiros métodos são comuns aos dois processos, sendo estes as fases para autenticar e pedir as credenciais para os procedimentos que serão realizados. Após a concretização destes processos, o utilizador pode seguir um dos dois caminhos para efetuar a assinatura. O utilizador seleciona o caminho pretendido, mediante o seu objetivo: se pretende assinar as hashs ou, por outro lado, se tenciona enviar o documento PDF, que será assinado, e recebê-lo no formato PAdES. De realçar que, enquanto a primeira opção referida é realizada de acordo com os métodos e parâmetros definidos no documento do CSC, a segunda foi idealizada e arquitetada com o propósito de assinar documentos PDF em formato PAdES. Por essa razão, já não faz parte do padrão CSC. No Diagrama de Sequência da figura 3.4, está descrito todo o processo, onde se pode verificar a divisão entre estes dois caminhos.

Em relação à assinatura que está formatada segundo o padrão do CSC, na figura 3.2, realça-se que este processo requer a comunicação com o servidor de assinatura da CMD. Como demonstrado no diagrama de sequência da figura 3.4, os métodos /credentials/authorize e /signatures/signHash comunicam com o servidor da CMD, tendo em conta os métodos designados na secção 2.5.2. Esta foi a correspondência idealizada entre o CMD e o *PAdES Server Signer*.

O método /credentials/authorize recorre ao método da CMD - "CMDSign- para uma assinatura ou ao - "CMDMultipleSign- para um número de assinaturas que pode ir de duas a dez. O /signatures/signHash chama o método validateOtp da CMD.

O método /credentials/info tem como objetivo retornar a cadeia de certificação ou somente o certificado de assinatura da chave de assinatura. É um método prescindível, pois a sua utilização não é necessária para realizar o processo de assinatura. No entanto, é um método existente e pode ser utilizado, na ordem descrita na figura 3.2.

Figura 3.2: Caminho do pedidos utilizados pelo PAdES Server Signer

Por outro lado, decidiu-se adaptar à assinatura do documento PDF. Assim, a API permite que o utilizador concretize os dois primeiros métodos do padrão do CSC e, posteriormente, possa seguir o caminho da assinatura do PDF. Para tal, foram criados dois métodos adicionais: o método 3 -"/dssFiler2Sign- e 4 -"dssFiler2SendOTP- da figura 3.3. Estes métodos foram idealizados para receber o ficheiro PDF e fazer a sua assinatura no formato PAdES. Estes métodos utilizam os métodos /credentials/authorize e /signatures/signHash para realização da assinatura da hash do documento.

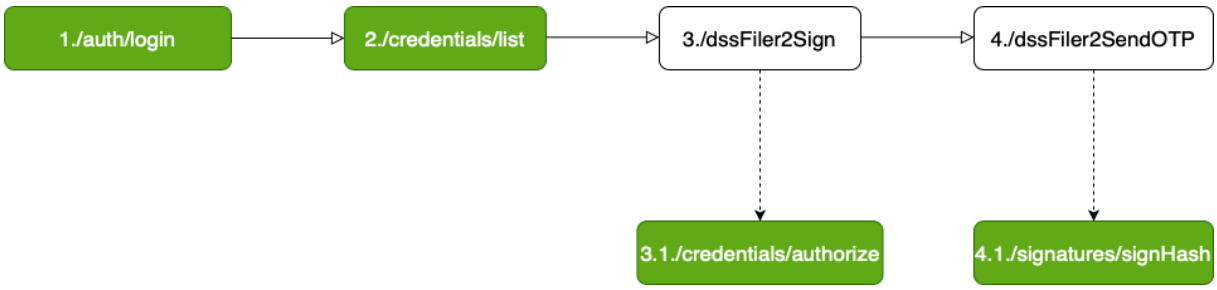


Figura 3.3: Caminho do pedidos utilizados pelo PAdES Server Signer para assinar o PDF

No diagrama de sequência B.2, disponível em apêndice, está apresentado o processo a usar para quando se pretende adquirir apenas as informações sobre a assinatura, como a cadeia de certificação, por exemplo. Este diagrama termina no método /credentials/info, que contém toda esta informação acerca da assinatura.

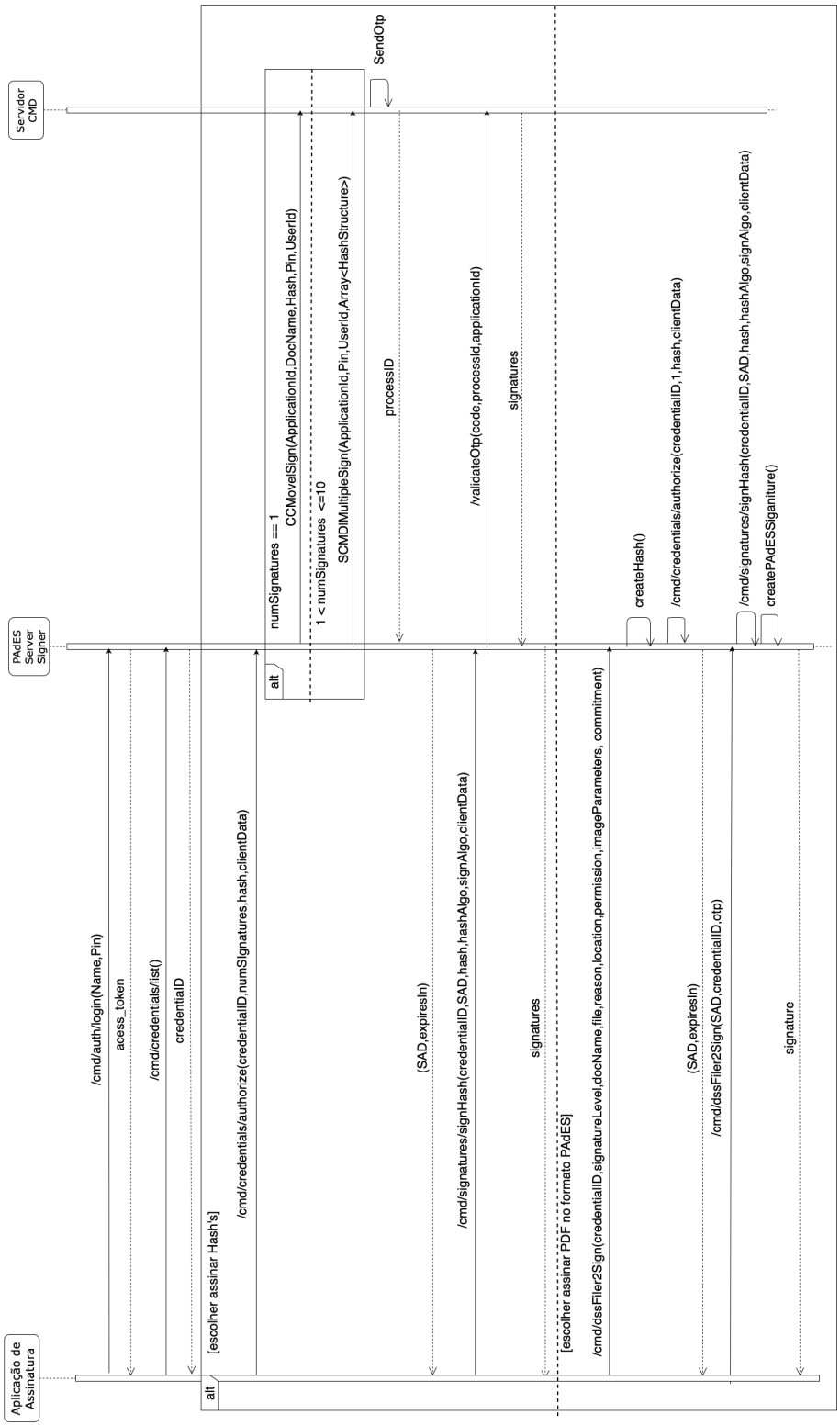


Figura 3.4: Diagrama de sequência do processo de assinatura do CMD no PAdES Server Signer

De seguida, serão apresentados os métodos e os campos que estes contêm, tanto os dados do pedido como os dados da resposta. Para cada método, serão expostas tabelas com os campos que devem ser feitos, quer no pedido como na resposta do método.

3.2.1 Adaptação ao CSC

1. **auth/login**

   Este método inicial é utilizado para se obter um token de autorização. O utilizador recorre à HTTP Basic Authentication com os dados de autenticação do CMD, isto é, o número de telemóvel e o o pin, servindo-se do access_token para utilizar nos métodos seguintes.

Atributo	Obrigatoriedade	Valor	Descrição
phoneNumber	Sim	String	O número do telemóvel que o utilizador tem na Chave Móvel Digital.
pin	Sim	String	O pin de 4 a 8 dígitos da Chave Móvel Digital.

Tabela 3.1: Pedido auth/login

Atributo	Obrigatoriedade	Valor	Descrição
access_token	Sim	String	O Token de curta duração (15 minutos), que vai ser utilizado para o utilizador fazer pedidos á API, este token deve ser usado no campo do cabeçalho "Authorizarion Bearer" do HTTP. Caso o token esteja expirado irá ser enviada uma mensagem de erro.

Tabela 3.2: Resposta auth/login

2. **credentials/list**

   Este método é chamado unicamente com o *token* retornado no método auth/login, inserido no cabeçalho *authorization* do pedido HTTP, e retorna a credencial associada ao utilizador para realizar os métodos de assinatura.

Atributo	Obrigatoriedade	Valor	Descrição
credentialID	Sim	String	A credencial associada ao utilizador, cuja função consiste em assegurar que a assinatura realiza os métodos de assinatura.

Tabela 3.3: Resposta credentials/list

3. **credentials/info**

Neste método, as informações a respeito da assinatura são retornadas. Como indicado na tabela, podem ler-se as principais informações sobre a chave de assinatura. O utilizador pode receber a cadeia de certificação ou somente o certificado de assinatura. Para efetivar a sua escolha, terá que selecionar o campo "certificates"do pedido.

Atributo	Obrigatoriedade	Valor	Descrição
credentialID	Sim	String	A credentialID pedida pelo utilizador no método cmd/credentials/list.
certificates	Sim	String "none|single|chain"	Especifica quais os certificados da cadeia de certificação que devem ser retornados. • "none": Nenhum certificado retornado. • "single": Apenas o certificado de assinatura. • "chain": Toda a cadeia de certificação.

Tabela 3.4: Pedido credentials/info

Atributo	Obrigatoriedade	Valor	Descrição
key/status	Sim	String “enable|disable”	O estado da chave de assinatura. Nesta aplicação, ao poder ser usada para assinar documentos, esta estará “enable”.
key/algo	Sim	Array of String	A lista de OIDs dos algoritmos de assinatura que são suportados. Neste caso, vai ser retornado o OID do RSA-SHA256.
key/len	Sim	Number	O comprimento da chave em bits.
cert/status	Não	String “valid”	O estado da validade do certificado de assinatura.
cert/certificates	Sim	Array of String	O certificado de assinatura ou toda a cadeia de certificação, consoante o campo que o utilizador selecionou no campo “certificates” do pedido.
authMode	Sim	String “implicit”	Especifica os modos de autorização da assinatura.
SCAL	Não	String “2”	A CMD é gerada de acordo com o SCAL 2.
multisign	Sim	Number	Número de assinaturas que é possível realizar em simultâneo.

Tabela 3.5: Resposta credentials/info

4. **credentials/authorize**

   Neste campo é efetuado o pedido de acesso para realizar a assinatura. O input deste método consiste nas informações a respeito da assinatura, assim como as hashs para serem assinadas pela Chave Móvel Digital. A resposta irá conter um SAD que é o ”processId”,ou seja, um identificador do

processo que o servidor da Chave Móvel Digital envia quando é realizado o pedido de assinatura. O utilizador, na sequência deste processo, irá ainda receber um OTP no seu telemóvel, no número que inseriu no "auth/login". Estes dois códigos deverão ser submetidos no último processo, ou seja, no "signatures/signHash".

Atributo	Obrigatoriedade	Valor	Descrição
credentialID	Sim	String	A credentialID pedida pelo utilizador no método cmd/credentials/list.
numSignatures	Sim	Number	O número de assinaturas autorizadas.
hash	Sim	Array of String	Os valores das hashs que vão ser assinadas em Base64.
clientData	Não	String	Array com o nome dos documentos na mesma ordem que as respetivas hashs.

Tabela 3.6: Pedido credentials/authorize

Atributo	Obrigatoriedade	Valor	Descrição
SAD	Sim	String	A Signiture Activation Data (SAD) que vai ser usado como input no método signitures/signHash.
expiresIn	Não	Number	O tempo até expirar o SAD em segundos. Neste caso é de 300.

Tabela 3.7: Resposta credentials/authorize

5. **signatures/signhash**

Por fim, depois de inseridas todas as credenciais, como o SAD, credentialID e o código recebido no telemóvel, o utilizador pode escolher os algoritmos de assinatura e de *hash*. No entanto, para a CMD, existe apenas uma escolha, uma vez que só é dada uma hipótese para cada um destes campos. Depois de estar tudo correto, as assinaturas são devolvidas ao utilizador em Base64. Para adaptar os pedidos ao servidor de assinatura da Chave Móvel Digital à arquitetura do CSC foi necessário que o OTP fosse inserido no campo "clientData".

Atributo	Obrigatoriedade	Valor	Descrição
credentialID	Sim	String	A credentialID pedida pelo utilizador no método cmd/credentials/list.
SAD	Sim	String	O SAD retornado no cmd/credentials/authorize.
hash	Sim	Array of String	Os valores das hashs que vão ser assinadas em Base64.
hashAlgo	Sim	String	O OID do algoritmo utilizado para calcular a hash. Neste caso, é o SHA256.
signAlgo	Sim	String	O OID do algoritmo utilizado para realizar a assinatura, que é o mesmo do credentials/info.
clientData	Sim	String	O OTP enviado para o número de telemóvel do utilizador.

Tabela 3.8: Pedido signatures/signhash

Atributo	Obrigatoriedade	Valor	Descrição
signatures	Sim	Array of String	As assinaturas das hash(s) em Base64.

Tabela 3.9: Resposta signatures/signhash

3.2.2 Assinatura do PDF no formato PAdES

1. **dssFiler2Sign**

   Esta situação verifica-se quando o utilizador opta por assinar um documento PDF a partir da CMD, selecionando o nível de assinatura PAdES que pretende. O ficheiro é enviado em Base64, tal como o próprio nome. Deste modo, o utilizador receberá o SAD para inserir no /dssFiler2SendOTP. Assim que este processo finalizar, o utilizador receberá, ainda, um OTP no seu telemóvel para inserir no passo seguinte. Neste pedido também se podem definir as permissões para alterações do PDF. Existem três tipos de permissões:

a) CHANGES_PERMITTED - Além das que estão na número dois, é permitido criar, apagar e modificar anotações.

b) MINIMAL_CHANGES_PERMITTED - As alterações permitidas são o preenchimento de formulários, realizar assinaturas e modificar os modelos das páginas. Qualquer alteração, para além das que foram mencionadas, irá anular a assinatura.

c) NO_CHANGE_PERMITTED - Não são permitidas alterações, qualquer alteração irá invalidar a assinatura.

Atributo	Obrigatoriedade	Valor	Descrição
credentialID	Sim	String	A credentialID pedida pelo utilizador no método cmd/credentials/list.
signatureLevel	Sim	String “pades_t|pades_lt|pades_lta|pades_b”	O nível de assinatura Pades que pretende ter no PDF. Por default o valor é pades_b.
docName	Sim	String	O nome do documento.
file	Sim	String	O ficheiro PDF em Base64.
reason	Não	String	O motivo para a realização da assinatura do documento.
location	Não	String	A localização onde foi realizada a assinatura.
imageParameters	Não	Dictionary	Dicionário com a imagem e o local onde esta vai estar.
permission	Sim	String “CHANGES_PERMITTED| MINIMAL_CHANGES_PERMITTED| NO_CHANGE_PERMITTED”	Permissões para alteração no PDF

Tabela 3.10: Pedido dssFiler2Sign

Atributo	Obrigatoriedade	Valor	Descrição
SAD	Sim	String	A Signiture Activation Data (SAD) que vai ser usado como input no método signitures/signHash.
expiresIn	Não	Number	O tempo até expirar o SAD em segundos. Neste caso é de 300.

Tabela 3.11: Resposta dssFiler2Sign

2. **dssFiler2SendOTP**

   Neste local, o utilizador envia o OTP, recebido no seu telemóvel, bem como o SAD, recebido no método anterior. De acordo com as especificações e com o tipo de assinatura que delegou no método anterior /dssFiler2Sign., irá receber o PDF assinado no formato PAdES

Atributo	Obrigatoriedade	Valor	Descrição
otp	Sim	String	O OTP enviado para o utilizador.
credentialId	Sim	String	A credentialID pedida pelo utilizador no método cmd/credentials/list.
SAD	Sim	String	O SAD retornado no /dssFiler2Sign.

Tabela 3.12: Pedido dssFiler2SendOTP

Atributo	Obrigatoriedade	Valor	Descrição
signatures	Sim	String	Documento PDF assinado em Base64.

Tabela 3.13: Resposta dssFiler2SendOTP

3.2.3 Comunicação com CMD

Para que a assinatura fosse efetuada pela Chave Móvel Digital, a comunicação entre a aplicação e o serviço de assinatura qualificada da Chave Móvel Digital, foi indispensável. Esse serviço disponibiliza quatro operações: a operação de assinatura de uma hash (SCMDSign); a operação de assinatura de várias hash (SCMDMultipleSign); a operação de validação de código de segurança OTP (ValidateOtp) e, por fim, a operação de obter a cadeia de certificação (GetCertificate). Note-se que, todas estas operações são fornecidas no protocolo para troca de informações SOAP baseado em XML.

1. **SCMDSign**

   Sempre que se pretende obter uma assinatura de uma única hash, este é o método selecionado.

   As informações enviadas neste pedido consistem na autenticação do utilizador, ou seja, o identificador da conta do utilizador, o número de telemóvel, o pin, a hash para assinar em bytes e o nome do documento. Realça-se que este último não é obrigatório.

   Se todo o processo estiver correto, a resposta obtida será um processId, que servirá de identificação no método validateOtp. A isto, acrescenta-se o envio de um OTP para o número de telemóvel do utilizador.

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ns1="http://Ama.Authentication.Service/">
<SOAP-ENV:Header/>
   <SOAP-ENV:Body>
      <ns1:CCMovelSign>
         <ns1:request xmlns:ns2="http://schemas.datacontract.org/2004/07/Ama.Structures.
             ↪ CCMovelSignature">
            <ns2:ApplicationId>APPLICATIONID</ns2:ApplicationId>
            <ns2:DocName>PADESDSSTESTER</ns2:DocName>
            <ns2:Hash>
                ↪ MDEwDQYJYIZIAWUDBAIBBQAEIALMe7117zrwDO8AaHfVM8BBlhf1D8YpVAyj31PVEj1M
                ↪ </ns2:Hash>
            <ns2:Pin>1111</ns2:Pin>
            <ns2:UserId>+351 966666666</ns2:UserId>
         </ns1:request>
      </ns1:CCMovelSign>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
<s:Body>
   <CCMovelSignResponse xmlns="http://Ama.Authentication.Service/">
      <CCMovelSignResult xmlns:a="http://schemas.datacontract.org/2004/07/Ama.Structures
          ↪ .CCMovelSignature" xmlns:i="http://www.w3.org/2001/XMLSchema-instance">
         <a:Code>200</a:Code>
```

```
6           <a:Field i:nil="true"/>
7           <a:FieldValue i:nil="true"/>
8           <a:Message>Success! Next step: invoke ValidateOTP</a:Message>
9           <a:ProcessId>33aecd2c-7732-4800-b73d-eabea4c58418</a:ProcessId>
10       </CCMovelSignResult>
11    </CCMovelSignResponse>
12 </s:Body>
13 </s:Envelope>
```

2. SCMDMultipleSign

   Sempre que urge a necessidade de se assinar, simultaneamente, múltiplas hashs, este deverá ser o método usado. No SCMDMultipleSign a autenticação é concretizada da mesma forma que o anterior. Contudo, como têm que ser enviadas várias hashs, é criada uma estrutura HashStructure que contém a Hash em Bytes, o nome do documento e um identificador em String. Assim, é inserido um array com várias HashStructure, no qual estão contidas as várias hashs para serem assinadas. A resposta é a mesma que no método anterior.

3. **GetCertificate** Nesta operação, apenas é necessário inserir o identificador do utilizador e será automaticamente retornada a cadeia de certificação.

4. **ValidateOtp**

   Neste método procede-se ao ao envio do processId, retornado anteriormente, com o OTP que o utilizador recebeu no seu telemóvel.

   A resposta difere em função do número de assinaturas. Se for somente uma assinatura, a assinatura estará no campo signature. Em situações em que o número de assinaturas é superior a um, no campo signatures será possível encontrar, no seu interior, um array de estruturas. Neste array, encontraremos as assinaturas relacionados com o seu identificador e o nome enviado no método SCMDMultipleSign.

```
1  <SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:ns1=
       ↪ "http://Ama.Authentication.Service/"><SOAP-ENV:Header/>
2     <SOAP-ENV:Body>
3        <ns1:ValidateOtp>
4           <ns1:code>601237</ns1:code>
5           <ns1:processId>33aecd2c-7732-4800-b73d-eabea4c58418</ns1:processId>
6           <ns1:applicationId>APPLICATIONID</ns1:applicationId>
7        </ns1:ValidateOtp>
8     </SOAP-ENV:Body>
9  </SOAP-ENV:Envelope>
```

```
1  <s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
2  <s:Body>
```

```
3     <ValidateOtpResponse xmlns="http://Ama.Authentication.Service/">
4        <ValidateOtpResult xmlns:a="http://schemas.datacontract.org/2004/07/Ama.Structures
             ↪ .CCMovelSignature" xmlns:i="http://www.w3.org/2001/XMLSchema-instance">
5           <a:ArrayOfHashStructure i:nil="true"/>
6           <a:Signature>SIGNATURE</a:Signature>
7           <a:Status><a:Code>200</a:Code>
8           <a:Field i:nil="true"/><a:FieldValue i:nil="true"/>
9           <a:Message>OTP code is valid</a:Message>
10          <a:ProcessId>33aecd2c-7732-4800-b73d-eabea4c58418</a:ProcessId>
11          </a:Status><a:certificate i:nil="true"/>
12       </ValidateOtpResult>
13    </ValidateOtpResponse>
14 </s:Body>
15 </s:Envelope>
```

3.3 Cartão de Cidadão

Nesta capitulo são apresentados os métodos da API para realizar a assinatura no CC. Este é um meio mais complexo para fazer a assinatura através de um servidor, uma vez que é um dispositivo que está na posse do assinante. Para a realização desta parte do projeto, tiveram de ser implementadas duas API's que devem estar em sintonia.

Em primeiro lugar, foi necessário elaborar os métodos para a Aplicação de assinatura (Aplicação Web) efetuar a comunicação com o servidor "PAdES Server Signer". No documento do CSC, identificado na secção 2.4, este não contém nenhum caso específico para assinaturas realizadas por smartcards. Por esse motivo, foi preciso adaptar a assinatura que está no cartão de cidadão com os métodos definidos pelo protocolo estabelecido pelo CSC. Como se pode verificar no diagrama de sequência da figura 3.5, os métodos na comunicação Aplicação Web - *PAdES Server Signer* são os mesmos que os utilizados no CMD para a assinatura segundo o protocolo definido pelo CSC.

Neste sentido, destaca-se que os métodos são os mesmos que foram utilizados para a CMD, mas existe uma diferença ao nível das variáveis. Uma nota importante a este respeito é a não existência de OTP no caso do CC. Em relação à assinatura de um ficheiro PDF no formato PAdES, ocorreu uma alteração significativa no método /dssFiler2Sign. Neste caso, no valor de retorno, a hash do documento PDF também é retornada para este poder enviar a assinatura para o cartão de cidadão.

O servidor PAdES ao invés de comunicar com o servidor da CMD para realizar a assinatura, terá de realizar a comunicação desses dados ao CC e vice-versa pelo facto deste estar conetado ao dispositivo do assinante.

Para comunicar com o cartão de cidadão existe um Plugin que é disponibilizado pela AMA, permitindo que qualquer aplicação, *Web* ou *Desktop*, que esteja a correr no computador do utilizador possa comunicar com o cartão de cidadão. Para isso, implementou-se uma aplicação, especificada na secção 3.3.1, que deverá estar a correr no computador do utilizador, comunicando com o CC. Só assim será possível realizar a assinatura de uma Hash a partir deste, caso este esteja conectado ao computador. Além disso, também permite recolher os certificados digitais que estão guardados no CC

Para além da comunicação entre a Aplicação de Assinatura e o PAdES Server Signer, no diagrama de sequência 3.5, também se pode reparar na existência da comunicação Aplicação de Assinatura-Plugin e Plugin-PAdES Server Signer. Dado que o Plugin tem dois métodos, isto pode acontecer em dois momentos.

Relativamente a esses métodos, convém explicar que um deles realiza a assinatura de uma hash e o outro responsabiliza-se pelo pedido da cadeia de certificação do CC. A Aplicação de assinatura, ao fazer os pedidos ao Plugin, tem de enviar as credenciais que recebeu anteriormente do "PAdES Server Signer". O valor da assinatura e a cadeia de certificação são enviadas diretamente para o "PAdES Server Signer" do Plugin, como se pode verificar no pedido "/sign" feito ao Plugin no diagrama de sequência 3.5. O valor da assinatura, como da cadeia de certificação são retornados diretamente do Plugin. O servidor PAdES possui um método para receber esta informação do PLugin: o /cc/sendPluginValues. A este respeito, salienta-se que este método tem de conter a informação de autenticação do utilizador.

Para exemplificar o processo de elaboração de uma assinatura anteriormente descrito, expõem-se, de seguida, o diagrama de sequência 3.5. Importa realçar que o processo do pedido das informações acerca da assinatura, incluindo o certificado de assinatura e a cadeia de certificação, se encontra demonstrado no diagrama de sequência B.1.

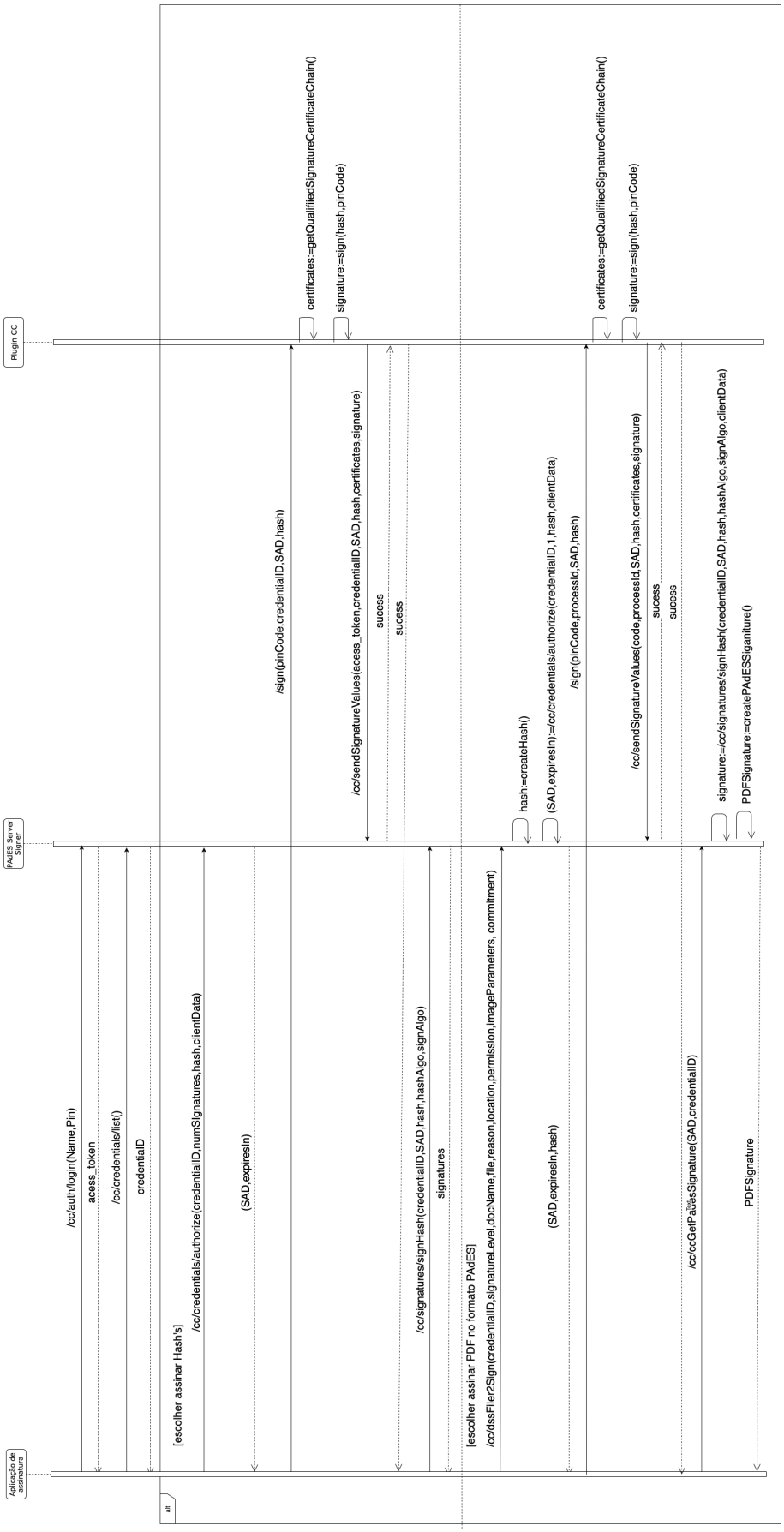


Figura 3.5: Diagrama de sequência do processo de assinatura do CC no PAdES Server Signer

3.3.1 Comunicação com o Plugin

Na comunicação com o CC, é necessário ter um software instalado no computador do utilizador para efetuar a comunicação com o smartcard. Para esse fim, foi desenvolvido um Plugin (que teve como base a API poreid, descrita na secção 4.5) que tem a função de intermediário na comunicação com o CC. Este Plugin irá conter dois métodos, um deles - "sign- será o método que irá receber os dados para realizar a assinatura e retorná-la. O outro irá retornar a cadeia de certificação ordenada da chave de assinatura do CC - "getCertificates".

O Plugin apenas recebe pedidos de aplicações que estão a correr no computador em que está instalado, por isso, só consegue receber pedidos do localhost. Desta forma, apenas as aplicações que estejam a correr no computador do assinante é que conseguem realizar pedidos às duas funções que o Plugin permite.

Como é possível observar no diagrama de sequência da figura 3.5, os pedidos ao Plugin têm como parâmetros as credenciais recebidas do PAdES Server Signer. Estes dados são utilizados pelo Plugin para retornar os valores diretamente para o PAdES Server Signer. Com este propósito, foram realizados dois métodos no servidor PAdES Server Signer, que recebe os valores do Plugin e os guarda, o que possibilita que, mais tarde, estes sejam usados. Os métodos no servidor PAdES Server Signer para onde o Plugin envia os dados são o cc/sendSignatureValues e o cc/sendCertificateValues. O primeiro é referente ao envio da assinatura e o segundo relaciona-se com os certificados.

O único valor de retorno que o Plugin envia para a API de assinatura é uma mensagem de sucesso ou insucesso, consoante o resultado da operação.

De seguida, são expostos os dois métodos que o Plugin permite (getCertificates e sign). Por fim, são apresentados os métodos do PAdES Server Signer para onde o Plugin envia os dados da assinatura e da cadeia de certificação.

1. **plugin/getCertificates**

   Método do *Plugin* que envia a cadeia de certificação ordenada da chave de assinatura qualificada do CC diretamente para o PAdES Server Signer. Não requer nenhum parâmetro de entrada para isso, apenas as credenciais de identificação para o PAdES Server Signer.

Atributo	Obrigatoriedade	Valor	Descrição
credentialID	Sim	String	A credentialID pedida pelo utilizador no método cc/credentials/list.

Figura 3.6: Pedido plugin/certificates

2. **cc/sendCertificateValues**

Método do PAdES Server Signer onde o *Plugin* envia a cadeia de certificação de assinatura qualificada do cartão de cidadão. Este método é chamado na função /plugin/certificates do *Plugin*.

Atributo	Obrigatoriedade	Valor	Descrição
credentialID	Sim	String	O credentialID pedido pelo utilizador no método cc/credentials/list.
certificates	Sim	Array of String	A cadeia de certificação ordenada da chave de assinatura qualificada do Cartão de Cidadão.

Figura 3.7: Pedido cc/sendCertificateValues

3. **plugin/sign**

Método do Plugin que realiza a comunicação com o Cartão de Cidadão para a realização da assinatura e a retorna para o PAdES Server Signer através do método /cc/sendSignatureValues

Atributo	Obrigatoriedade	Valor	Descrição
pinCode	Sim	String	Pin de assinatura digital do cartão de cidadão.
credentialID	Sim	String	A credentialID pedida pelo utilizador no método cmd/credentials/list.
SAD	Sim	String	A Signiture Activation Data (SAD) que vai ser usado como input no método signitures/signHash.
hash	Sim	Array of String	Os valores das hashs que vão ser assinadas em Base64.

Tabela 3.14: Pdido plugin/sign

4. **cc/sendSignatureValues**

Método do PAdES Server Signer para onde o Plugin envia as assinaturas das Hash's, realizadas no Cartão de Cidadão.

Atributo	Obrigatoriedade	Valor	Descrição
credetialID	Sim	String	O credentialID pedido pelo utilizador no método cmd/credentials/list.
SAD	Sim	String	A Signiture Activation Data (SAD) que vai ser usado como input no método signitures/signHash.
hash	Sim	Array of String	Os valores das hashs que vão ser assinadas em Base64.
signatures	Sim	Array of String	As assinaturas realizadas pelo Cartão de Cidadão.

Figura 3.8: Pedido cc/sendSignatureValues

Todos os métodos que foram demonstrados anteriormente nesta secção têm sempre o mesmo tipo de retorno. Desta forma, existem duas possibilidades: o sucesso ou o insucesso. Mesmo num caso de insucesso, é também enviada uma mensagem de erro.

Atributo	Obrigatoriedade	Valor	Descrição
status	Sim	String "sucess|error"	Valor do resultado do acontecimento. Se for bem-sucedido o valor será *sucess,* caso contrário será *error*.
message	Não	String	Mensagem de erro quando, não existe sucesso no método.

Figura 3.9: Pedido cc/sendSignatureValues

3.3.2 Adaptação ao CSC

Tal como na CMD, foi necessário escolher os métodos do padrão do CSC compatíveis com o cartão de cidadão. Deste modo, foram selecionados os mesmos métodos que foram definidos para a CMD. Por essa razão, as tabelas que estão descritas na secção 3.2.1 são iguais, já que se procedeu a pequenas alterações. De recordar que o CC está na mão do assinante. Como os atributos das tabelas que estão na parte do CMD são muito semelhantes aos do CC, não vão ser elaboradas novas tabelas para o CC, pelo que só serão apresentadas as diferenças, caso estas existam.

1. **auth/login**

   Este método é utilizado para receber o token de autenticação. O utilizador insere o número de telemóvel e o pin para realizar a autenticação na aplicação e receber o acess_token para utilizar nos métodos seguintes. A tabela 3.10 de pedido está apresentada a baixo, enquanto a de resposta é igual á tabela 3.2.

Atributo	Obrigatoriedade	Valor	Descrição
phoneNumber	Sim	String	O número do telemóvel que o utilizador pretende utilizar.
pin	Sim	String	O pin de 4 a 8 dígitos.

Figura 3.10: Pedido auth/login

2. **credentials/list**

   Mesmo valor do retratado na secção 3.2.1 refentente ao método credentials/list apresentado para a CMD.

3. **credentials/info**

   Os valores são exatamente os mesmos que estão na secção 3.2.1 credentials/info da CMD. A diferença é que este método só irá responder com sucesso se a Aplicação de assinatura tiver chamado o método do Plugin /plugin/certificates para o plugin pedir os certificados ao CC e enviar para o PAdES Server Signer. Caso contrário, o servidor não irá conter os certificados e, por isso, não os poderá retornar. O diagrama de sequência B.1 apresenta o processo de quando o utilizador quer utilizar este método para receber as informações acerca da assinatura.

4. **credentials/authorize**

   Este método apresenta os mesmos valores que na secção 3.2.1 credentials/authorize da CMD. Os campos de pedidos e resposta são os mesmos. Neste caso, não irá existir OTP enviada para o utilizador, porque como o CC está na mão do assinante, tendo já esse grau de segurança, não é necessário utilizar OTP.

5. **signatures/signhash**

   Este método tem como objetivo retornar a assinatura da Hash para o utilizador. A assinatura é realizada no smartcard, como se verifica no diagrama de sequência 3.5. Antes deste método ser chamado, a aplicação de assinatura que está no computador do assinante tem de comunicar com o Plugin para este realizar a assinatura da hash.

O Plugin serve como intermediário para comunicar com o smartcard. Para isso, é chamado o método /plugin/sign que recebe a hash para assinar, bem como os métodos de autenticação do utilizador para o método. Depois de a assinatura ser realizada com sucesso, o Plugin irá enviá-la para o PAdES Server Signer através da comunicação com o método /cc/sendSignatureValues.

Logo que este processo esteja realizado com sucesso, pode ser chamado o método /cc/signatures/signhash, no qual o servidor PAdES Server Signer já contém a assinatura e pode retorná-la para a Aplicação de Assinatura.

Os campos que estão definidos para este método são exatamente os mesmos que foram definidos para o CMD na secção 3.2.1. Os parâmetros para a comunicação com o Plugin estão definidos na secção 3.3.1.

3.3.3 Assinatura do PDF no formato PAdES

Para a realização da assinatura no CC de um documento PDF em formato PAdES os métodos da API diponibilizados são o /ccFiler2Sign e o /ccGetPadesSignature.

1. **ccFiler2Sign**

   Este método é imprescindível porque o CC está na posse do utilizador. Desta forma, é necessário realizar a Hash do documento. Para isso, realizou-se este método que recebe todos os parâmetros necessários para a elaboração da assinatura no formato PAdES do PDF, assim como o documento e o credentialID retornado na /cc/credentials/list. Deste modo, é retornada a hash do documento, assim como o SAD e o tempo de expiração. Além da realização da hash, este método chama o método /cc/credentials/authorize.

   Depois deste método ser executado, a API de assinatura deve executar o método do plugin/sign, descrito na secção 3.3.1, com a hash que foi recebida e os dados que esta exige.

Atributo	Obrigatoriedade	Valor	Descrição
SAD	Sim	String	A Signiture Activation Data (SAD) que vai ser usado como input no método signitures/signHash.
expiresIn	Sim	String	O tempo até expirar o SAD em segundos. Neste caso é de 300.
hash	Sim		O valor da hash do PDF que vai ser assinado.

Figura 3.11: Resposta ccFiler2Sign

2. **ccGetPadesSignature**

   Depois do Plugin ter enviado a assinatura da hash para o PAdES Server Signer, este método pode ser chamado, retornando a assinatura do PDF no formato desejado pelo utilizador. O método chama a função cc/signature/signHash para receber a assinatura, depois disso, realiza a assinatura do documento no fomato PAdES.

3.4 Base de Dados

Foi necessário ter um local para armazenar os dados que são utilizados em diferentes pedidos ao *PAdES Server Signer*. Para isso, foi decidido utilizar uma base de dados relacional para armazenar os dados. Este tipo de base de dados é uma maneira intuitiva e direta de representar dados em tabelas. As linhas destas tabelas são um registo de dados. As colunas correspondem a cada atributo que a tabela permite guardar. Cada linha tem ou não valor para cada atributo, podendo este ser ou não obrigatório consoante o que é definido. Uma das colunas deverá ser o identificador que tem que ser diferente em cada linha sendo esta a chave primária. Neste tipo de base de dados, para garantir a integridade dos dados não existem linhas duplicadas. A linguagem utilizada para criar e consultar os dados nas base de dados é o SQL.

As bases de dados relacionais são as melhores bases de dados para manter a consistência dos dados, sendo difícil para base de dados não relacional (NoSQL) manter o mesmo nível de consistência para grandes quantidades de dados. A este respeito, sublinha-se que este modelo é muito simples e não apresenta dificuldades na sua compreensão. De facto, apresenta uma grande precisão de dados e os mesmos não são repetidos desnecessariamente.[1]

Nestes casos, é possível dar permissões sobre as tabelas a que um utilizador tem acesso, podendo especificar esse mesmo acesso a determinadas tabelas para os utilizadores. A segurança, neste tipo de base de dados, é um dos principais destaques, já que oferecem um maior nível de segurança e um maior nível de confidencialidade e integridade, relativamente às base de dados não relacionais. [2]

Para os dois serviços de assinatura é necessário guardar os dados de autenticação do utilizador, os dados para a realização da assinatura e os dados do documento que irá ser assinado, no caso de ser no formato PAdES. Desta forma, foi definida uma tabela para guardar os dados da assinatura, em ambos os serviços, pois os atributos de dados são os mesmos. A chave primária desta tabela é o credentialID, o qual é retornado no método credentials/list. Na tabela 3.15 estão apresentados os campos que a base de dados contém, tanto para CC como para a CMD.

Como os dados são enviados para o servidor em diferentes pedidos, estes têm de ser guardados, principalmente os dados do documento PDF no pedido /dssFiler2Sign. De facto, é crucial que isso aconteça, já que estes dados são extremamente importantes para chamar o método /dssFiler2SendOTP, onde é realizada a assinatura depois do envio do OTP, por parte do cliente. Caso o utilizador queira usar uma imagem para inserir na assinatura, tanto a imagem como o local onde estará localizada serão guardados.

[1] https://www.ibm.com/cloud/blog/sql-vs-nosql

[2] https://www.softwaretestinghelp.com/sql-vs-nosql/#SQL_vs_NoSQL_Security

Base de Dados		
Parâmetro	**Tipo**	**Descrição**
credential_id	VARCHAR (45) - Chave Primária	Identificador da aplicação que efetua o request
phone_number	VARCHAR (45)	Número de telefone do utilizador.
pin	VARCHAR(64)	Pin cifrado recebido no auth/login
sad	VARCHAR (45)	SAD
document	blob	documento PDF para realizar a assinatura no formato PAdES
signing_date	VARCHAR (45)	Data da assinatura
signing_level	VARCHAR (15)	Nível de assinatura PAdES(B,T,LT,LTA)
num_signatures	int	Número de assinaturas
signature_reason	VARCHAR (45)	Razão para a assinatura
signature_location	VARCHAR (45)	Localização da assinatura
permission	VARCHAR (45)	Tipo de permissão para alterações do PDF
image_file	blob	Imagem para a realização de assinatura visivel
image_originx	int	Local da assinatura - eixo X
image_originy	int	Local da assinatura visisvel - eixo Y
image_width	int	Largura da assinatura visível
image_heigth	int	Altura da assinatura visível

Tabela 3.15: Tabela Base de dados

3.5 Aplicação *Web*

Dado que o PAdES Server Signer é uma plataforma/interface CSC para realizar assinaturas, surgiu a necessidade de elaborar uma Aplicação Web com o intuito de corresponder a uma Aplicação de Assinatura. Esta aplicação fica encarregue de concretizar a interação com o utilizador, permitindo que este possa assinar um documento PDF, através do PAdES Server Signer. Assim, o utilizador consegue realizar a assinatura de um documento no formato PAdES.

A aplicação terá uma interface para o utilizador poder escolher as diferentes opções para a assinatura. Esta aplicação irá permitir a realização da assinatura do PDF, com CMD como com o CC. Para realizar a assinatura com a CMD, apenas terá de comunicar com o servidor PAdES. Por outro lado, em relação ao CC, a comunicação terá de ser feita tanto com o Plugin, que deve estar instalado no computador do utilizador, como o servidor PAdES Servder Signer. Para a utilização da aplicação num Browser no telemóvel, só será possível fazer a assinatura através da CMD, pois não existe possibilidade de comunicar com um smartcard.

Na interface com utilizador, esta aplicação permite-lhe escolher qual o serviço de assinatura em que pretende realizar a assinatura: CMD ou CC. A partir desse momento, a Aplicação Web terá um comportamento diferente, de acordo com o serviço escolhido.

Em relação à assinatura para a CMD, esta aplicação tem como objetivo realizar a mesma tarefa da aplicação de Assinatura descrita no diagrama de sequência 3.4 do CMD. A aplicação apenas tem

de recolher os dados do utilizador e realizar o procedimento para assinar o documento que o utilizador pretende, comunicando com o servidor PAdES. Assim, esta atua como um intermediário deste servidor.

Para a assinatura com o CC, como está descrito no diagrama 3.5, enquanto Aplicação de Assinatura, esta comunica com 2 elementos. É importante mencionar que o Plugin deve estar instalados no dispositivo onde esta aplicação está a correr. Se o Plugin não tiver sido instalado no dispositivo ou caso o CC não esteja conectado ao dispositivo, este processo não terá condições para se efetuar.

Como esta aplicação é apenas feita para assinar documentos PDF, para os dois serviços de assinatura permitidos na API, no momento de selecionar o modo como a assinatura irá ser realizada, a única opção consiste em assinar o PDF no formato PAdES. Os diagramas de sequência 3.4 e 3.5, revelam o que é efetuado.

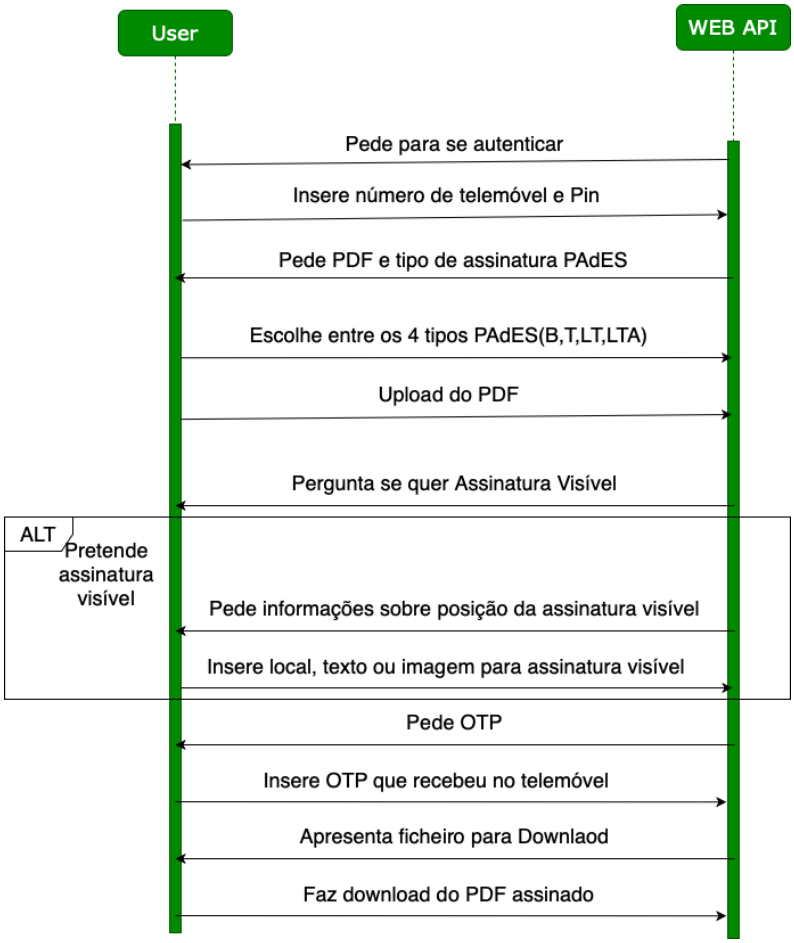


Figura 3.12: Processo de assinatura

Inicialmente, na página inicial da aplicação Web, o utilizador poderá optar por fazer a assinatura através do CMD ou através do CC. Se o utilizador optar pelo segundo, então a aplicação terá de verificar se o Plugin está a instalado e ligado no seu computador. Se isso se verificar, prossegue-se para a página de assinatura, representada no diagrama de sequência 3.12 e na figura 3.13, que é comum aos dois serviços de assinatura. Em primeiro lugar, são pedidos os dados para a autenticação, ou seja, o contacto telefónico e o PIN. De seguida, seleciona-se o tipo de assinatura PAdES (B,T,LT,LTA) que se pretende realizar.

Após isso, é feito o *upload* o ficheiro PDF para assinar, assim como a escolha de algumas definições de assinatura. Por fim, resta ao utilizador decidir se pretende que a assinatura fique visível no ficheiro. Caso seja essa a sua vontade, pede-se ao utilizador que insira a imagem e selecione o local onde se localizará a assinatura. Logo que concluída esta parte, abre-se uma página com um local para o utilizador inserir o OTP. Se esta etapa for bem sucedida, o ficheiro assinado é transferido para o seu computador.

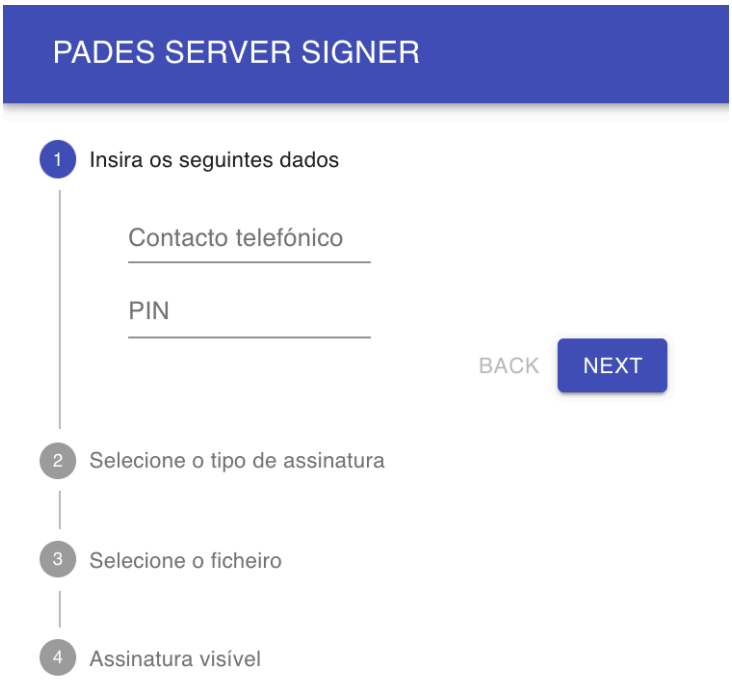


Figura 3.13: Interface da Aplicação Web

Capítulo 4

Tecnologias

4.1 *RESTful Web Service*

Os Restful Web Services é um serviço leve, sustentável e escalável, que é construído de acordo com a arquitetura REST(REpresentational State Transfer). Esta trata-se de um estilo de arquitectura própria para sistemas, tal como descrito por Roy Thomas Fielding no documento "Architectural Styles and the Design of Network-based Software Architectures"[13].

O REST é uma maneira de aceder a recursos que se encontram num ambiente particular. Um exemplo de recursos são documentos, fotos, dados de utilizadores e funcionalidades. Se um cliente necessitar de algum recurso terá de fazer um pedido para aceder aos recursos. O REST define o modo como esses recursos devem ser acedidos. Os recursos são acedidos por niform Resource Identifiers (URI)[1], normalmente um *URL HTTP*. Esta arquitetura permite a utilização de 4 métodos HTTP:

1. *GET* - para pedir um recurso do servidor;
2. *POST* - Para criar um recurso no servidor;
3. *DELETE* - para apagar um recurso no servidor;
4. *PUT*- para modificar um recurso no servidor.

O REST foi desenhado para utilizar um protocolo de comunicação *stateless*, isto é, o estado não é mantido pela aplicação. Esta arquitetura funciona num sistema de pedido-resposta, em que o cliente faz um pedido e o servidor rejeita o pedido, ou responde ao cliente.

Um RESTful Web Service permite que várias aplicações escritas em diferentes linguagens comuniquem entre si, como também aplicações que correm em diferentes sistemas operativos. Além disso, também facilita a comunicação através de diferentes dispositivos, como por exemplo *smartphones*.

[1] https://www.ietf.org/rfc/rfc2396.txt

A transferência de dados numa API RESTful pode ser realizada através de JSON (Javascript Object Notation)[2], HTML, XLT, Python, PHP ou texto limpo. O mais popular, neste tipo de serviços, é o JSON porque é de fácil entendimento por parte dos programadores, sendo uma linguagem leve e de rápido processamento por parte das máquinas.[3]

O *PAdES Server Signer* disponibiliza uma API REST, pois mediante a arquitetura apresentada anteriormente e os métodos a que o *PAdES Server Signer* tem de responder, o REST fornece facilidade de compreensão e de implementação porque apenas utiliza 4 métodos do HTTP, que são facilmente correspondidos aos métodos do *PAdES Server Signer*. O REST ao permitir a utilização de JSON faz com que a transferência de dados seja mais rápida comparada com o SOAP que utiliza XML, pois o JSON é menos complexo e mais leve que o XML, sendo relativamente fácil realizar a trancrição de JSON para um objeto da linguagem de programação. As variáveis que os pedidos do *PAdES Server Signer* tem definidos são facilmente reproduzidos em JSON, pois é utilizado um formato chave-valor, como demonstrado nas tabelas da arquitetura do *PAdES Server Signer*, sendo fácil a transcrição dos métodos para este formato de transferência de dados.[4]

4.2 Spring Framework

Dado o grande leque de funcionalidades codificadas capazes de ajudar a desenvolver esta aplicação, optou-se pela bibloteca Digital Signature Service (DSS) em Java (ver secção 4.3). Neste sentido, surgiu a necessidade de selecionar uma linguagem e framework desenvolver com este projeto.

Posto isto, selecionou-se a Framework Spring[5]. Trata-se de um framework open source que fornece a estrutura e os padrões para o desenvolvimento de uma aplicação Java, abstraindo do programador da gestão dos objetos, ciclos de vida e dependências para os componentes e serviços no código através de um *Application Context* e de *Dependency Injection* (DI).[6]

De facto, esta é uma das *frameworks* para *Java enterprise edition* mais usadas, uma vez que permite criar diferentes tipos de aplicações e apresenta um grande número de modelos em função do tipo de aplicação que se pretende desenvolver.

A propósito da realização de uma API Rest, utilizou-se o módulo Spring Web MVC framework[7]. Este módulo revelou-se, assim, uma grande ajuda para a execução desta aplicação.

Entre os vários motivos considerados para esta seleção, salienta-se o facto do controlador conseguir lidar com todos os tipos de método HTTP necessários.

No caso de uma API Rest, a representação de dados é extremamente importante. Esta framework permite transferir, diretamente, os dados dos pedidos à aplicação para a classe em que se pretende

[2] https://www.oracle.com/database/what-is-json/
[3] https://www.guru99.com/restful-web-services.html
[4] https://www.freecodecamp.org/news/benefits-of-rest/
[5] https://spring.io
[6] https://spring.io/why-spring
[7] https://docs.spring.io/spring-framework/docs/3.2.x/spring-framework-reference/html/mvc.html

utilizá-los, assim como enviar a resposta a um cliente. Realça-se que, os dados de resposta são automaticamente adicionados ao *body* da resposta HTTP, no formato pretendido. Com efeito, esta simplificação de processos facilita a compreensão dos dados tratados pela aplicação

4.3 Digital Signature Service

Nesta secção, dar-se-á destaque ao modo como a realização da assinatura em formato PadES do PDF foi concretizada para os vários tipos de assinatura PAdES-B, PAdES-T, PAdES-LT, PAdES-LTA.

Para este fim, utilizou-se a biblioteca *Digital Signature Service*, em Java, disponibilizada pela União Europeia para a realização de assinaturas digitais, segundo os seus padrões[10]. Esta biblioteca possui várias funcionalidades para assinaturas digitais para PDF no formato PAdES, como para outros tipos de assinatura como o CAdES(CMS Advanced Electronic Signatures) e o XAdES(XML Signature Extension).

4.3.1 Criação da assinatura PADES

No que ao PAdES diz respeito, esta biblioteca fornece um grande leque de funcionalidades para realizar as assinaturas do PDF nesse formato. Em primeiro lugar, começou por se estudar a biblioteca, com vista a perceber como se podia integrá-la na realização do projeto. Depois de analisada em detalhe, deu-se inicio à codificação de uma assinatura em formato PAdES-B, devido ao facto de ser mais simples e mais fácil de realizar.

Nesse sentido, a biblioteca fornece uma classe com o nome de PAdESService, sendo esta a classe que tem todas as operações necessárias para a realização da assinatura. Esta classe recebe um CertificateVerifier, que tem a sua implementação como CommonCertificateVerifier determina como o DSS se conecta aos recursos externos, não sendo necessária nenhuma modificação até á realização da assiantura no formato PAdES-LTA. Esta classe contém várias funções. No entanto, neste projeto, foram apenas usadas a getDataToSign e a função signDocument. Relativamente à getDataToSign, esta trata-se do método que fornece uma lista de bytes para realizar a assinatura a partir do documento PDF e dos parâmetros PAdES que se pretende para realizar a assinatura. Quanto à função signDocument, esta assina o PDF a partir do documento, dos parâmetros PAdES e da hash assinada do documento.

4.3.2 Parâmetros para a realização da assinatura

Para a realização da assinatura, foi necessário criar a classe PAdESSignatureParameters, que contêm os parâmetros PAdES em que a assinatura se vai basear.

Os parâmetros que são permitidos na API para a realização da assinatura tratam-se de informações pessoais como o contacto, a razão e a localização em que se realizou a assinatura.

Depois disso, podem-se definir as permissões para alterações do PDF. Existem três permissões permitidas: CHANGES_PERMITTED, MINIMAL_CHANGES_PERMITTED e NO_CHANGE_PERMITTED, estas

três permissões estão definidas na secção 3.2.2 na descrição do pedido *dssFiler2Sign*, e a opção selecionada nesse pedido é a utilizada pelo PAdES Server Signer.

O algoritmo selecionado pela nossa API para efetuar a *digest* do documento PDF para a realização da assinatura é o SHA256.

O tamanho para guardar a assinatura terá que ser definido, uma vez que este precisa de ser calculado tendo em conta os certificados que serão guardados com a assinatura, que será posteriormente verificada. Este tamanho aumentará em função do nível de assinatura. No PADES-B, apenas se guarda a assinatura e o seu certificado, juntamente com a cadeia de certificação. Já no PADES-LTA, este terá que registar a assinatura times-stamp, como terá que conter todos os certificados e cadeias de certificação, tanto da assinatura do documento como da assinatura time-stamp.

O SignatureFieldID é uma string com o id do local onde será colocada a assinatura visualmente. A DSS framework permite criar assinaturas visíveis no PDF e produzir esses campos, possibilitando a inserção de texto ou imagens, ou dos dois elementos em simultâneo.

```
1  public static PAdESSignatureParameters createPAdESSignatureParametersBlevel(String
       ↪ signatureField,String signatureLevel,String contactInfo,String reason,String
       ↪ location,String permission){
2  PAdESSignatureParameters pAdESSignatureParameters = new PAdESSignatureParameters();
3  pAdESSignatureParameters.setContactInfo(contactInfo);
4  pAdESSignatureParameters.setReason(reason);
5  pAdESSignatureParameters.setLocation(location);
6  pAdESSignatureParameters.setPermission(permission);
7  pAdESSignatureParameters.setSignatureFieldId(signatureField);
8  pAdESSignatureParameters.setDigestAlgorithm(DigestAlgorithm.SHA256);
9  pAdESSignatureParameters.setContentSize(25000);
10 pAdESSignatureParameters.setSignaturePackaging(SignaturePackaging.ENVELOPED);
11 switch (signatureLevel){
12    case ("pades_t"):
13       pAdESSignatureParameters.setSignatureLevel(SignatureLevel.PAdES_BASELINE_T);
14       break;
15    case ("pades_lt"):
16       pAdESSignatureParameters.setSignatureLevel(SignatureLevel.PAdES_BASELINE_LT);
17       break;
18    case ("pades_lta"):
19       pAdESSignatureParameters.setSignatureLevel(SignatureLevel.PAdES_BASELINE_LTA);
20       break;
21    default:
22       pAdESSignatureParameters.setSignatureLevel(SignatureLevel.PAdES_BASELINE_B);
23       break;
24 }
25
26 pAdESSignatureParameters.bLevel().setSignaturePolicy(getSignaturePolicy());
27
28 return pAdESSignatureParameters;
```

```
29 }
```

4.3.3 Assinatura visível no PDF

A aplicação permite realizar assinaturas visíveis, como especificado no ETSI EN 319 142-1 [14]. No objeto SignitureParameters, existe um atributo especial denominado por SignatureImageParameters, que permite inserir uma assinatura visível no PDF, como possibilita a inserção de texto, imagem ou os dois elementos em simultâneo. A assinatura poderá estar em qualquer página, ocupando qualquer espaço dentro da mesma. A framework permite definir o tamanho da assinatura, isto é, permite definir o local, o ponto inicial onde começa a assinatura, definindo o ponto x e y do começo. De seguida, permite precisar a largura e a altura, sendo definida da esquerda para a direita e de cima para baixo. O utilizador pode inserir o texto e a imagem que pretende nesse espaço.

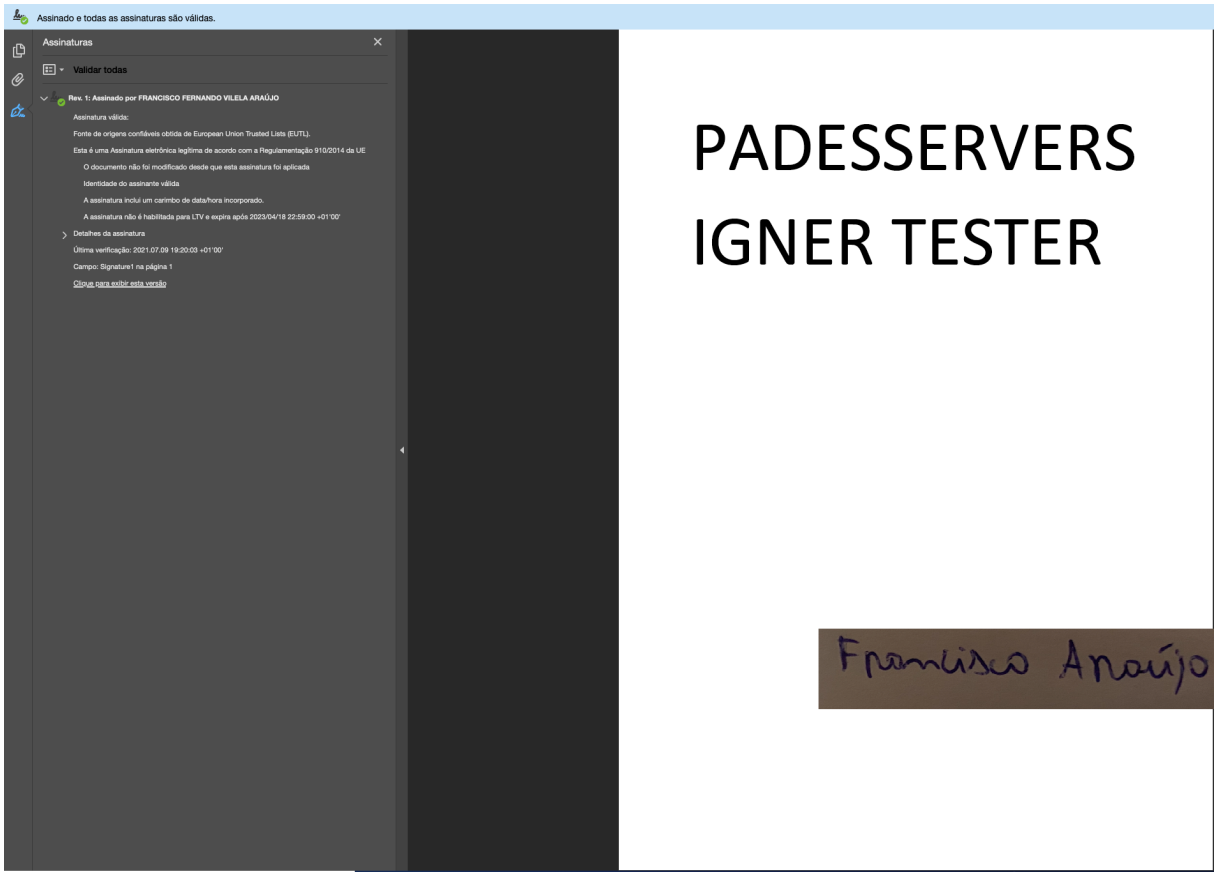


Figura 4.1: Assinatura de PDF no formato PAdES-T com assinatura visível através do PAdES Server Signer

4.3.4 Assinatura PAdES-B

Depois de definidos os parâmetros para a assinatura, procede-se à execução da mesma. A assinatura do documento PDF é necessariamente realizada em dois passos: o primeiro consiste na realização do pedido da hashs do PDF para enviar para o serviço de assinatura; o segundo consiste na adição da assinatura

do PDF com base na assinatura recebida pelo serviço de assinatura. Por esse motivo, foram criadas dois métodos.

No primeiro método, após o preenchimento dos parâmetros, considerando as escolhas do cliente apresentadas no pedido, procede-se à solicitação da cadeia de certificação dos certificados da assinatura para serem inseridos no PDF em conjunto com a assinatura. Esta cadeia é também inserida na classe PAdESSignatureParameters, a qual contém os parâmetros de assinatura. Concluída esta etapa, resta executar o método getDataToSign que retorna o digest para ser assinado pelo serviço de assinatura.

A segunda parte do processo, consiste na inserção da assinatura retornada pelo serviço de assinatura(CMD e CC), que é introduzida no PDF. A função do DSS utilizada, neste caso, é signDocument, que recebe como parâmetros o documento a assinar, os parâmetros da assinatura e a assinatura.

4.3.5 Outras Assinaturas PAdES

Para realizar a assinatura em PAdES-T, PAdES-LT e PAdES-LTA, o processo é similar ao anterior, sendo necessário especificar-se o tipo de assinatura. Acrescenta-se, igualmente, o tamanho onde se vai guardar as assinaturas e o URL da Time Stamping Authority (TSA), para a framework DSS adicionar o timestamp à assinatura.

4.4 MySQL

Para realizar a assinatura no formato pedido era necessário guardar o documento, bem como os dados da assinatura e os dados de identificação do utilizador. Para isso, foi criada, em MySQL[8], um sistema para gerir os dados da aplicação que utiliza a linguagem SQL. Este é um dos mais populares do mundo, tendo sido desenvolvido pela Oracle. Além disso, é também muito estável e apresenta uma grande compatibilidade com a linguagem JAVA, através da API JDBC.

4.5 poreid

A biblioteca poreid é uma biblioteca open-source (disponível em `https://github.com/poreid/poreid`) para JAVA que possibilita a comunicação com o CC realizada por Rui Martinho e António Braz. Esta biblioteca apresenta várias funções para a realização de diferentes processos no CC. Entre as várias funcionalidades que permite, destacam-se duas funções muito importantes utilizadas para a realização do *Plugin*, como especificado na secção 3.3.1.

1. **sign**: Esta função tem como objetivo realizar a assinatura digital no CC a partir de alguns parâmetros fornecidos: a hash do que se pretende assinar, que se trata de um código pin de assinatura do CC e o algoritmo que foi utilizado para realizar a hash. É retornada a assinatura elaborada com a chave privada que está no CC, em bytes.

[8] `https://www.mysql.com`

2. **getQualifiedSignatureCertificateChain**: Tem como função retornar a cadeia de certificação ordenada da chave de assinatura qualificada do CC, começando no certificado de assinatura.

Esta biblioteca é muito simples de utilizar e, para este projeto, apenas foi necessário utilizar estes dois métodos. Contudo, além dos métodos usados, esta biblioteca disponibiliza muitos outros para a comunicação com o CC, que são úteis não só para a realização da assinatura, como também para autenticação a partir do CC, modificação e verificação dos diferentes pins que o CC contém.

4.6 React.js

O React.js[9] é uma das bibliotecas mais utilizadas de *front-end* para a linguagem *Javascript* quando se procura realizar aplicações Web, tendo sido a biblioteca utilizada para realizar a aplicação Web deste projeto.

Sendo uma biblioteca open-source de javascipt que é usada para realizar interfaces para o utilizador, esta é utilizada para realizar as views para aplicações Web.

A simplicidade do React.js torna-o muito fácil de usar. De facto, a utilização de componentes e o uso apenas de javascript facilita muito na criação de aplicações Web. O React usa uma sintaxe especial chamada JavaScript Syntax Extension (JSX) que permite a mistura de HyperText Markup Language (HTML) no javascript, o que facilita o trabalho do programador. Posto isto, constata-se que, efetivamente, para quem tem algum conhecimento de Cascading Style Sheets (CSS) e HTML, não será difícil aprender a usar esta biblioteca.

A par do que referi, esta biblioteca apresenta uma outra vantagem, já que pode ser usada para criar aplicações mobile, a partir do React Native[10]. As aplicações escritas em React são facilmente testadas. Além disso, já foi anteriormente utilizada por quem desenvolveu este projeto.

[9] https://reactjs.org
[10] https://reactnative.dev

Capítulo 5

Teste e segurança

5.1 OWASP Sheet Series

O Open Web Application Security Project (OWASP)[1] é uma fundação não lucrativa que tem como objetivo melhorar a segurança do *software*. O OWASP tem um grande leque de projetos *open-source* para ajudar os programadores a desenvolver aplicações Web seguras. Um dos projetos publicados pelo OWASP é o *OWASP Cheat Sheet Series*[2] que contem informação especifica sobre vários tópicos para a construção de aplicações Web seguras. Um dos tópicos utilizados nesta dissertação é o *REST Security Cheat Sheet*[3], sendo este um documento com as melhores práticas de segurança para o desenvolvimento de uma aplicação *REST*.

1. **Validação do Input**

   Na aplicação PAdES Server Signer procedeu-se à validação do input, tal como está explicado no OWASP/CHeatSheetSeries. Esta validação é realizada para assegurar que apenas os dados que estão de acordo com o está especificado entram no sistema, impedindo que os dados que não estejam no formato correto entrem na base de dados ou causem problemas na aplicação. A validação do input deve ser a primeira coisa que ocorre no sistema. Por essa razão, nesta aplicação, sempre que os dados vêm de uma entidade externa que não é de confiança, estes devem ser validados. A validação do input não deverá ser o método principal para prevenir vulnerabilidades e ataques ao sistema. No entanto, pode contribuir para os reduzir, se for implementada corretamente.

   Para a realização da validação no PAdES Server Signer concretizou-se a validação do input contra um JSON Schema - objeto escrito em JSON que define a estrutura e conteúdo de objetos JSON.

[1] https://owasp.org
[2] https://cheatsheetseries.owasp.org
[3] https://cheatsheetseries.owasp.org/cheatsheets/REST_Security_Cheat_Sheet.html

Além disso, permite criar regras para validar objetos JSON de acordo com o esquema definido, validando os valores de cada campo JSON e verificando se estão válidos consoante as regras descritas. A sua função passa por validar se a sintaxe é correta; validar os atributos que são permitidos e obrigatórios, validando semanticamente os dados; validar o tipo de dados recebidos (string,number,integer,boolean,null,array,object); validar o comprimento; validar a partir de uma expressão regular.

Para todos os métodos que a API permite é realizada a validação a partir de um JSON Schema. Um exemplo disso é esquema a seguir apresentado, relativamente ao método credentials/info para CC e CMD. Tal como se constata, é possível observar que o atributo credentialID e certificates são obrigatórios. O credentialID trata-se de uma string e tem uma expressão regular a defini-lo. Quanto ao atributo certificates, este contém apenas três valores definidos.

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "type": "object",
  "properties": {
    "credentialID": {
      "type": "string",
      "pattern": "[0-9a-f]{8}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{12}"
    },
    "certificates": {
       "enum" : ["chain","none","single"]
    }
  },
  "required":["credentialID","certificates"]
}
```

2. **Validação de *Content Types***

   Uma aplicação REST deve conter o formato do corpo dos seus pedidos e respostas, de acordo como o "content type"que é definido no cabeçalho HTTP. Se não existir essa conformidade pode acontecer uma compreensão errada por parte do utilizador e também pode dar origem a "code injection/execution".

   Todos os pedidos que tenham sido realizados sem o cabeçalho HTTP "content type"devem ser rejeitados com os valores "406 Unacceptable"ou "415 Unsupported Media Type".

   Nos controladores do Spring devem estar inseridos no pedido e na resposta o valor do "content type"que estes utilizam, devendo estar definidos da seguinte forma: @consumes("application/json") e @produces("application/json"), para pedidos e reposta no formato JSON.

3. **Autenticação e autorização**

O método de autenticação inicial do PAdES Server Signer é o Basic Authetication. A este respeito, esclarece-se que este é um mecanismo que possibilita o acesso a recursos por HTTP. As credenciais para a autenticação são enviadas no cabeçalho HTTP do pedido no campo Authorization, estando estas no formato Base64. Esta autenticação tem de ser usada com TLS/HTTPS para ser considerada segura. Deste modo, este deve ser o primeiro método a ser chamado na API, sendo o /auth/info. Este método retorna uma token para ser usado com autenticação para os restantes métodos.

Nos métodos seguintes, a API utiliza uma Token Based Authetication, diferente da autenticação, na qual se envia o username e a password em cada pedido. Neste caso, um token é enviado em cada pedido para realizar a autenticação. Em primeiro lugar, o servidor tem um método que utiliza uma autenticação com credenciais, tal como a Basic Authetication, por exemplo. Este retorna um token para, mais tarde, utilizar e inserir no campo Authorization em cada pedido que faz à API, que fica guardado do lado do cliente para usar quando pretender aceder à API. Sempre que este pedido acontece, o servidor verifica se o Token é válido. Se este for válido, é possível realizar-se aquele método. O token é assinado com uma chave privada que está no servidor, sendo o acesso ao token exclusivo desse mesmo servidor. Por isso, aquele token pode ser validado pelo servidor. Sobre o token, é igualmente importante mencionar que este contém uma data de expiração. No PAdES Server Signer, por exemplo, é de 15 minutos, desde o momento em que foi gerado. Assim que atinge os 15 minutos, é expirado.

Para a autenticação do PAdES Server Signer, selecionou-se o JWT(Json Web Token), que é baseado no standart (RFC 7519[4]). Sublinha-se que este pode ser usado para a autorização e para trocar informação segura através de um objeto. Por ser em JSON, este token é mais simples do que os baseados em XML. De facto, o JSON é muito comum nas linguagens de programação, fazendo com que seja fácil mapear o JSON diretamente para objetos da linguagem.

O JWT consiste numa *string* que pode ser passada no cabeçalho, no corpo de um pedido HTTP ou até no URL, sendo que todo o conteúdo do token pode ser visualizado do lado do cliente. O JWT divide-se em três partes, que surgem separadas por um ponto: o cabeçalho, o payload e a assinatura, conforme na figura 3. Estes três pontos devem ser codificados no formato Base64 e concatenados.

a) No cabeçalho encontra-se o algoritmo para a realização da hash da assinatura(”alg”) e o tipo do token(”typ”).

b) No payload estarão os dados que se pretendem introduzir no token, chamados de claims. Nesta API, foi inserido o tempo em que o token expira (”exp”), o tempo em que ele foi emitido (”iat”), bem como o credentialID(”sub”).

[4] https://datatracker.ietf.org/doc/html/rfc7519

c) Na assinatura está assinada a concatenação do cabeçalho e do payload. A assinatura pode ser realizada quer com um segredo (usando o algoritmo HMAC), quer com um par de chaves RSA ou ECDSA. Nesta aplicação é usado o HMAC com SHA-256, isto é, utiliza-se uma função de hashing com uma chave secreta. Neste caso, a chave secreta é usada para assinar e validar, uma vez que este processo é realizado na nossa API. Caso contrário, se fosse necessário verificar fora da aplicação, seria preciso utilizar uma assinatura RSA com SHA-256, por exemplo.

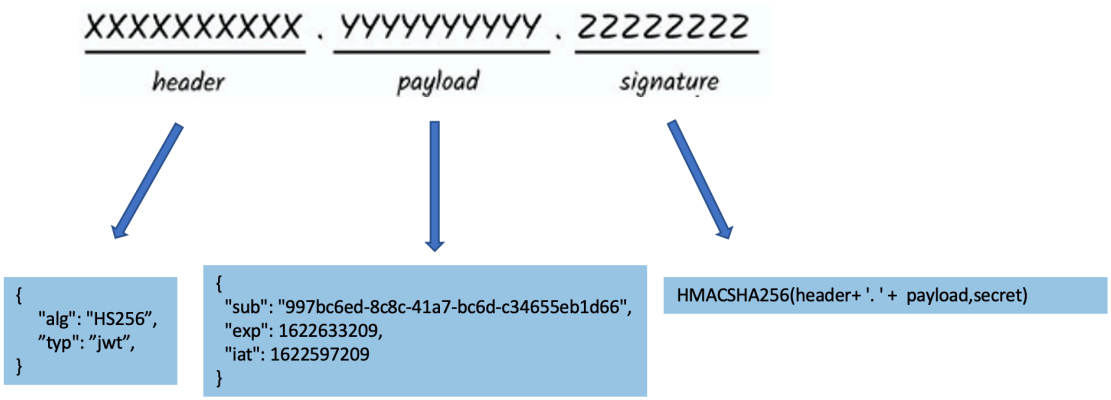


Figura 5.1: Estrutura de um JWT

5.2 Spring Security

O Spring Security[5] é uma framework que fornece autenticação, autorização e proteção contra ataques informáticos em algumas vulnerabilidades comuns para aplicações realizadas em Spring. Destaca-se que este é muito fácil de integrar com o Spring MVC.

Esta framework é baseada em Servlet Filters. Significa isto que o cliente faz um pedido à aplicação e, antes do pedido chegar ao controlador, este passa por um conjunto de filtros designado por FilterChain. Estes filtros são utilizados para duas situações. Em primeiro lugar, pode mencionar-se que este previne que os filtros que estão abaixo ou o Servlet sejam invocados. Este filtro escreve, normalmente, para o HttpServletResponse (resposta do pedido HTTP) e este é terminado. Em segundo lugar, refere-se que o filtro é utilizado para realizar alterações no pedido, quer seja nas resposta HTTP, como no sistema.

No caso do PAdES Server Signer, para a autenticação criou-se um filtro para verificar o JWT. Se essa operação for realizada com sucesso, então o processo continua, avançando com o pedido. Pelo contrário, se não for bem sucedido, será elaborada uma resposta com um erro.

A spring security permite a realização de uma configuração, onde são inseridas as funcionalidades desta biblioteca que serão, mais tarde, utilizadas pela aplicação. Nesta configuração, é necessário apontar

[5] https://spring.io/projects/spring-security

quais os caminhos na aplicação que o user deve estar autenticado ou não. Se um determinado caminho necessitar de autenticação, primeiro irá passar nos filtros de autenticação, neste caso, no filtro do JWT. Depois, proceder-se-á para a continuação do pedido. Em alguns casos, pode não ser preciso que o caminho passe no filtro do JWT. Por esse motivo, deve ser especificado, nessa configuração, quais os caminhos permitidos sem autenticação, como o auth/login, por exemplo. Este é pedido com as credenciais que irá ter como resposta o token. De realçar que as configurações de segurança para a proteção contra ataques devem, igualmente, ser ativadas nesta configuração.

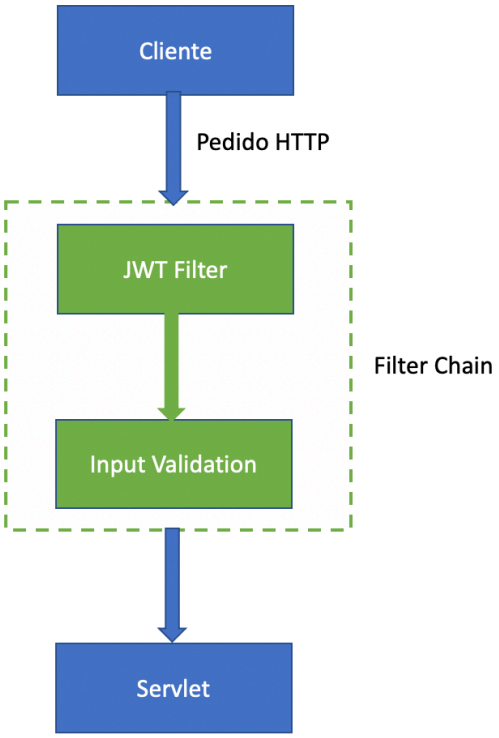


Figura 5.2: auth/loginResponse

5.3 Sast

Um Static Application Security Testing (SAST), também conhecido como *White Box Testing*, é uma metodologia de teste que analisa o código fonte para encontrar vulnerabilidades na segurança de aplicações, aumentando a segurança contra ataques informáticos. O SAST faz *scan* ao código fonte da aplicação sem necessitar do código compilado. É uma metodologia utilizada durante o desenvolvimento do software, pois pode identificar vulnerabilidades no código mesmo que este não esteja acabado. O código pode ser validado pelo SAST sem a necessidade da aplicação executar, ajudando a encontrar vulnerabilidades no código mesmo nas partes iniciais do desenvolvimento [22].

Uma grande qualidade do SAST, está no facto de este analisar o código fonte rapidamente e dar como resultado as vulnerabilidades e Bugs que estão no código, muito mais rápido que se for feita uma *review* por um programador, identificando vulnerabilidades críticas como as que estão no "OWASP Top Ten".

Um problema destas ferramentas são a grande quantidade de Falsos Positivos que estas lançam, sendo muitas vezes em muita maior percentagem do que os resultados positivos, mas compensando pelo facto de o programador só precisar de verificar os resultados que o SAST devolve.

O SonarQube[6] é um SAST que permite realizar scans em várias linguagens de programação sendo uma delas a linguagem Java, avaliando e apresentando resultados em três categorias:

1. ***Code Smell***: Um problema de manutenção que torna o código confuso e difícil de manter.
2. ***Bug***: Um erro no código que pode levar a um comportamento inesperado da aplicação quando esta está a correr.
3. ***Vulnerability***:Um local no código onde está susceptível a um ataque.

Estes resultados são retratadas em 5 categorias de risco, sendo apresentadas de seguida do mais critico para o menos:

1. ***Blocker***: Um Bug que pode ter uma grande impacto no comportamento da aplicação, neste caso o código dever ser imediatamente arranjado.
2. ***Critical***:Um Bug com pouca probabilidade de impacto na aplicação ou um erro de segurança. O código deve ser revisto de imediato.
3. ***MAJOR***: Uma falha de qualidade que pode ter um grande impacto na produtividade do programador.
4. ***Minor***: Uma falha de qualidade que pode ter um algum impacto na produtividade do programador.
5. ***Info***: Resultado que não é um Bug ou falha de qualidade.

No domínio das vulnerabilidades de segurança o SonarQube apresenta dois campos,

1. a ***Vulnerability***, em que, existe um problema de segurança na aplicação que tem que ser arranjado o mais rápido possível, e
2. o ***Security Hotspot***, em que é realçado um ponto sensível a nível de segurança que o programador deve rever para verificar se naquele local do código existe uma vulnerabilidade ou não.

O SonarQube verifica o código contra as vulnerabilidades existentes no *OWASP TOP TEN* [17], que é um documento na área da segurança de aplicações web, em que apresenta as 10 vulnerabilidades mais criticas para este tipo de aplicações. A verificação contra o *OWASP TOP TEN* é provavelmente o primeiro passo mais eficaz para produzir código com maior segurança [17].

[6] https://www.sonarqube.org

Como se pode verificar na figura 5.3, estas são as vulnerabilidades do *OWASP TOP TEN* de 2017 que são examinadas pelo SonarQube para a linguagem Java, sendo diferenciadas, umas como *Security Hotspot* e outras como *Vulnerability*.

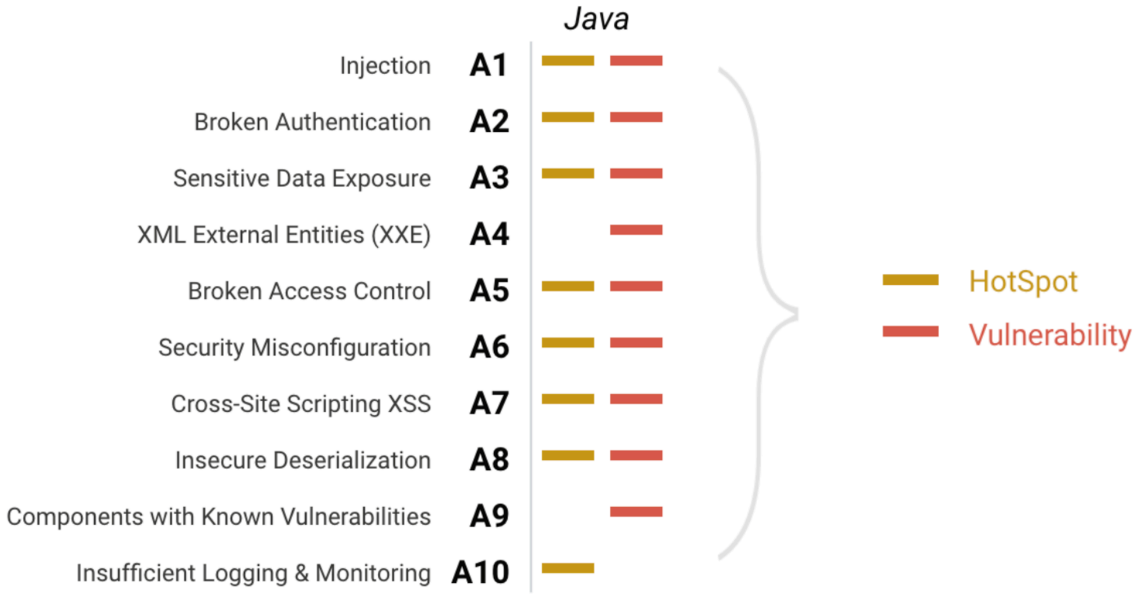


Figura 5.3: OWASP TOP TEN 2017 no SonarQube Fonte:[21]

O SonarQube também verifica a aplicação contra o *CWE Top 25*[7]. Este top identifica as 25 vulnerabilidades mais perigosas para o software. As vulnerabilidades deste top são perigosas porque são por vezes, fáceis de explorar e podem ter um impacto devastador no software como a entrada num sistema, roubo de dados importantes e confidenciais, ou parar uma aplicação.

5.3.1 Resultados

Depois de efetuado um *scan* sobre o código fonte da aplicação *PAdES Server Signer* por parte do *sonarQube*, foram identificadas as vulnerabilidades e Bugs que este possui. Os resultados que o *sonarQube* apresentou relativamente ao código fonte foram cinco *Bugs*, uma *Vulnerability* e cinco *Security Hotspots*:

1. ***Bug***: No scan foram encontrados 2 tipos de Bugs:

    a) Um "NullPointerException"ocorre quando uma variável é acedida e não está a apontar para nenhum objeto, estando a apontar para null, neste caso antes daquele objeto ser chamado deve-se verificar se este é Null.

    b) As conexões, ficheiros e classe que utilizam a interface de Java "Closeable", devem ser fechadas depois da suar utilização, e isto deve ser realizado numa clausula "finally".

[7] https://cwe.mitre.org/top25/archive/2021/2021_cwe_top25.html

Figura 5.4: *Bugs - Sonarqube*

2. ***Security Hotspots***: Em relação aos *Security Hotspots* foi necessário realizar uma triagem sobre os resultados para verificar quais destes eram vulnerabilidades e quais eram falsos-positivos. Depois de realizada a triagem apenas ficou marcada como confirmada a vulnerabilidade que assenta na categoria "Insecure Configuration"que é uma severidade de baixo risco. Neste caso, o código está a dar *print* de informação possivelmente sensível, sendo má prática usar o "printStackTrace"nos servidores de produção.

Figura 5.5: *Secutiry Hotspots - Sonarqube*

3. ***Vulnerability***: O *Scan* não apresentou nenhuma vulnerabilidade deste tipo.

Os resultados apresentados neste capitulo foram corrigidos pois apresentavam falhas de segurança que deveriam ser removidos rapidamente da aplicação.

Capítulo 6

Conclusões e Trabalho Futuro

De acordo com o que foi sido mencionado no documento, verifica-se uma crescente utilização de assinaturas eletrónicas nas organização públicas e privadas. A sua utilização assegura algumas vantagens como a facilitação dos processos para quem necessita de realizar a assinatura. Além disso, trata-se de um processo mais rápido, relativamente ao realizado na assinatura manuscrita. Neste sentido, a União Europeia tem vindo a apoiar a sua utilização apresentando regulamentação, que permite que uma assinatura eletrónica seja reconhecida como uma assinatura manuscrita. Do mesmo modo, também têm sido elaborados diversos projetos que contribuem para a implementação de aplicações que realizem assinaturas eletrónicas. Enquadrando o país nesta matéria, em Portugal, existem duas formas de realizar uma assinatura eletrónica: assinatura através de um *smartcard* ou *token* e assinatura realizada em *Cloud*. Relativamente à primeira, a chave de assinatura encontra-se na posso do assinante, tal como se verifica no CC. Ao contrário desta, na segunda assinatura referida, a chave de assinatura não está na posse do assinante, tal como acontece na CMD.

No que respeita aos objetivos que estavam destinados para este projeto, refletir-se-á sobre os mesmos, de forma a compreender se estes foram alcançados com sucesso.

Assim, o principal objetivo deste projeto consistia na elaboração de uma arquitetura que realizasse uma assinatura em formato PAdES de documentos PDF, utilizando, para isso, tanto a CMD, como o CC. Desta forma, procurava-se que esta assinatura permitisse a um utilizador realizar uma assinatura neste formato a partir do *Browser*. Através do trabalho desenvolvido, constata-se que este objetivo foi atingido, uma vez que foi realizada um arquitetura para assinaturas eletrónicas, capaz de dar resposta ao que era inicialmente pretendido. Para isto, foi apresentada uma arquitetura geral da aplicação e diversos diagramas, que demonstram a sequência do processo de assinatura, bem como todos os pedidos e respostas apresentados na íntegra do respetivo processo de assinatura. Além disso, integrou-se, nesta arquitetura, a possibilidade de apresentar a assinatura visível e definir o local da mesma.

Em relação à utilização da especificação do CSC na arquitetura, nomeadamente a especificação

"Architetures and protocols for remote siganture appliations"[3], representada na secção 2.4, esta foi integrada na arquitetura do projeto seguindo as normas que a mesma exige.Com esse propósito, o PAdES Server Signer fez uma correspondência entre os métodos que esta especificação exigia com os que estão idealizados no servidor da CMD. A adaptação ao CC foi mais difícil do que a concretizada para CMD, já que a API que a especificação do 2.4 apresenta destina-se a assinaturas realizadas na *Cloud* e não apresenta um caso especifico para o CC.

O terceiro objetivo consistia na verificação da necessidade de utilização de um *Plugin* para a comunicação com o CC. Para esse efeito, realizou-se um *Plugin* para comunicar com o CC. Este tinha como função estabelecer a comunicação entre uma Aplicação de Assinatura que se encontra-se no computador do utilizador e o *Smartcard*. Nesta fase, decidiu-se que o *Plugin* iria enviar diretamente os dados que recebesse do *smartcard* para o servidor PAdES Server Signer, para que este enviasse estes dados de seguida para a Aplicação de Assinatura.

Posteriormente, procedeu-se à utilização de plataformas que pudessem ajudar a construção de assinatura eletrónicas no formato PAdES. De uma forma breve, explica-se o contributo de cada uma delas.

1. A biblioteca ***Digital Signature Service***, disponibilizada pela União Europeia foi um grande auxilio na implementação da dissertação, uma vez que apresenta um grande leque de funcionalidades. Deste modo, auxiliou à construção de uma assinatura PAdES de um documento PDF. De facto, a União Europeia dispõe de um grande leque de opções para ajudar a realizar assinaturas eletrónicas.

2. O ***Spring Framework***, que é uma *framework open source* para Java que ajudou à construção e gestão da aplicação.

3. O ***MySQL*** foi o serviço de Base de Dados utilizado para gerir os dados da aplicação.

4. O ***poreid*** é uma biblioteca para Java que realiza a comunicação do *Plugin* com o CC. Esta foi responsável pela definição dos métodos que realizam a assinatura através do CC, como também pelo método que retorna a cadeia de certificação da assinatura qualificada do Cartão de Cidadão.

5. O ***React.js*** por se tratar de uma biblioteca para *JavaScript*, auxiliou a construção da Aplicação Web.

Em relação à implementação das componentes de software, elaborou-se, com sucesso, a codificação da Aplicação PAdES Server Signer baseada na arquitetura que tinha sido definida. Para tal, utilizaram-se as bibliotecas e *frameworks* referidas anteriormente. Como resultado, a assinatura realiza-se de acordo com o estipulado na especificação do CSC[3] ou de um documento PDF em qualquer um dos 4 formatos *PAdES*, para CC ou CMD. Para demonstrar a aplicação, foi também realizada uma aplicação Web. Esta permite comunicar com o PAdES Server Signer e realizar uma assinatura eletrónica a partir do *Browser*.

O código desta aplicação está disponivel de maneira *open-source* no github:`https://github.com/franciscoaraujo51/PadesServerSigner`. Existe ainda uma máquina virtual disponibilizada pela Devise Futures Lda., para testar a aplicação: `https://vm4.devisefutures.com/PAdESServerSigner.ova`.

O objetivo final direcionou-se para a utilização de ferramentas, com o intuito de verificar a qualidade do código desenvolvido. Neste caso, em primeiro lugar, foram implementadas medidas para aumentar a segurança do sistema. Estas medidas encontram-se documentadas no OWASP, mais propriamente no projeto *OWASP Cheat Sheet Series*[1], para REST. Nesta fase, surgiu a necessidade de criar validação de *Input* e de *Content Types*, assim como a utilização de tokens JWT para a autenticação do *PAdES Server Signer*. Além disso, também foi utilizada a *framework Spring Security* que ajudou a aumentar a segurança da aplicação. A utilização da plataforma *Spring Framework* facilitou a sua implementação. Para a realização de testes à aplicação, foram feitos *scans*, a partir do SAST *SonarQube*. No domínio da segurança do software, todos os resultados encontrados por esta ferramenta foram resolvidos, o que contribuiu para uma maior segurança do *PAdES Server Signer*. Na categoria *Code Smells*, o *SonarQube* também apresentou resultados para *Code Coverage* e *Code Duplication*, bem como pontuações para a capacidade de manutenção e segurança do código.

Quanto ao trabalho futuro, pretende-se fazer alterações à Base de Dados. Para que este serviço seja disponibilizado publicamente, devem minimizar-se os dados guardados de acordo com o RGPD, assim como avaliar os dados a manter e os que devem ser anónimos. Ademais, também devem ser verificados os resultados de *Code Smell* fornecidos pelo *SonarQube*. Posteriormente, deve proceder-se à sua alteração do código, para que estes sejam corrigidos. Deste modo, melhorar-se-á a leitura e manutenção do código.

[1] `https://cheatsheetseries.owasp.org`

Bibliografia

[1] url: https://helpx.adobe.com/hk_en/document-cloud/kb/european-union-trust-lists.html.

[2] J. Adobe. *Document management—Portable Document 493 Format—Part 1: PDF 1.7*. 2008.

[3] "Architectures and protocols for remote signature applications". Em: (2019).

[4] A. Caccia, L. Rizzo, J. C. Cruellas Ibarz, A. Funk e A. Röck. "ETSI EN 319 162-1 v1. 1.1. Associated Signature Containers (ASiC). Part 1: Building blocks and ASiC baseline containers". Em: (2016).

[5] A. Caccia, L. Rizzo, J. C. Cruellas Ibarz, A. Funk e A. Röck. "ETSI EN 319 162-2 v1. 1.1. Associated Signature Containers (ASiC). Part 2: Additional ASiC containers". Em: (2016).

[6] J. C. Cruellas Ibarz. "ETSI EN 319 122-2 v1. 1.1. CAdES signatures. Part 2: Extended CAdES signatures". Em: (2016).

[7] J. C. Cruellas Ibarz, O. Delos, N. Pope e A. Röck. "ETSI TR 119 100 v0. 0.2:"Business Driven Guidance for Signature Creation and Validation"". Em: (2013).

[8] J. C. Cruellas Ibarz, A. Röck, L. Rizzo, A. Funk e A. Caccia. "ETSI EN 319 132-1 v1. 1.1. XAdES digital signatures. Part 1: Building blocks and XAdES baseline signatures". Em: (2016).

[9] J. C. Cruellas Ibarz, A. Röck, L. Rizzo, A. Funk e A. Caccia. "ETSI EN 319 132-2 v1. 1.1. XAdES signatures. Part 2: Extended XAdES signatures". Em: (2016).

[10] *Digital Signature Service version : 5.9 - 2021-09-17*. url: https://ec.europa.eu/cefdigital/DSS/webapp-demo/doc/dss-documentation.html#_pades_signature_pdf.

[11] *e-Signature standards*. url: https://ec.europa.eu/cefdigital/wiki/display/CEFDIGITAL/Standards+and+specifications#Standardsandspecifications-XAdES(XMLAdvancedElectronicSigBaselineProfile.

[12] ETSI. "ETSI TS 119 182-1 V1.1.1. Electronic Signatures and Infrastructures (ESI); JAdES digital signatures;Part 1: Building blocks and JAdES baseline signatures". Em: (2021).

[13] R. T. Fielding. "Architectural Styles and the Design of Network-based Software Architectures". Em: (2000).

[14] A. Funk, L. Rizzo, J. C. Cruellas Ibarz, A. Röck e A. Caccia. "ETSI EN 319 142-1 v1. 1.1. PAdES digital signatures. Part 1: Building blocks and PAdES baseline signatures". Em: (2016).

[15] R. Housley. *Cryptographic message syntax (CMS)*. Rel. téc. RFC 5652, September, 2009.

[16] B. Kaliski. *PKCS# 7: Cryptographic message syntax version 1.5*. 1998.

[17] *OWASP Top 10 - 2017 - The Ten Most Critical Web Application Security Risks*. url: https://owasp.org/www-project-top-ten/.

[18] E. Parliament e of the Council. *Regulation (EU) No 910/2014 on electronic identification and trust services for electronic transactions in the internal market (eIDAS Regulation)*. 2014. url: https://eur-lex.europa.eu/legal-content/PT/TXT/?uri=celex%3A32014R0910.

[19] L. Rizzo, A. Funk, J. C. Cruellas Ibarz, A. Röck e A. Caccia. "ETSI EN 319 142-2 v1. 1.1. PAdES signatures. Part 2: Additional PAdES signatures profiles". Em: (2016).

[20] A. Röck, J. C. Cruellas Ibarz, L. Rizzo, A. Funk e A. Caccia. "ETSI EN 319 122-1 v1. 1.1. CAdES digital signatures. Part 1: Building blocks and CAdES baseline signatures". Em: (2016).

[21] *SonarQube*. url: https://www.sonarqube.org/features/security/owasp/.

[22] *Static Application Security Testing*. url: https://www.synopsys.com/glossary/what-is-sast.html.

[23] *Trusted List Portugal - Trust service providers*. url: https://esignature.ec.europa.eu/efda/tl-browser/#/screen/tl/PT.

[24] *X.690 : Information technology - ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)*. Rel. téc. ITU. url: https://www.itu.int/rec/T-REC-X.690.

Apêndice A

Pedidos para o servidor do CMD

A.1 SMDMultipleSign

Listagem A.1: Pedido SMDMultipleSign

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:ns1="http
    ↪ ://Ama.Authentication.Service/">
<SOAP-ENV:Header/>
<SOAP-ENV:Body>
   <ns1:CCMovelMultipleSign>
      <ns1:request xmlns:ns2="http://schemas.datacontract.org/2004/07/Ama.Structures.
          ↪ CCMovelSignature">
         <ns2:ApplicationId>APPLICATIONID</ns2:ApplicationId>
         <ns2:Pin>1111</ns2:Pin>
         <ns2:UserId>+351 966666666</ns2:UserId>
      </ns1:request>
      <ns1:documents xmlns:ns2="http://schemas.datacontract.org/2004/07/Ama.Structures.
          ↪ CCMovelSignature">
         <ns2:HashStructure>
            <ns2:Hash>
               MDEwDQYJYIZIAWUDBAIBBQAEIAwt4XYWo4tKQFgCEiCpUqMH2eKtdUsQyAlEx2xr/uYi
            </ns2:Hash>
            <ns2:Name>Padesdsstester1</ns2:Name>
            <ns2:id>Padesdsstester1</ns2:id>
         </ns2:HashStructure>
         <ns2:HashStructure>
            <ns2:Hash>
               MDEwDQYJYIZIAWUDBAIBBQAEIAwt4XYWo4tKQFgCEiCpUqMH2eKtdUsQyAlEx2xr/uYi
            </ns2:Hash>
            <ns2:Name>Padesdsstester2</ns2:Name>
```

```
            <ns2:id>Padesdsstester2</ns2:id>
         </ns2:HashStructure>
      </ns1:documents>
   </ns1:CCMovelMultipleSign>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Listagem A.2: Resposta SMDMultipleSign

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
<s:Body>
   <CCMovelMultipleSignResponse xmlns="http://Ama.Authentication.Service/">
      <CCMovelMultipleSignResult xmlns:a="http://schemas.datacontract.org/2004/07/Ama.
          ↪ Structures.CCMovelSignature" xmlns:i="http://www.w3.org/2001/XMLSchema-instance">
         <a:Code>200</a:Code>
         <a:Field i:nil="true"/>
         <a:FieldValue i:nil="true"/>
         <a:Message>Success! Next step: invoke ValidateOTP</a:Message>
         <a:ProcessId>ba3a2be5-d305-41d3-b59a-bd32e6defb43</a:ProcessId>
      </CCMovelMultipleSignResult>
   </CCMovelMultipleSignResponse>
</s:Body>
</s:Envelope>
```

A.2 GetCertificate

Listagem A.3: Pedido GetCertificate

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:ns1="http
    ↪ ://Ama.Authentication.Service/"><SOAP-ENV:Header/>
   <SOAP-ENV:Body>
      <ns1:GetCertificate>
         <ns1:applicationId>APPLICATIONID</ns1:applicationId>
         <ns1:userId>+351 966666666</ns1:userId>
      </ns1:GetCertificate>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Listagem A.4: Resposta GetCertificate

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
<s:Body>
<GetCertificateResponse xmlns="http://Ama.Authentication.Service/">
   <GetCertificateResult>-----BEGIN CERTIFICATE-----
   MIIKBTCCB+2gAwIBAgIIE1GXnjvfrpIwDQYJKoZIhvcNAQELBQAwgekxCzAJBgNVBAYTAlBUMUIw&#xD;
```

```
QAYDVQQKDDlBTUEgLSBBR8OKTkNJQSBQQVJBIEEgTU9ERVJOSVpBw4fDg08gQURNSU5JU1RSQVRJ&#xD;
VkEgSS4gUC4xHDAaBgNVBAsME0NhcnTDo28gZGUgQ2lkYWTDo28xFDASBgNVBAsMC3N1YkVDRXN0&#xD;
YWRvMWIwYAYDVQQDDFlFQyBkZSBDaGF2ZSBNw7N2ZWwgRGlnaXRhbCBkZSBBc3NpbmF0dXJhIERp&#xD;
Z2l0YWwgUXVhbGlmaWNhZGEgZG8gQ2FydMOjbyBkZSBDaWRhZMOjbyAwMDAwMjAeFw0yMDAzMDcw&#xD;
MzI3MThaFw0yMzA0MTgyMTU5MDBaMIIBGTELMAkGA1UEBhMCUFQxHDAaBgNVBAoME0NhcnTDo28g&#xD;
ZGUgQ2lkYWTDo28xQzBBBgNVBAsMOkNoYXZlIE3Ds3ZlbCBEaWdpdGFsIGRlIEFzc2luYXR1cmEg&#xD;
UXVhbGlmaWNhZGEgZG8gQ2lkYWTDo28xETAPBgNVBAsMCENpZGFkw6NvMR0wGwYDVQQLDBRSZW1v&#xD;
dGVRU0NETWFuYWdlbWVudDEXMBUGA1UEBAwOVklMRUxBIEFSQcOaSk8xGzAZBgNVBCoMEkZSQU5D&#xD;
SVNDTyBGRVJOQU5ETzETMBEGA1UEBRMKQkkxNTM1OTcxOTEqMCgGA1UEAwwhRlJBTkNJU0NPIEZF&#xD;
Uk5BTkRPIFZJTEVMQSBBUkHDmkpPMIIBojANBgkqhkiG9w0BAQEFAAOCAY8AMIIBigKCAYEAtZca&#xD;
mqzMRrbu92nlFN6AGCyihl76oG3SusV9M1fEp0kLX24YiDh9mw1rKJWrikZ5aeaYDXosf3WPSmCq&#xD;
wQhkt6RWApXwIC8ods6Vy2yS0zhboe/iLQdgQCAIP9z6STPh3yRAPV0l45W2GJMJg3UOXq6D1Cgo&#xD;
Peh8FjzJ5Pnf7QBKE/d1F51K1qLdMc7CHhwSt/3aVoN6eBWDIs8EYRxbEF4/EwhlEFT02prubCLs&#xD;
rd6QyG/OOrTxzdeeHuPiHqhIxjK49W7yN/lqVvj2pg1Nmp+bcg4MzR90CiFGN6i4iAKkBCVbNif0&#xD;
hE4jjhrnWs8ZbLocCK0J06LTrcddA0e84fDSNP1MNPJXOKsyQIUq1Ac9zwzsXyof/syWZILCEzs7&#xD;
jFSWV23mvjyYko33Kph0PUZVsFo7V21WBBy0lseAz0PTUHmwuNi4mTFjX3Ors1Ukr5IpCD1Lm+GO&#xD;
RC6M31abCZyAYGrV+Db3Z1rtD+KRwiHaVORTmivSn+JYrOPnrTA5AgMBAAGjggP8MIID+DAMBgNV&#xD;
HRMBAf8EAjAAMB8GA1UdIwQYMBaAFPVzPi2J1qTWgD7dz9lkCYAgpu+cMEsGCCsGAQUFBwEBBD8w&#xD;
PTA7BggrBgEFBQcwAYYvaHR0cDovL29jc3AuY21kLmNhcnRhb2RlY2lkYWRhby5wdC9wdWJsaWNv&#xD;
L29jc3AwaAYDVR0uBGEwXzBdoFugWYZXaHR0cDovL3BraS5jYXJ0YW9kZWNpZGFkYW8ucHQvcHVi&#xD;
bGljby9scmMvY2Nfc3ViLWVjX2NpZGFkYW9fY21kX2NybDAwMDJfZGVsdGFfcDAwMDEuY3JsMIIB&#xD;
OwYDVR0gBIIBMjCCAS4wNgYIYIRsAQEBAgowKjAoBggrBgEFBQcCARYcaHR0cDovL3d3dy5zY2Vl&#xD;
Lmdvdi5wdC9wY2VydDAJBgcEAIvsQAECMG4GDGCEbAEBAQIEAwABATBeMFwGCCsGAQUFBwIBFlBo&#xD;
dHRwOi8vcGtpLmNhcnRhb2RlY2lkYWRhby5wdC9wdWJsaWNvL3BvbGl0aWNhcy9wYy9jY19zdWIt&#xD;
ZWNfY2lkYWRhb19jbWRfcGMuaHRtbDBvBgtghGwBAQECBAMABzBgMF4GCCsGAQUFBwIBFlJodHRw&#xD;
Oi8vcGtpLmNhcnRhb2RlY2lkYWRhby5wdC9wdWJsaWNvL3BvbGl0aWNhcy9kcGMvY2Nfc3ViLWVj&#xD;
X2NpZGFkYW9fY21kX2RwYy5odG1sMAgGBgQAj3oBAjAoBgNVHQkEITAfMB0GCCsGAQUFBwkBMREY&#xD;
DzE5OTcxMDAzMTIwMDAwWjCCARIGCCsGAQUFBwEDBIIBBDCCAQAwCAYGBACORgEBMAgGBgQAjkYB&#xD;
BDBaBgcEAI5GAQYBDE9DZXJ0aWZpY2F0ZSBmb3IgZWxlY3Ryb25pYyBzaWduYXR1cmVzIGFzIGRl&#xD;
ZmluZWQgaW4gUmVndWxhdGlvbiAoRVUpIE5vIDkxMC8yMDE0MIGNBgYEAI5GAQUwgYIwPxY5aHR0&#xD;
cHM6Ly9wa2kuY2FydGFvZGVjaWRhZGFvLnB0L3B1YmxpY28vcG9saXRpY2FzL2Nwcy5odG1sEwJQ&#xD;
VDA/FjlodHRwczovL3BraS5jYXJ0YW9kZWNpZGFkYW8ucHQvcHVibGljby9wb2xpdGljYXMvY3Bz&#xD;
Lmh0bWwTAkVOMGIGA1UdHwRbMFkwV6BVoFOGUWh0dHA6Ly9wa2kuY2FydGFvZGVjaWRhZGFvLnB0&#xD;
L3B1YmxpY28vbHJjL2NjX3N1Yi1lY19jaWRhZGFvX2NtZF9jcmwwMDAyX3AwMDAxLmNybDAdBgNV&#xD;
HQ4EFgQUXeCSWShyv911UdFbuCCjy+CukIIwDgYDVR0PAQH/BAQDAgZAMA0GCSqGSIb3DQEBCwUA&#xD;
A4ICAQBcxGOl39XGMd0TPnRZJetZ7CW9WkWZATyUY0esLWIcfTyJAReo0hPMMb0BvpcchUPaVUu+&#xD;
JTtXshb1BzarcaoI6t+/wgx5d3brV/9kfqnP3FHrQXX87uTdxeOEjEXuOoM16kg5YkASrP3m230u&#xD;
Hrlpv8m2+nsqM371q1EboyPgVC0sWJji4axa1E8Yo+F92LW+RNBmEEamsukGb5s7MG3o6aQ06jNB&#xD;
CN73D4j0osxgegkx/3Ks5WySwcHMlndi7iMvUMMboTA0HOgvSbliyUWTaH0MiBO7kNiRTnnbmf6p&#xD;
fkPnFjZN6n8dZ5OTUcRcSzRdGfgZBb5M/EP2fbx5OgxEqNFgudNkFAx8vSUPSzX30eVZ20vRKjuU&#xD;
dDSNQmCgUBdDBMXgU8n9RR5ZaO6CMhwTCIJmvofUDRKmgSj5u54QNEepCZFuZ9jZuEWLR5qV2LrY&#xD;
5TsQBEP9hqQA2cSbt93Cqn5BQLTsJHkO8RrwsAdWGRb+WzK4iO9q5opPzd2GX0hp9cp/qN/Ymucu&#xD;
mwury+1nu/abdaKtNVjgDUvJvDkLHqaglr1o7XBD2HIDUAvl2rWAq4MWXzX57p3uoFxQTfNVM4Kj&#xD;
aasnG0YOB+Ff1MmDg0OZsh9h4BlV5bn/qAYEcZNbr/epMVBbWR2PM8K71+cC6N/elSav/uwKyztp&#xD;
Y6mB6A==&#xD;
```

```
-----END CERTIFICATE-----

-----BEGIN CERTIFICATE-----
MIIGgzCCBGugAwIBAgIQQbhynRSongpdMfUrdkKb+TANBgkqhkiG9w0BAQsFADAzMQswCQYDVQQG&#xD;
EwJQVDENMAsGA1UECgwEU0NFRTEVMBMGA1UEAwwMRUNSYWl6RXN0YWRvMB4XDTE5MDcxOTE2NTE1&#xD;
NVoXDTMwMDYxOTE2NTE1NVowgYQxCzAJBgNVBAYTAlBUMUAwPgYDVQQKDDdTQ0VFIC0gU2lzdGVt&#xD;
YSBkZSBDZXJ0aWZpY2HDp8OjbyBFbGVjdHLDs25pY2EgZG8gRXN0YWRvMREwDwYDVQQLDAhFQ0Vz&#xD;
dGFkbzEgMB4GA1UEAwwXQ2FydMOjbyBkZSBDaWRhZMOjbyAwMDUwggIiMA0GCSqGSIb3DQEBAQUA&#xD;
A4ICDwAwggIKAoICAQCniDybK6adl7PFqNLJWsy8C667YCRYCiWF64CXkHXU0Nfwgpc9oxClplM+&#xD;
Tn/PpDA6rtIrMu/tq7sZfu/5oyLLJAnvbQfq4gQeyjORSmtDRcjzbsoH/lOg2xzlJRFkKjDPuk0G&#xD;
ujLrfbcS0hlD4ep2iPgwhGJ8hPiH/c4djiZMw+6hdmIb/7HzwpEO4YYgMC7wFK7Yg5oW0sogRuN/&#xD;
sfnqJHY9RIvVplvT/Ldo3WtaNAIpDt0BuwdSSROA1ZRnvkKO8n65D1Sp5BjMcKjyuVdwliXMfDF5&#xD;
m1sZColTGvw9uhulp13wsUdooSMquptzrUN0XiQExynuTrMHYZtyjhCBnafNFOTYTKI/yuus1LwO&#xD;
ArWrc07p7WqDp1W9qOG9se4PPKsP5tEBgqZvTXvrC9fiPpeL67Sq+UB24OXYzkk29kXUxxLh8RTu&#xD;
RP6VeeJ0HbSRp1/vZSMkjCR/rObvgnTMeyy9Bn7PTyuyUHZRP90fNsfcpGt6O3uKysW6rVgPBpDm&#xD;
mgc4iHR9xgX+k4mTvQxpj+//zT0roXbAQl0rX7JBEgOWd3l2ldOlgW1wmEEWjX15Ou4w2FAXuh9j&#xD;
mPXKn0GphG4wA0ybNhVpXRyKDoh5a6V8/UZfqs98bF4dwMZVQjWHkqFj2bNP55r/zMJggK5Ulc/n&#xD;
WiLfhdptbTk1S/PEIQIDAQABo4IBPzCCATswDwYDVR0TAQH/BAUwAwEB/zAdBgNVHQ4EFgQU9suD&#xD;
so8DYFqMZFx9kdFOtTlm4SkwHwYDVR0jBBgwFoAUcX813vV3cW0dEpzhkKS68KmDj4AwDgYDVR0P&#xD;
AQH/BAQDAgEGMDsGA1UdIAQ0MDIwMAYEVR0gADAoMCYGCCsGAQUFBwIBFhpodHRwOi8vd3d3LmVj&#xD;
ZWUuZ292LnB0L2RwYzBkBggrBgEFBQcBAQRYMFYwIwYIKwYBBQUHMAGGF2h0dHA6Ly9vY3NwLmVj&#xD;
ZWUuZ292LnB0MC8GCCsGAQUFBzAChiNodHRwOi8vdHJ1c3QuZWNlZS5nb3YucHQvZWNyYWl6LmNy&#xD;
dDA1BgNVHR8ELjAsMCqgKKAmhiRodHRwOi8vY3Jscy5lY2VlLmdvdi5wdC9jcmxzL0FSTC5jcmww&#xD;
DQYJKoZIhvcNAQELBQADggIBAG2HFjdAAcaCnMX3OJ4kUKx7VEhIr1SItQEWNkTDwqZfJ04NQwLG&#xD;
bIezlsxLIq4VAmpBLk4vfoBApDz4HOLgUVYUYSchygp/wGqqLswRwwarvLn5V63+owP7AMY+deQo&#xD;
HJ+cdFE4elpF9rXinkXsDJCotV/+vPkW9qgM/GKIqrtta+9aWv9XPyzGSfW/Qb7oC4J8oyAa2+1L&#xD;
GRMsSEcc7VLdybhodbMLKWrZVlgm8XvXzq02cAgq50/unqK9HrUjPtXPojiQt4xFypcekXO7Nc4A&#xD;
paJBFrX+DbjBn/5WEbckVZ2HiY4N9D5T4/M57UryH+Zh5CPhuww+ml+A5nRZPvVop7VtJrmbMDV9&#xD;
/4GSygc80xvOktFLIgzusk72NpWTyDBQ4Jj1zICXGbHnENvv3sj7dTHHn/Ja66WOb460xe8Cgwct&#xD;
tBuArc2mYNb32x2LT54J5hBuVvvwLWukArSXvs37NlX2UY6acCb72iO03ZOn6vhq1M9EyRpn8rNc&#xD;
CdH7STxHNlrmqcdRgdGUejcAzy9aS0G2XvXi73SZX6KnzXwiEAD9+40i4M9iH41yMs4Y4kwABRGU&#xD;
GTsGehHVzv2HXyJ2+ry+E58QPuKfXySTAfxPs4UBB3HDAKj9/w3Oc2lgZpcKg81qr43PpwqQXB/f&#xD;
n4TsFr403WWdKwbVnjBOeYvr&#xD;
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
MIIH8TCCBdmgAwIBAgIIQ9bwUxVebGwwDQYJKoZIhvcNAQELBQAwgYQxCzAJBgNVBAYTAlBUMUAw&#xD;
PgYDVQQKDDdTQ0VFIC0gU2lzdGVtYSBkZSBDZXJ0aWZpY2HDp8OjbyBFbGVjdHLDs25pY2EgZG8g&#xD;
RXN0YWRvMREwDwYDVQQLDAhFQ0VzdGFkbzEgMB4GA1UEAwwXQ2FydMOjbyBkZSBDaWRhZMOjbyAw&#xD;
MDUwHhcNMTkwNTI5MTA0ODA5WhcNMzEwNTI5MTA0ODA5WjCB6TELMAkGA1UEBhMCUFQxQjBABgNV&#xD;
BAoMOUFNQSAtIEFHw4pOQ0lBIFBBUkEgQSBNT0RFUk5JWkHDh8ODTyBBRE1JTklTVFJBVElWQSBJ&#xD;
LiBQLjEcMBoGA1UECwwTQ2FydMOjbyBkZSBDaWRhZMOjbzEUMBIGA1UECwwLc3ViRUNFc3RhZG8x&#xD;
YjBgBgNVBAMMWUVDIGRlIENoYXZlIE3Ds3ZlbCBEaWdpdGFsIGRlIEFzc2luYXR1cmEgRGlnaXRh&#xD;
bCBRdWFsaWZpY2FkYSBkbyBDYXJ0w6NvIGRlIENpZGFkw6NvIDAwMDAyMIICIjANBgkqhkiG9w0B&#xD;
AQEFAAOCAg8AMIICCgKCAgEA6om1x4cU+qtwaBZAbwdsGuKXveXxcpyN/sRKKQPstBaBHolDnC/9&#xD;
0ghHMEwO95HH8UBKMahVbW+T7kG7UomUczrTy4pjCRWnIWbBQNmgcSSwK67iaENEZZWENuGUJBtm&#xD;
```

```
96    b2ZUqK8baW3eYN9Yq8xL3vPGD3WAaDiGcPjxc/3BxyfLHIddzfDNPyX7qOEEy4ZPkNGkaCAObri4&#xD;
97    XrnBNRQ8sE0imiLAJ821fab/DZBNbCd8WJU2MwOK0grOjmzqxJhVt5HDcz2pT6Aqh6bvxwHuQ7Ib&#xD;
98    gcGXusbpU62W9/Rb71bq9QHkGaCJEe/l0YNzWOy2dWSzUh8issu1sHKU8a0oyU7fbl0fmnsndKi0&#xD;
99    M7mD9EqFm+3QU4LW0Gxe/hlppgwkmGucRr+qd72hVUf9EmYkzAPGDs26tp3xN+fzGLczU7/4binW&#xD;
100   HfEREmHuVpO6vB47vr1HnMBSUuZgzQKcGz40lEl/itii1ICdEvYxn+a6Z4SzlVn8ZzpoT0EjwLSa&#xD;
101   1mLBKy6IeIk0zOdf2Tf1foaldeb2AUqdaNut2pbXF7tWJcPQgeaRB7DDZ6hLIVvhkcz8l4AXHYr2&#xD;
102   IXt58CL6HT/TFJKeMjjE0M1w9oEwiwIkX359/80p+IIKM8g8OTNFpF88YxhAKeowZ+jc3hj5J8A0&#xD;
103   oewdjD7hNXJVgmlSw3v0lG8CAwEAAaOCAf4wggH6MEwGCCsGAQUFBwEBBEAwPjA8BggrBgEFBQcw&#xD;
104   AYYwaHR0cDovL29jc3Aucm9vdC5jYXJ0YW9kZWNpZGFkYW8ucHQvcHVibGljby9vY3NwMB0GA1Ud&#xD;
105   DgQWBBT1cz4tidak1oA+3c/ZZAmAIKbvnDASBgNVHRMBAf8ECDAGAQH/AgEAMB8GA1UdIwQYMBaA&#xD;
106   FPbLg7KPA2BajGRcfZHRTrU5ZuEpMIHsBgNVHSAEgeQwgeEwVQYLYIRsAQEBAgQAAQkwRjBEBggr&#xD;
107   BgEFBQcCARY4aHR0cHM6Ly9wa2kuY2FydGFvZGVjaWRhZGFvLnB0L3B1YmxpY28vcG9saXRpY2Fz&#xD;
108   L2NwLmh0bWwwVQYKYIRsAQEBAgQABzBHMEUGCCsGAQUFBwIBFjlodHRwczovL3BraS5jYXJ0YW9k&#xD;
109   ZWNpZGFkYW8ucHQvcHVibGljby9wb2xpdGljYXMvY3BzLmh0bWwwMQYEVR0gADApMCcGCCsGAQUF&#xD;
110   BwIBFhtodHRwczovL3d3dy5zY2VlLmdvdi5wdC9yZXAwVwYDVR0fBFAwTjBMoEqgSIZGaHR0cDov&#xD;
111   L3BraS5jYXJ0YW9kZWNpZGFkYW8ucHQvcHVibGljby9scmMvY2NfZWNfY2lkYWRhb19jcmwwMDVf&#xD;
112   Y3JsLmNybDAOBgNVHQ8BAf8EBAMCAQYwDQYJKoZIhvcNAQELBQADggIBADHCWQNScOXXIL8xN0hf&#xD;
113   6GTHhDd5tX6wioULIJ2scxI3qQzEwyGvWNV258AcN/py+e12d6Nf/o1SmrPLqrRZl5E0Of0BE6rM&#xD;
114   O6m290FbIgN4Vw9dScrZgGIyEndtiAfcGLOKkyzHdFcjlKziRfmvdwlr5HIFx3LeDB/PMnVTF6bm&#xD;
115   VVlJg14LyV43uVr6qAPP7GIBBqO8zTUtF5aYRDJEaLhsL7wBSImhp4VacXtw57rJg4HkSCyHssh5&#xD;
116   ziVnWPgXS4LPdkhwcWGUWP6hNeUUNSZ7i4lmitd3rvVYEHHFpjC8PHNn31g6jHnNSIYrZSiVGe7i&#xD;
117   gbcabU8P6uHnhZT7eYe5Dl1y3G4ApR6U6F8oAfnM7U/xcp2DPa3OA7rXgnTo7BqfAAaPOHZjtQSA&#xD;
118   WUu8I1N/l81goz5n2vzztLCC7BDDF1IYJ9rWfbT/EdFCvYf6LYUskfki78IGaWWBNJG5T2F5/xTp&#xD;
119   HYkeXQzpB/iB6b/uH7adyQsLz+jqnvpmSPBWGLims/SLWBTeNnideZuq6eI1604aphQ1H+u2Y3Ap&#xD;
120   dlg/PqUBbT4X/MKPv3S4tfeJOILqg4GXc0cy4y/98FHtbxkfTsve1FiypFVdUOTrv0vXy6IrgE/l&#xD;
121   BLNd/l+kngVmSQBJ4nNelp269YaSWIOe9KRMVZga30RSEIPxSeirmZ2e&#xD;
122   -----END CERTIFICATE-----
123   </GetCertificateResult>
124 </GetCertificateResponse>
125 </s:Body>
126 </s:Envelope>
```

A.3 ValidateOtp para SMDMultipleSign

Listagem A.5: Pedido ValidateOtp

```
1  <SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:ns1="http
       ↪ ://Ama.Authentication.Service/"><SOAP-ENV:Header/>
2     <SOAP-ENV:Body>
3        <ns1:ValidateOtp>
4           <ns1:code>054645</ns1:code>
5           <ns1:processId>ba3a2be5-d305-41d3-b59a-bd32e6defb43</ns1:processId>
6           <ns1:applicationId>APPLICATIONID</ns1:applicationId>
7        </ns1:ValidateOtp>
```

```
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Listagem A.6: Resposta ValidateOtp

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
<s:Body><ValidateOtpResponse xmlns="http://Ama.Authentication.Service/"><ValidateOtpResult
    ↪ xmlns:a="http://schemas.datacontract.org/2004/07/Ama.Structures.CCMovelSignature" xmlns
    ↪ :i="http://www.w3.org/2001/XMLSchema-instance">
   <a:ArrayOfHashStructure>
      <a:HashStructure>
         <a:Hash>SIGNATURE1</a:Hash>
         <a:Name>Padesdsstester1</a:Name>
         <a:id>9030369</a:id>
      </a:HashStructure>

      <a:HashStructure>
         <a:Hash>SIGNATURE2</a:Hash>
         <a:Name>Padesdsstester2</a:Name>
         <a:id>9030370</a:id>
      </a:HashStructure>
   </a:ArrayOfHashStructure>
   <a:Signature i:nil="true"/>
   <a:Status>
      <a:Code>200</a:Code>
      <a:Field i:nil="true"/>
      <a:FieldValue i:nil="true"/>
      <a:Message>OTP code is valid</a:Message>
      <a:ProcessId>ba3a2be5-d305-41d3-b59a-bd32e6defb43</a:ProcessId>
   </a:Status>
   <a:certificate i:nil="true"/
</ValidateOtpResult>
</ValidateOtpResponse>
</s:Body>
</s:Envelope>
```

Apêndice B

Diagramas

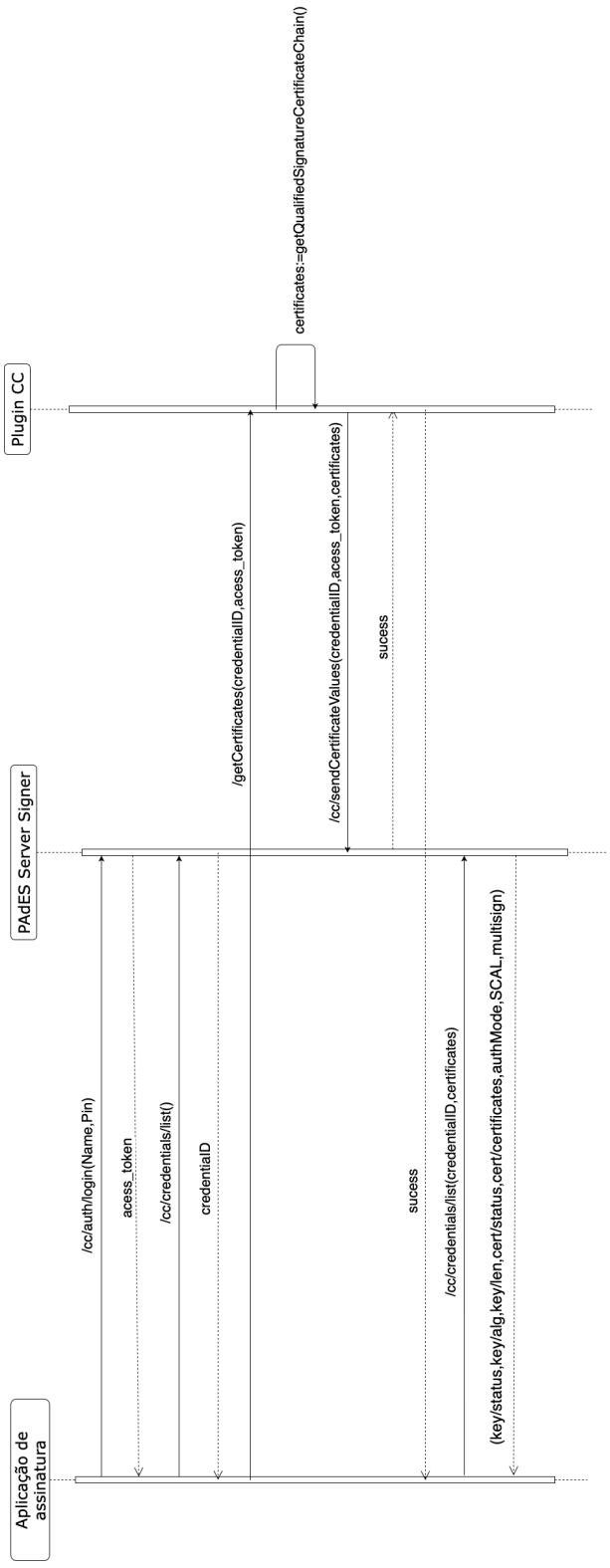


Figura B.1: Diagrama de Sequência do pedido /credentials/list do Cartão de Cidadão

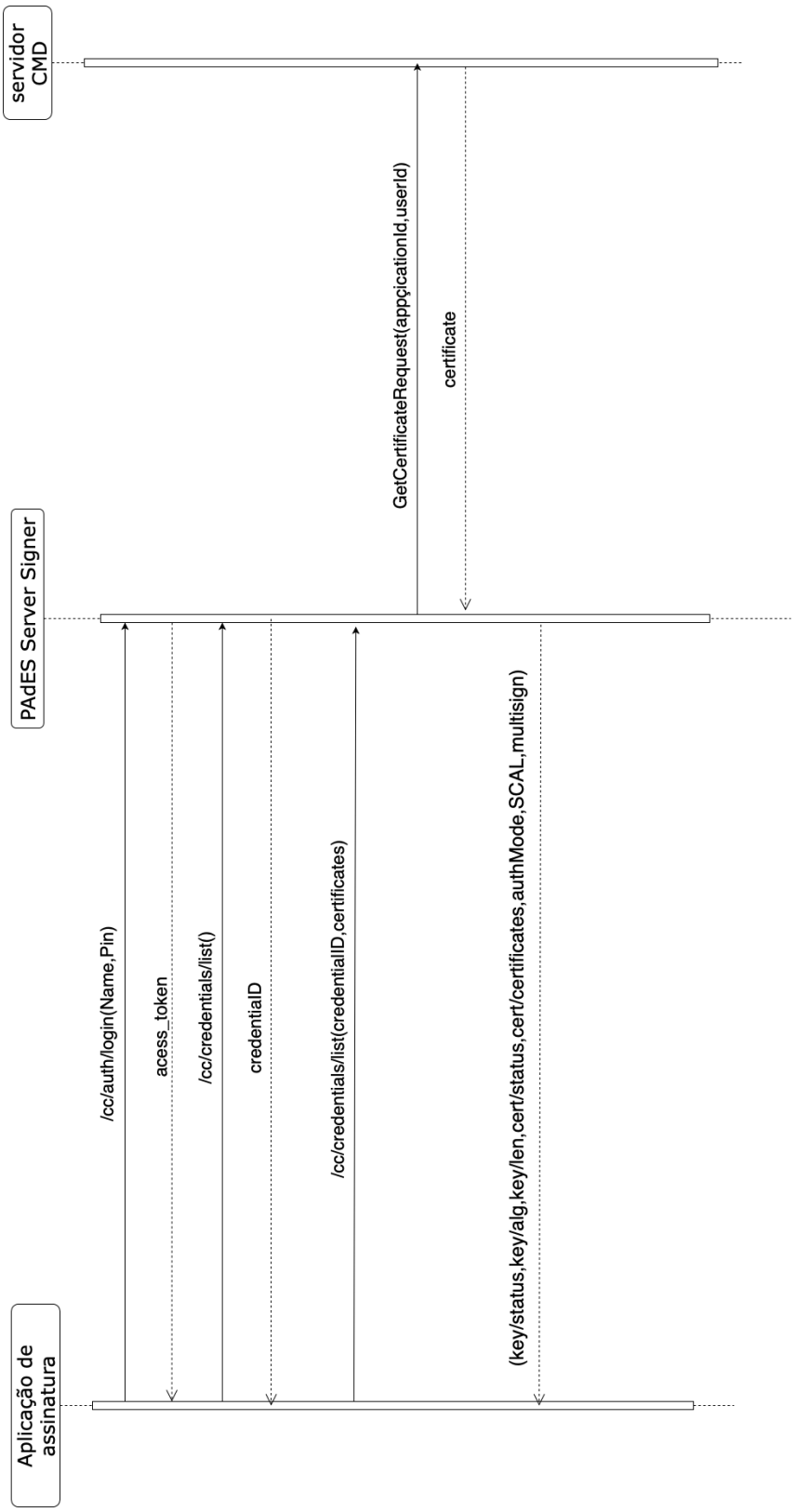


Figura B.2: Diagrama de Sequência do pedido /credentials/list do Chave Móvel Digital

Apêndice C

Pedidos e Respostas do PAdES Server Signer

C.1 auth/login

Listagem C.1: Pedido auth/login

```
{
   "phoneNumber": "+351 913043028",
   "pin":"1234"
}
```

Listagem C.2: Resposta auth/login

```
{
   "access_token": "eyJhbGciOiJIUzI1NiJ9.
       ↪ eyJzdWIiOiI1MmQzNDY5Ni1lYTljLTRiNWQtYmRiZi03NDcxNWVlZD
    IzNDgiLCJleHAiOjE2NDMxOTQwODIsImlhdCI6MTY0MzE1ODA4Mn0.
    ↪ rrgWHcOuEWF2C4vhsPA1as2BYCCd1BztSyRvJmoyo-k"
}
```

C.2 credentials/list

Listagem C.3: Resposta credentials/list

```
{
   "credentialID": [
      "0a629801-4e16-4d8b-9465-131080ebf19b"
   ]
}
```

C.3 credentials/info

Listagem C.4: Pedido credentials/info

```
{
   "credentialID": "0a629801-4e16-4d8b-9465-131080ebf19b",
   "certificates":"single"
}
```

Listagem C.5: Resposta credentials/info

```
{
   "key": {
      "status": "enable",
      "algo": "1.2.840.113549.1.1.11",
      "len": "3072"
   },
   "cert": {
      "status": "valid",
      "certificates": "-----BEGIN CERTIFICATE-----
        \nMIIKBTCCB+2gAwIBAgIIE1GXnjvfrpIwDQYJKoZIhvcNAQELBQAwgekxCzAJBgNVBAYTAlBUMUIw\r\n
        QAYDVQQKDDlBTUEgLSBBR8OKTkNJQSBQQVJBIEEgTU9ERVJOSVpBw4fDg08gQURNSU5JU1RSQVRJ\r\n
        VkEgSS4gUC4xHDAaBgNVBAsME0NhcnTDo28gZGUgQ2lkYWTDo28xFDASBgNVBAsMC3N1YkVDRXN0\r\n
        YWRvMWIwYAYDVQQDDFlFQyBkZSBDaGF2ZSBNw7N2ZWwgRGlnaXRhbCBkZSBBc3NpbmF0dXJhIERp\r\n
        Z2l0YWwgUXVhbGlmaWNhZGEgZG8gQ2FydMOjbyBkZSBDaWRhZMOjbyAwMDAwMjAeFw0yMDAzMDcw\r\n
        MzI3MThaFw0yMzA0MTgyMTU5MDBaMIIBGTELMAkGA1UEBhMCUFQxHDAaBgNVBAoME0NhcnTDo28g\r\n
        ZGUgQ2lkYWTDo28xQzBBBgNVBAsMOkNoYXZlIE3Ds3ZlbCBEaWdpdGFsIGRlIEFzc2luYXR1cmEg\r\n
        UXVhbGlmaWNhZGEgZG8gQ2lkYWTDo28xETAPBgNVBAsMCENpZGFkw6NvMR0wGwYDVQQLDBRSZW1v\r\n
        dGVRU0NETWFuYWdlbWVudDEXMBUGA1UEBAwOVklMRUxBIEFSQcOaSk8xGzAZBgNVBCoMEkZSQU5D\r\n
        SVNDTyBGRVJOQU5ETzETMBEGA1UEBRMKQkkxNTM1OTcxOTEqMCgGA1UEAwwhRlJBTkNJU0NPIEZF\r\n
        Uk5BTkRPIFZJTEVMQSBBUkHDmkpPMIIBojANBgkqhkiG9w0BAQEFAAOCAY8AMIIBigKCAYEAtZca\r\n
        mqzMRrbu92nlFN6AGCyihl76oG3SusV9M1fEp0kLX24YiDh9mw1rKJWrikZ5aeaYDXosf3WPSmCq\r\n
        wQhkt6RWApXwIC8ods6Vy2yS0zhboe/iLQdgQCAIP9z6STPh3yRAPV0l45W2GJMJg3UOXq6D1Cgo\r\n
        Peh8FjzJ5Pnf7QBKE/d1F51K1qLdMc7CHhwSt/3aVoN6eBWDIs8EYRxbEF4/EwhlEFT02prubCLs\r\n
        rd6QyG/OOrTxzdeeHuPiHqhIxjK49W7yN/lqVvj2pg1Nmp+bcg4MzR90CiFGN6i4iAKkBCVbNif0\r\n
        hE4jjhrnWs8ZbLocCK0J06LTrcddA0e84fDSNP1MNPJXOKsyQIUq1Ac9zwzsXyof/syWZILCEzs7\r\n
        jFSWV23mvjyYko33Kph0PUZVsFo7V21WBBy0lseAz0PTUHmwuNi4mTFjX3Ors1Ukr5IpCD1Lm+GO\r\n
        RC6M31abCZyAYGrV+Db3Z1rtD+KRwiHaVORTmivSn+JYrOPnrTA5AgMBAAGjggP8MIID+DAMBgNV\r\n
        HRMBAf8EAjAAMB8GA1UdIwQYMBaAFPVzPi2J1qTWgD7dz9lkCYAgpu+cMEsGCCsGAQUFBwEBBD8w\r\n
        PTA7BggrBgEFBQcwAYYvaHR0cDovL29jc3AuY21kLmNhcnRhb2RlY2lkYWRhby5wdC9wdWJsaWNv\r\n
        L29jc3AwaAYDVR0uBGEwXzBdoFugWYZXaHR0cDovL3BraS5jYXJ0YW9kZWNpZGFkYW8ucHQvcHVi\r\n
        bGljby9scmMvY2Nfc3ViLWVjX2NpZGFkYW9fY21kX2NybDAwMDJfZGVsdGFfcDAwMDEuY3JsMIIB\r\n
        OwYDVR0gBIIBMjCCAS4wNgYIYIRsAQEBAgowKjAoBggrBgEFBQcCARYcaHR0cDovL3d3dy5zY2Vl\r\n
        Lmdvdi5wdC9wY2VydDAJBgcEAIvsQAECMG4GDGCEbAEBAQIEAwABATBeMFwGCCsGAQUFBwIBFlBo\r\n
        dHRwOi8vcGtpLmNhcnRhb2RlY2lkYWRhby5wdC9wdWJsaWNvL3BvbGl0aWNhcy9wYy9jY19zdWIt\r\n
        ZWNfY2lkYWRhb19jbWRfcGMuaHRtbDBvBgtghGwBAQECBAMABzBgMF4GCCsGAQUFBwIBFlJodHRw\r\n
        Oi8vcGtpLmNhcnRhb2RlY2lkYWRhby5wdC9wdWJsaWNvL3BvbGl0aWNhcy9kcGMvY2Nfc3ViLWVj\r\n
```

```
        X2NpZGFkYW9fY21kX2RwYy5odG1sMAgGBgQAj3oBAjAoBgNVHQkEITAfMB0GCCsGAQUFBwkBMREY\r\n
        DzE5OTcxMDAzMTIwMDAwWjCCARIGCCsGAQUFBwEDBIIBBDCCAQAwCAYGBACORgEBMAgGBgQAjkYB\r\n
        BDBaBgcEAI5GAQYBDE9DZXJ0aWZpY2F0ZSBmb3IgZWxlY3Ryb25pYyBzaWduYXR1cmVzIGFzIGRl\r\n
        ZmluZWQgaW4gUmVndWxhdGlvbiAoRVUpIE5vIDkxMC8yMDE0MIGNBgYEAI5GAQUwgYIwPxY5aHR0\r\n
        cHM6Ly9wa2kuY2FydGFvZGVjaWRhZGFvLnB0L3B1YmxpY28vcG9saXRpY2FzL2Nwcy5odG1sEwJQ\r\n
        VDA/FjlodHRwczovL3BraS5jYXJ0YW9kZWNpZGFkYW8ucHQvcHVibGljby9wb2xpdGljYXMvY3Bz\r\n
        Lmh0bWwTAkVOMGIGA1UdHwRbMFkwV6BVoFOGUWh0dHA6Ly9wa2kuY2FydGFvZGVjaWRhZGFvLnB0\r\n
        L3B1YmxpY28vbHJjL2NjX3N1Yi1lY19jaWRhZGFvX2NtZF9jcmwwMDAyX3AwMDAxLmNybDAdBgNV\r\n
        HQ4EFgQUXeCSWShyv911UdFbuCCjy+CukIIwDgYDVR0PAQH/BAQDAgZAMA0GCSqGSIb3DQEBCwUA\r\n
        A4ICAQBcxGOl39XGMd0TPnRZJetZ7CW9WkWZATyUY0esLWIcfTyJAReo0hPMMb0BvpcchUPaVUu+\r\n
        JTtXshb1BzarcaoI6t+/wgx5d3brV/9kfqnP3FHrQXX87uTdxeOEjEXuOoM16kg5YkASrP3m230u\r\n
        Hrlpv8m2+nsqM371q1EboyPgVC0sWJji4axa1E8Yo+F92LW+RNBmEEamsukGb5s7MG3o6aQ06jNB\r\n
        CN73D4j0osxgegkx/3Ks5WySwcHMlndi7iMvUMMboTA0HOgvSbliyUWTaH0MiBO7kNiRTnnbmf6p\r\n
        fkPnFjZN6n8dZ5OTUcRcSzRdGfgZBb5M/EP2fbx5OgxEqNFgudNkFAx8vSUPSzX30eVZ20vRKjuU\r\n
        dDSNQmCgUBdDBMXgU8n9RR5ZaO6CMhwTCIJmvofUDRKmgSj5u54QNEepCZFuZ9jZuEWLR5qV2LrY\r\n
        5TsQBEP9hqQA2cSbt93Cqn5BQLTsJHkO8RrwsAdWGRb+WzK4iO9q5opPzd2GX0hp9cp/qN/Ymucu\r\n
        mwury+1nu/abdaKtNVjgDUvJvDkLHqaglr1o7XBD2HIDUAvl2rWAq4MWXzX57p3uoFxQTfNVM4Kj\r\n
        aasnG0YOB+Ff1MmDg0OZsh9h4BlV5bn/qAYEcZNbr/epMVBbWR2PM8K71+cC6N/elSav/uwKyztp\r\n
        Y6mB6A==\r\n
        -----END CERTIFICATE-----”
    },
    ”authMode”: ”implicit”,
    ”SCAL”: ”2”,
    ”multisign”: ”10”
}
```

C.4 credentials/authorize

Listagem C.6: Pedido credentials/authorize

```
{
      ”credentialID”: ”0a629801-4e16-4d8b-9465-131080ebf19b”,
      ”clientData”: [
            ”Padesdsstester1”,
            ”Padesdsstester2”
      ],
      ”hash”: [
            ”MDEwDQYJYIZIAWUDBAIBBQAEIOoBVbNqSgp5FqjT9VknAp9SAaiCY3lL9JyhHiLAtlwp”,
            ”MDEwDQYJYIZIAWUDBAIBBQAEIOoBVbNqSgp5FqjT9VknAp9SAaiCY3lL9JyhHiLAtlwp”
      ],
      ”numSignatures”: 2
}
```

Listagem C.7: Resposta credentials/authorize

```
{
   "SAD": "166381cd-8a4c-43aa-8eea-ffe59e23a89b",
   "expiresIn": "300"
}
```

C.5 signatures/signhash

Listagem C.8: Pedido signatures/signhash

```
{
   "credentialID": "0a629801-4e16-4d8b-9465-131080ebf19b",
   "SAD": "166381cd-8a4c-43aa-8eea-ffe59e23a89b",
   "code": "172067",
   "hash": [
            "MDEwDQYJYIZIAWUDBAIBBQAEIOoBVbNqSgp5FqjT9VknAp9SAaiCY3lL9JyhHiLAtlwp",
            "MDEwDQYJYIZIAWUDBAIBBQAEIOoBVbNqSgp5FqjT9VknAp9SAaiCY3lL9JyhHiLAtlwp"
      ]
}
```

Listagem C.9: Resposta signatures/signhash

```
{
   "signatures": [
      {
         "Hash": "GDEJIW6eIPwJPeXJMpZFGxcOG8skRW4UxRsdbsZnvIdh0xLRJRtE51SGr+Y3wCTv+/
            TWi0fWh234qXDliKIVg4DTG9ewHLR27eO0I2OOPjJMAjRyR4x0YF/Wot5t1OyZyRRPSXT/GuCSe
            miZxJFkqLJpWMA5+Oz42L247GjSCUhtguh+s+q1tOTEHO7UhWFysYjJQ6wLW8Gea3nt+Safuygi
            nFk6KG3u9DvGcmMkF5iE4IW9Htv9QPjEShrqIkrmYESyAhx2qBb9a9UGWn4EmiwJnBEyxd2H5zP
            EkN95O2yHZEt6heICssn9fY3tYMUxPqHu28bjM4ZH5g0gWY8qoTzvAelmzNLy0ovDBs2UX5mGN1
            PAC8nu4zW5fZ75IadNuHN+Y4fsIOzSqFNOtGWi2y+2qK2mMvgYfRFRH5JqpfxIk9wOUASmow7xO
            h6nGIWnI8JItGw+9XPg2+XqkUunQhhlOB75FoaqSsdeafBNIq4YOGUVUVTWbF6JjoUzw6n1",
         "name": "Padesdsstester1",
         "id": "13564455"
      },
      {
         "Hash": "GDEJIW6eIPwJPeXJMpZFGxcOG8skRW4UxRsdbsZnvIdh0xLRJRtE51SGr+Y3wCTv+/
            TWi0fWh234qXDliKIVg4DTG9ewHLR27eO0I2OOPjJMAjRyR4x0YF/Wot5t1OyZyRRPSXT/GuCSe
            miZxJFkqLJpWMA5+Oz42L247GjSCUhtguh+s+q1tOTEHO7UhWFysYjJQ6wLW8Gea3nt+Safuygi
            nFk6KG3u9DvGcmMkF5iE4IW9Htv9QPjEShrqIkrmYESyAhx2qBb9a9UGWn4EmiwJnBEyxd2H5zP
            EkN95O2yHZEt6heICssn9fY3tYMUxPqHu28bjM4ZH5g0gWY8qoTzvAelmzNLy0ovDBs2UX5mGN1
            PAC8nu4zW5fZ75IadNuHN+Y4fsIOzSqFNOtGWi2y+2qK2mMvgYfRFRH5JqpfxIk9wOUASmow7xO
            h6nGIWnI8JItGw+9XPg2+XqkUunQhhlOB75FoaqSsdeafBNIq4YOGUVUVTWbF6JjoUzw6n1",
         "name": "Padesdsstester2",
         "id": "13564456"
```

```
      }
   ]
}
```

C.6 dssFiler2Sign

Listagem C.10: Pedido dssFiler2Sign

```
{
   "credentialID": "28fbb532-e241-46c6-8951-c6becaff8fd4",
   "signatureLevel": "pades_t",
   "docName":"PADESDSSTESTER",
   "file" : "document",
   "imageParameters": {
      "image": "image",
      "xAxis": "200",
      "yAxis": "600",
      "width": "400",
      "height": "80"
   },
   "permission": "CHANGES_PERMITTED"
}
```

Listagem C.11: Resposta dssFiler2Sign

```
{
   "SAD": "4906dc36-9be5-412b-b8c2-56c4bd6532f8",
   "expiresIn": "300"
}
```

C.7 dssFiler2SendOTP

Listagem C.12: Pedido dssFiler2SendOTP

```
{
   "credentialID": "28fbb532-e241-46c6-8951-c6becaff8fd4",
   "otp": "681401",
   "SAD":"4906dc36-9be5-412b-b8c2-56c4bd6532f8"
}
```

Listagem C.13: Pedido dssFiler2SendOTP

```
{
   "signatures": "document"
```

```
3 }
```