UNIVERSIDADE DE LISBOA
FACULDADE DE CIÊNCIAS
DEPARTAMENTO INFORMÁTICA

ASSINATURAS DIGITAIS EM APLICAÇÕES LOW-CODE COM RECURSO A FLUXOS DE TRABALHO

Mafalda Pinheiro Luz

Mestrado em Engenharia Informática
Especialização em Sistemas de Informação

Trabalho de Projeto orientado por:
Prof. Doutora Maria Dulce Pedroso Domingos
Prof. Doutor Carlos Nuno Da Cruz Ribeiro

2021

Agradecimentos

Primeiramente, quero agradecer aos meus orientadores a Professora Dulce Domingos e o Professor Carlos Ribeiro por todo o apoio e disponibilidade ao longo do projeto. Quero agradecer à Reitoria da Universidade de Lisboa pela oportunidade que me deu para realizar a minha tese de mestrado.

A todo o Departamento de Informática da Reitoria, nomeadamente aos colegas que participaram no teste de usabilidade e preencheram o questionário, André Filipe, Andreia Rainha, Catarina Silva, Daniela Mendes, Fábio Ferreira, Nuno Heitor, Pedro Gonçalves, Pedro Moita, Tânia Castelo e Tânia Crespo. Aos restantes colegas do departamento Ana Rute, Daniel Vitoriano, José Lima, Ricardo Jesus e Ricardo Rito. Aos meus colegas bolseiros que me acompanharam e ajudaram durante esta etapa Ana Espinheira, Diana Marques, Diogo Godinho, Francisco Caeiro e Pedro Ladino.

Quero agradecer a toda a minha família por estarem sempre presentes mas especialmente aos meus pais, Fátima Pinheiro e Pedro Luz, à minha irmã Mónica Pinheiro Luz e ao meu namorado Renato Mendonça por nunca me deixarem desistir, por me motivarem para conseguir atingir todos os meus objetivos e por me fazerem sempre sentir muito amada.

Aos meus amigos mais queridos, que me viram crescer e que sei que passe o tempo que passar vão estar sempre presentes, Soraia Fernandes, Yara Lobo, Siobhan Fernandes, Miguel Octávio, João Veiga, Diogo Assunção e Yuriy Kozak.

Por fim, eu quero agradecer, a todos os que tornaram esta vida académica tão memorável. Obrigada a todos a todos os 15/16 e aos mais velhos os 13/14 que de uma forma ou outra me ajudaram a perceber que ”União, Humildade e Inteligência” são os verdadeiros 3 princípios. Desta legião de pessoas quero agradecer especialmente aos que me fizeram sentir parte de uma família no D.I. Brandi, Madrinha, Padrinho, Paella, Queixosa, EstáOn, Firewall, Maias, DeRastos, Ti’jolo, Nagini e aos meus pequeninos Arcanjo, Imortal, NadaSei e Ratér.

A todos um grande obrigada!

"Whatever you do in this life, it's not legendary, unless your friends are there to see it."

- Barney Stinson, How I Met Your Mother

Resumo

O desenvolvimento de aplicações *low-code* permite a criação de aplicações móveis e web por parte de utilizadores com variados níveis de experiência em programação. Esta forma de desenvolvimento veio responder, de forma ágil, à crescente necessidade de criação rápida de aplicações. Um bom exemplo deste tipo de ferramentas é o Joget Workflow, correntemente conhecido apenas por Joget. Esta plataforma tem vindo a ser utilizada pelos serviços centrais da reitoria da Universidade de Lisboa.

Atualmente, muitas das aplicações desenvolvidas têm uma componente para o utilizador inserir documentos que devem ser previamente assinados. Existem diversas formas de o fazer, sendo a mais comum a assinatura manual. Esta forma de assinaturas implica não só um trabalho acrescido como um gasto desnecessário de papel, visto que é necessário imprimir o documento, assinar, digitalizá-lo para depois fazer o *upload* do mesmo. Em alternativa temos as assinaturas digitais. Estas podem ser feitas de diversas formas, sendo que o seu grau de confiança está dependente da autoridade certificadora.

As assinaturas digitais com cartão de cidadão são uma opção muito conhecida devido ao seu elevado grau de confiança, porém continuam a não ser muito utilizadas por ser necessário um leitor para estes cartões. De forma a resolver este problema, surgiram as assinaturas com Chave Móvel Digital (CMD). Estas assinaturas são certificadas pela mesma autoridade que as anteriores, tendo exatamente a mesma validade legal, sem a desvantagem de ser necessário o uso de um leitor de cartões.

Este trabalho teve como objetivo desenvolver um novo artefacto para a plataforma Joget tornando possível a assinatura digital de documentos. Pragmaticamente foi escolhida a CMD para fazer estas assinaturas. A solução desenvolvida permite ao utilizador escolher o documento que pretende assinar e visualizá-lo. É ainda possível escolher se pretende uma assinatura visível ou invisível e indicar o local e motivo da assinatura. Uma vez que esteja assinado, o documento é guardado junto ao documento original e o utilizador pode visualizá-lo. Este novo artefacto pode integrar não só todas as aplicações já criadas como as que serão posteriormente desenvolvidas.

Palavras-chave: Joget; *Low-Code*; Assinaturas Digitais; Chave Móvel Digital; Artefacto;

Abstract

Low code development platforms allow users with various levels of programming experience to develop web and mobile applications. This way of development came as a response to the lack of programmers for the number of applications that need to be developed. A good example of this kind of platform is Joget Workflow. This platform has been used by the central services of the University of Lisbon.

Nowadays, many applications have a component for the user to upload previously signed documents. There are a lot of ways for doing it but the most common one is the manual signature. This way of signing implies not only an unnecessary workload but also a waste of paper since the user has to print the paper, sign it, scan it and only then upload it. In contrast to manual signatures, we have digital signatures. This type of signatures can be done in a lot of ways but the level of trust is associated with the certification authority.

Digital signatures made with the citizen card are a well-known option in Portugal due to their high degree of trust, but they are still not widely used since a card reader is required. In order to solve this problem, signatures with the Chave Móvel Digital (CMD) emerged. Chave Móvel Digital (CMD) is translated as Digital Mobile Key. This type of signatures is certificated by the same authority as the ones with the citizen card without the disadvantage of needing a card reader.

The focus of this work was to develop a new artifact for the Joget platform making possible the digital signature of documents. Pragmatically, it was chosen the CMD to do the signatures. The developed solution allows the user to choose the document that is going to be signed and then visualize it. It is even possible to choose if the signature is going to be visible or not, the location and reason of the signature. Once it is signed, the document is saved in the same location as the original and the user can visualize the result. This new artifact will be able to integrate not only every application already developed using Joget as well as those that will be developed in the future.

Keywords: Joget; Low-code; Digital Signatures; Artifact;Chave Móvel Digital;

Conteúdo

Lista de Figuras

Lista de Tabelas

Índice de Listagens

Capítulo 1

Introdução

Descreve-se ao longo deste documento o projeto de Assinaturas digitais em aplicações *low-code* com recurso a fluxos de trabalho. Este projeto foi desenvolvido na Reitoria da Universidade de Lisboa no âmbito da disciplina de Dissertação/Projeto em Engenharia Informática do Mestrado de Engenharia Informática da Faculdade de Ciências da Universidade de Lisboa.

No presente capítulo são apresentadas a motivação, os objetivos, a contribuição do projeto, assim como a estrutura do documento.

1.1 Motivação

Nos dias que correm, cada vez mais, existe uma necessidade de criar aplicações web e móveis. Porém, não existem programadores suficientes para responder às necessidades. Portanto, com o intuito de responder a este problema, surgiram as plataformas de desenvolvimento de aplicações *low-code*. As plataformas *low-code* funcionam principalmente com *drag and drop*, ou seja, o utilizador escolhe o artefacto que quer utilizar e arrasta-o para o sítio pretendido. Estes artefactos têm uma funcionalidade específica que pode ser configurada intuitivamente pelos utilizadores. Isto permite a personalização de cada aplicação. Desta forma, pessoas sem experiência em programação conseguem desenvolver aplicações e permite a pessoas com alguma experiência desenvolvê-las com maior facilidade, reduzindo assim o esforço associado a esta tarefa. Um bom exemplo deste tipo de plataformas é o Joget Workflow que tem vindo a ser utilizado pela reitoria da Universidade de Lisboa.

Considerada a maior de Portugal, a Universidade de Lisboa é constituída por 18 escolas e possui cerca de 50 mil alunos inscritos [16]. Os serviços centrais da Universidade de Lisboa organizam, coordenam e auxiliam todas as suas unidades orgânicas. Sendo uma instituição de grande dimensão existe a necessidade de criação de diversas aplicações web e móveis com os mais variados intuitos. De forma a dar resposta a esta necessidade a reitoria da Universidade de Lisboa começou a utilizar a plataforma Joget.

Passam pelos serviços centrais da Universidade de Lisboa inúmeros documentos que precisam de ser assinados. Tipicamente quando numa aplicação existe a necessidade de obter um documento assinado, o utilizador imprime o documento, assina-o manualmente, digitaliza-o e por fim faz o *upload* do documento para a aplicação. Com este processo as assinaturas não são legalmente válidas e ainda levanta alguns problemas nomeadamente, o tempo despendido na realização do mesmo. Além disso existe um

gasto desnecessário de recursos como o papel, os tinteiros, entre outros. Justifica-se assim a necessidade de criar um artefacto que permita tornar este processo ágil.

1.2 Objetivos

Como referido anteriormente, devido à grandeza da Universidade de Lisboa, pelos serviços centrais da Universidade de Lisboa passam inúmeros documentos. Como tal existem muitos processos em que os documentos precisam de ser assinados.

O objetivo principal deste projeto é criar um artefacto para o Joget que permita assinar documentos digitalmente no contexto de um fluxo de trabalho, agilizando assim este processo nas aplicações desenvolvidas pela reitoria da Universidade de Lisboa. Este artefacto deve poder ser integrado tanto em aplicações já desenvolvidas como em aplicações que serão criadas no futuro.

Atualmente, sempre que é preciso obter um documento assinado, esta ação tem de ser feita fora da aplicação, sendo que a forma mais comum de assinar um documento é manualmente e tem associada os problemas descritos anteriormente que resultam num desperdício de recursos. Em alternativa à assinatura manual existem alguns tipos de assinatura digital, porém não são muito utilizados por habitualmente ser necessário o uso de *hardware* ou *software* adicional. Um dos requisitos deste projeto é desenvolver um artefacto que não necessite de *hardware* ou *software* adicional e seja assim uma solução sem custos e de fácil utilização por parte de todos os utilizadores.

Por se tratar de assinaturas e a falsificação destas ser um problema que pode ter consequências graves, um fator muito importante é também a segurança tanto dos dados dos utilizadores como da assinatura em si.

Por todos estes motivos, o objetivo principal deste projeto é desenvolver um artefacto que produza assinaturas de forma segura e que seja de fácil utilização.

1.3 Contribuição

Neste momento, existem diversas formas de assinar digitalmente sendo que o seu grau de confiança está dependente da autoridade certificadora. Uma opção muito conhecida, devido ao seu elevado grau de confiança, são as assinaturas digitais com cartão de cidadão. Neste caso o utilizador precisa de ter instalado um software próprio e possuir um leitor de cartões. De forma a simplificar este processo, a mesma autoridade certificadora desenvolveu a Chave Móvel Digital (CMD) que associa o Número de Identificação Civil (NIC) ao número de telemóvel do cidadão. Dado isto, o cidadão consegue obter o mesmo resultado utilizando apenas o seu número de telemóvel, um *pin* definido por si e um código único que recebe no telemóvel.

Tendo em conta que esta forma de assinar digitalmente cumpre todos os requisitos descritos anteriormente, optou-se por implementar esta solução no artefacto criado. Com esta ferramenta a assinatura passa apenas pela escolha ou *upload* do documento e pelo preenchimento dos dados associados à CMD. Este artefacto, na plataforma Joget, permite que todas as aplicações já criadas ou que serão posteriormente criadas através desta plataforma usufruam desta ferramenta. De forma a atingir esta meta foi feita uma análise dos requisitos funcionais e não funcionais pretendidos de modo a planear a solução. Para tal

foi também estudada a arquitetura da CMD, dos artefactos do Joget e dos documentos Portable Document Format (PDF). Após a análise de requisitos e o estudo das ferramentas concluído, fez-se um estudo sobre o desenho e a arquitetura do artefacto de forma a que este cumpra todos os requisitos.

Concluído o planeamento passou-se à implementação da solução. Esta implementação divide-se em duas componentes cliente e servidor. No cliente encontra-se tudo o que envolva interação com o utilizador, são recolhidos os dados do utilizador relativos à CMD e enviados para o servidor. No servidor são processados os dados vindos do cliente, é implementada a comunicação com a AMA e os seus métodos para obtermos uma assinatura. São ainda implementados métodos para manipular ficheiros PDF.

Por fim, fez-se uma prova de conceito implementando a solução na aplicação SIADAP3 e pediu-se a funcionários da reitoria que preenchessem um questionário de forma a perceber o resultado obtido na escala SUS que avalia a usabilidade.

1.4 Estrutura do documento

Este documento tem 6 capítulos e segue a seguinte estrutura:

- **Capítulo 1 - Introdução**
  Apresenta a motivação, os objetivo do trabalho e a contribuição no mesmo . É ainda apresentada a estrutura do documento.

- **Capítulo 2 - Contexto**
  Explica os conceitos base subjacentes ao projeto.

- **Capítulo 3 - Análise e desenho**
  Apresenta os resultados da análise de requisitos e descreve a arquitectura e o desenho da solução.

- **Capítulo 4 - Implementação**
  Explica como foi feita a implementação do artefacto no Joget.

- **Capítulo 5 - Prova de conceito e validação**
  Descreve a forma como foi feita a prova de conceito que consistiu na implementação do artefacto na aplicação SIADAP3. Apresenta ainda a validação feita com o questionário SUS.

- **Capítulo 6 - Conclusões e trabalho futuro**
  Resume, descreve e conclui o trabalho realizado e apresenta o trabalho futuro.

Capítulo 2

Contexto

Este capítulo tem como propósito contextualizar o leitor sobre os tópicos abordados no projeto. Apresentam-se as plataformas *low-code*, explicando a sua utilidade e o modo de funcionamento, fazendo especial referência ao Joget Workflow e aos seus artefactos. Detalham-se ainda os conceitos de assinatura digital e Chave Móvel Digital (CMD), assim como o seu funcionamento conjunto. É ainda feita uma explicação sobre o formato PDF, a sua estrutura em geral e os aspetos mais relevantes para as assinatura digitais.

2.1 Plataformas *low-code*

As plataformas *low-code* são plataformas que permitem o desenvolvimento de aplicações web e móveis de forma rápida e ágil. São plataformas com uma grande componente visual, cujo desenvolvimento é realizado principalmente com ”drag and drop”.

Este tipo de plataformas contém um conjunto de funcionalidades genéricas que podem ser personalizadas pelo utilizador. Note-se que o utilizador deste tipo de programas não é o típico utilizador final das aplicações, mas sim um ”middleman”, que utiliza software com diversos artefactos desenvolvidos por um programador de forma a criar aplicações para o utilizador final.

A grande vantagem destas plataformas é permitir que tanto indivíduos que não sabem programar como programadores experientes consigam criar aplicações, fazendo com que o processo para os programadores experientes se torne muito mais rápido.

Um exemplo deste tipo de plataformas é o Joget que é a ferramenta utilizada na reitoria da Universidade de Lisboa.

2.1.1 Joget Workflow

O Joget Workflow é uma plataforma *low-code* que permite a criação de aplicações específicas dedicadas às necessidades de cada utilizador. A plataforma permite que utilizadores com ou sem experiência em programação consigam criar aplicações. Este sistema, por ter uma utilização simples, permite ao utilizador focar-se mais nas necessidades de negócio, uma vez que não tem de se preocupar com as questões mais técnicas relacionadas com o desenvolvimento. Desta forma, aumenta-se a eficiência e produtividade, reduzindo consequentemente os custos de produção.

Visto que o Joget é uma ferramenta que se destina a criar aplicações orientadas a processos, para desenvolver uma aplicação, o primeiro passo deve ser desenhar o fluxo de processos. O desenho é feito na *Process Builder* como se pode observar na Figura 2.1.

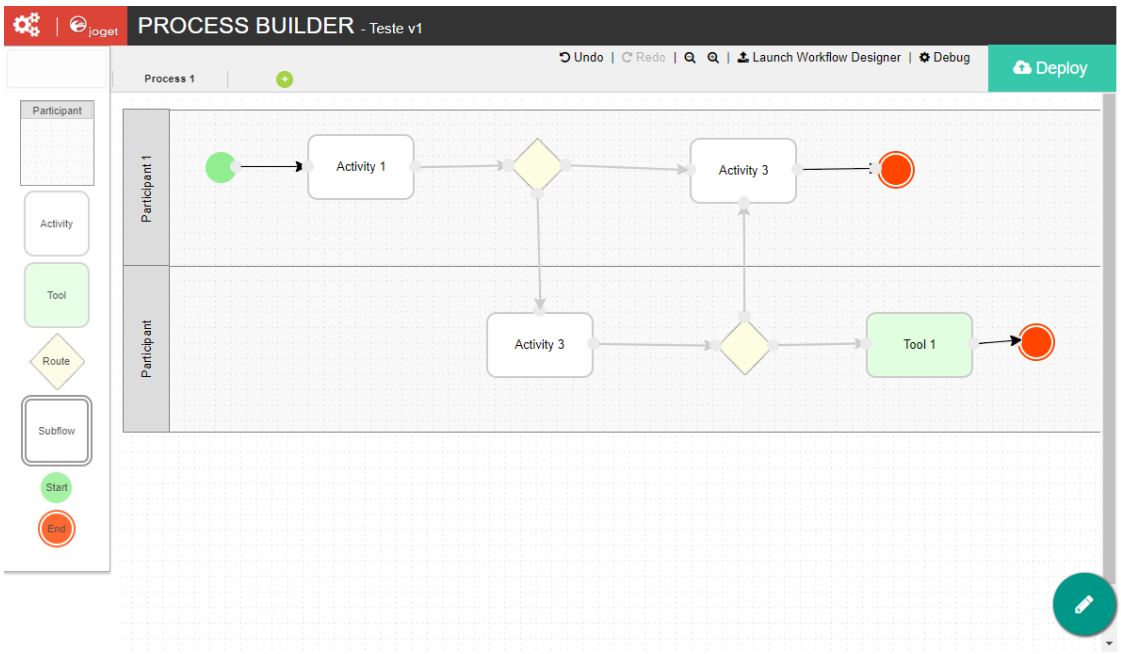

Figura 2.1: Ecrã para a criação de um fluxo de processos

Depois de termos o *workflow* desenhado, o segundo passo é criar formulários de forma a recolher os dados que o utilizador tem de inserir para completar o processo. Os formulários são criados no *Form Builder*, composto por um conjunto de artefactos, aos quais correntemente chamamos *plugins*. Para preencher o formulário escolhemos o *plugin* que queremos utilizar e arrastamo-lo para a secção que se encontra no centro da Figura 2.2.

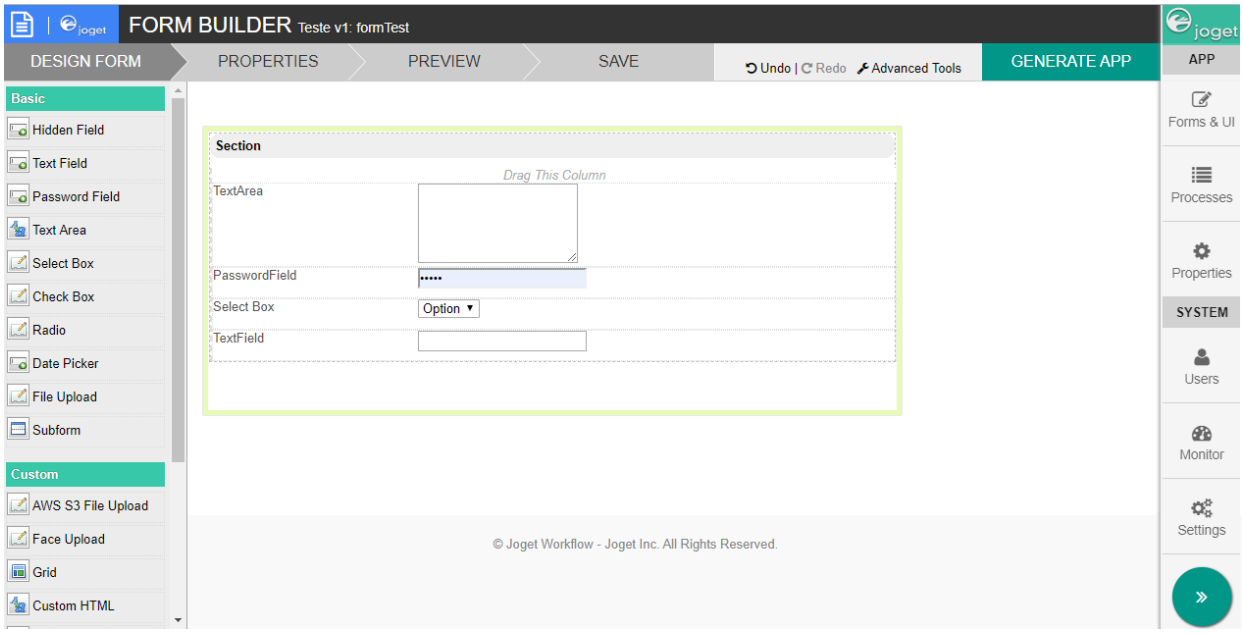

Figura 2.2: Ecrã para a criação de um formulário

De forma a concluir a criação da aplicação, uma vez que tenhamos todos os formulários e *workflows* criados, devemos ir ao *Userview Builder* para personalizar o desenho da aplicação, como ilustrado na Figura 2.3. Uma vez que esta tarefa esteja completa, ficamos com a aplicação criada.

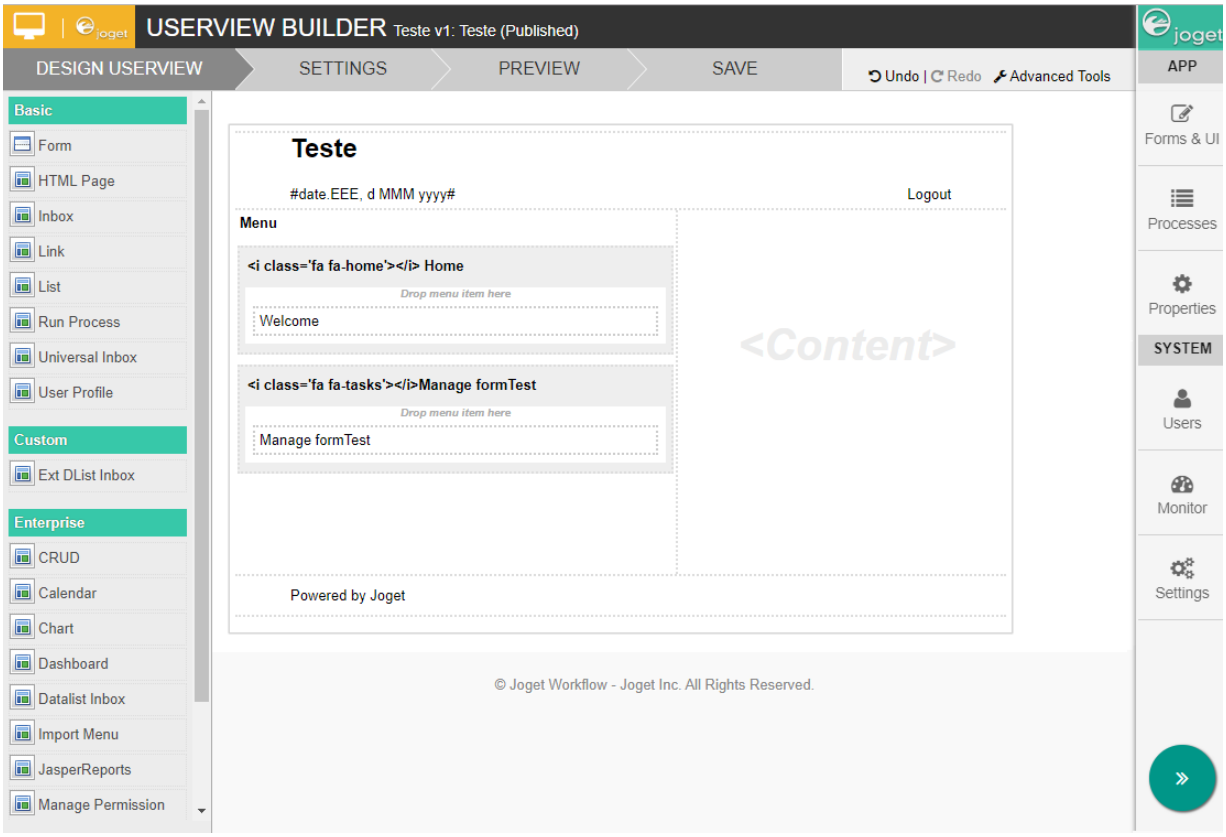

Figura 2.3: Ecrã para personalizar o *design* da aplicação

2.1.2 Arquitetura dos *plugins* no Joget Workflow

O Joget é um sistema composto por artefactos, denominados de *plugins*. Estes *plugins* estendem o sistema e permitem a criação de aplicações direcionadas ao âmbito do negócio. Existem vários tipos de *plugins*, como se pode observar na Figura 2.4. A plataforma permite que sejam criados e adicionados novos *plugins* desde que se enquadrem numa destas categorias.

Pode-se observar os diferentes plugins nas Figuras anteriores, como por exemplo na barra lateral da Figura 2.2 encontramos plugins do tipo *Form Field Element* e no centro da Figura 2.1 encontramos plugins do tipo *Userview Menu*.

Existem dois tipos de arquitecturas sobre as quais podem ser desenvolvidos estes *plugins*, são elas a ***Standard Java Plugin*** e a ***Dynamic OSGi Plugin***. A escolha da arquitetura a utilizar no desenvolvimento deve ter em conta as características de cada *plugin*, porém quando se utilizam muitas bibliotecas diferentes das bibliotecas base do Joget é usual utilizar-se a arquitetura *Dynamic OSGi Plugin*. A Tabela 2.1 apresenta as principais diferenças entre os dois tipos de arquitetura.

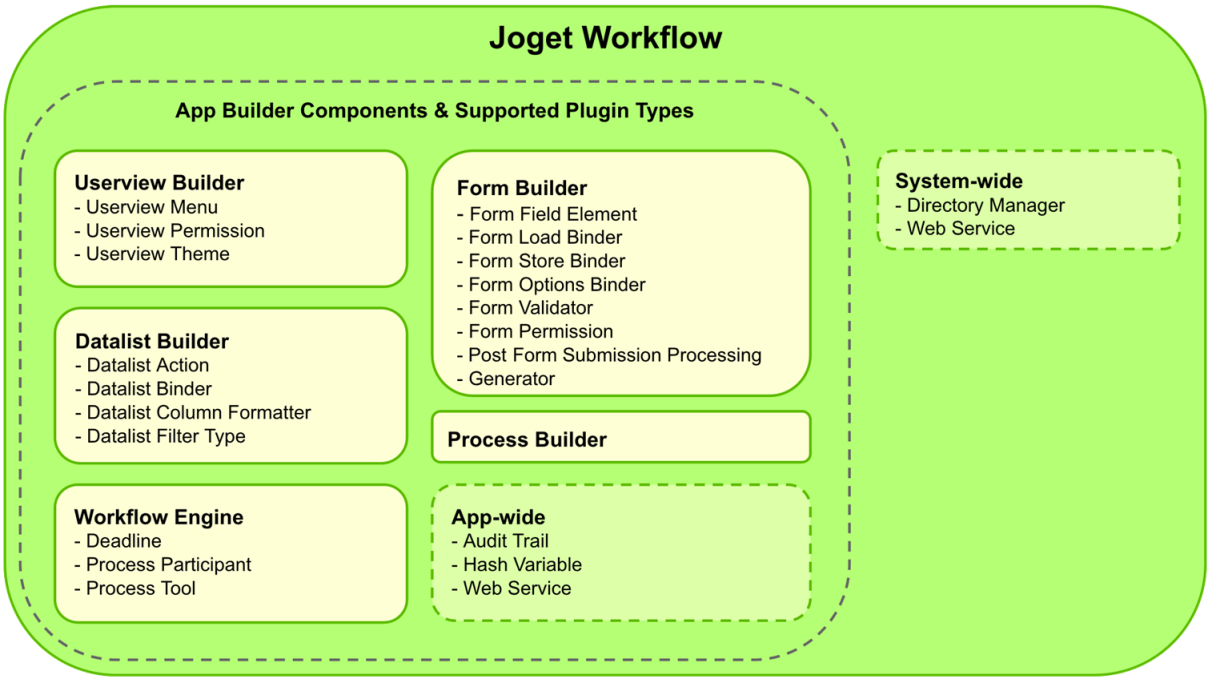

Figura 2.4: Tipos de Plugins
Fonte: Joget Community - Knowledge Base for v6 [12]

Tabela 2.1: Diferenças entre os dois tipos de arquiteturas de *plugins* no Joget

Standard Java JAR bundle	**Dynamic OSGi Plugin**
O JAR tem de ser disponibilizado no *classpath* do Java	O JAR tem de ser implementado através da ferramenta *Manage Plugins*
Esta arquitetura facilita o desenvolvimento e a fase de teste quando se utilizam classes e bibliotecas Java normais	Esta arquitetura dificulta um pouco mais o desenvolvimento e a fase de testes devido a sua configuração OSGi e isolamento
Os artefactos com esta arquitetura podem entrar em conflito com as bibliotecas base do Joget ou com outros *plugins*	Os artefactos correm em modo isolado o que previne conflitos com as bibliotecas base do Joget e com outros *plugins*
Para se implementar ou fazer alterações neste tipo de artefactos é necessário recomeçar a *Java Virtual Machine (JVM)*	Suporta um carregamento/ descarregamento/ recarregamento dinâmico de artefactos sem ter de recomeçar

2.2 Assinatura Digital

Uma assinatura digital, tal como o nome indica, serve para assinar documentos digitalmente.
As garantias que estas assinaturas [11] [15] nos dão são:

- Autenticação - Autenticam quem produziu o documento, permitindo assim que quem lhe aceda

possa confiar na fonte.

- Não repúdio - Quem assinou o documento não pode negar que o assinou, permitindo que desta forma se consiga resolver eventuais disputas.
- Integridade - Permite confirmar que o documento não foi modificado.

Atualmente, quando se falam em assinaturas digitais, existe uma tendência de se confundir assinaturas digitais com assinaturas eletrónicas, porém não são sinónimos. Pode-se utilizar o termo assinatura eletrónica (e-signature) para um qualquer processo eletrónico que demonstre que foi aceite um registo ou acordo [9]. As assinaturas eletrónicas não são obrigadas a utilizar normas de segurança, o que faz com que possam ser facilmente falsificadas e que seja mais difícil de provar quem realmente assinou o documento. Um exemplo de uma assinatura eletrónica são as assinaturas feitas manualmente com um tablet num documento PDF.

Sabemos a partir desta descrição que as assinaturas digitais fazem parte das assinaturas eletrónicas. No entanto, estas utilizam um identificador digital baseado em certificados que permitem validar a autenticidade do assinante.

As assinaturas digitais subdividem-se ainda em assinaturas qualificadas ou não qualificadas, sendo que as qualificadas são as únicas que têm valor legal. As assinaturas qualificadas têm de obedecer a um conjunto de requisitos, sendo um desses requisitos por exemplo é a assinatura ter que ser feita com um *token* e esse *token* tem de ser emitido com verificação física. Um exemplo deste tipo de assinatura são as assinaturas feitas com o cartão de cidadão ou Chave Móvel Digital (CMD). Estas assinaturas, nos dias que correm, são legalmente aceites, como previsto no Decreto-Lei n.º 290-D/99 [5].

De seguida é apresentado um exemplo prático para clarificar como são feitas as assinaturas digitais que tem associado as Figuras 2.5, 2.6 e 2.7.

Supondo que existe um indivíduo 'A' que quer obter um documento assinado digitalmente, para tal o indivíduo 'A' deve gerar a chave privada e a chave pública. Caso o indivíduo 'A' queira partilhar o seu documento assinado com o indivíduo 'B' deve partilhar a sua chave pública, para que mais tarde este possa verificar a assinatura.

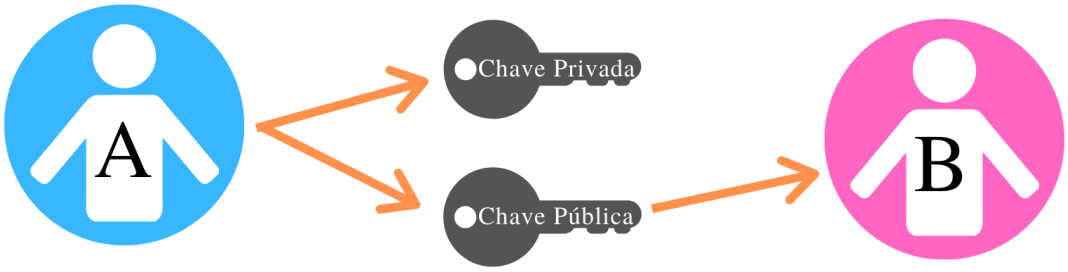

Figura 2.5: Geração das Chaves

Partindo do pressuposto que o indivíduo 'A' já gerou as suas duas chaves, de forma a criar a assinatura que este vai colocar no documento, são feitos os seguintes passos:

1. O indivíduo 'A' escolhe o documento que pretende assinar.
2. É utilizado um algoritmo de síntese sobre o documento.

3. Utiliza-se a chave privada do indivíduo 'A' para cifrar a síntese gerada no passo anterior e ficamos assim com a assinatura. Uma vez que tenhamos a assinatura podemos então juntá-la ao documento.

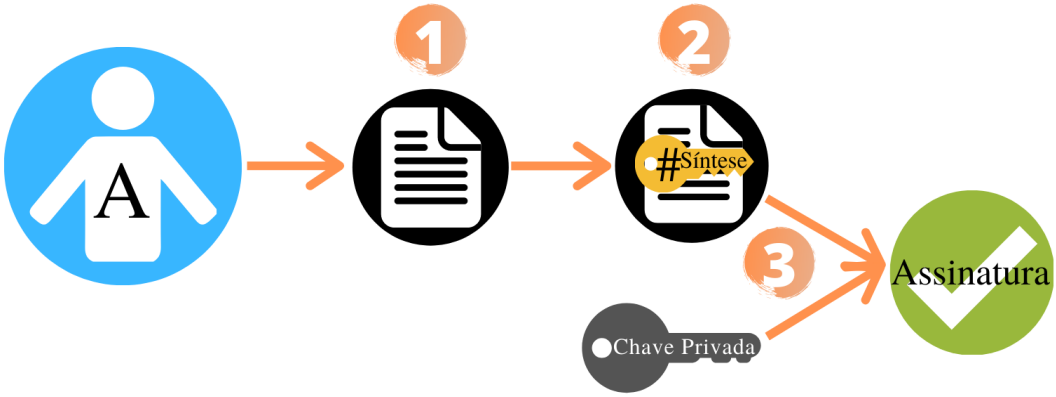

Figura 2.6: Processo de criação da assinatura digital em 3 passos

Admitindo que o indivíduo 'A' assinou um documento e enviou-o ao indivíduo 'B', de forma a confirmar se o documento não foi modificado e se foi indubitavelmente assinado pelo indivíduo 'A', este precisa que 'A' tenha partilhado a sua chave pública consigo. A verificação é feita em 3 passos, sendo que a ordem dos dois primeiros não é importante:

1. O indivíduo 'B', utilizando a chave pública de 'A', decifra a assinatura do documento ficando assim com a síntese do documento.

2. O indivíduo 'B', utiliza um algoritmo de síntese sobre o documento original. O algoritmo de síntese têm de ser o mesmo utilizado pelo indivíduo que assinou o documento, neste caso 'A'.

3. Por fim o indivíduo 'B' deve verificar se o resultado obtido nos passos anteriores é igual. Caso afirmativo o indivíduo 'B' sabe que a assinatura no documento é válida.

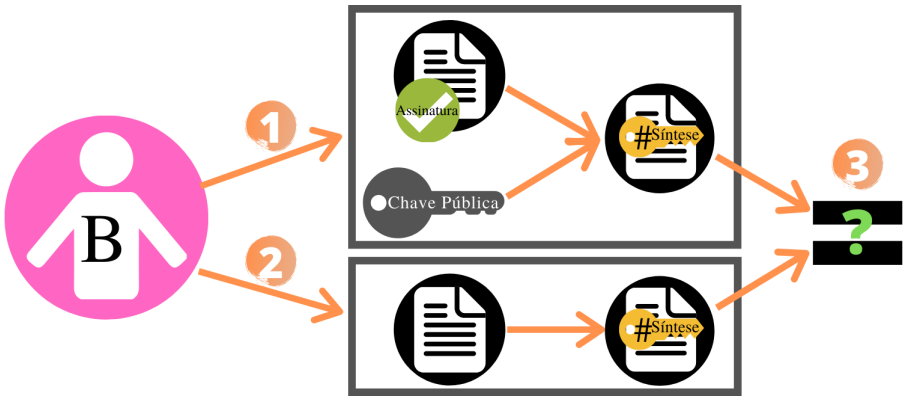

Figura 2.7: Processo de verificação da assinatura digital em 3 passos

A grande dificuldade em garantir a segurança das chaves públicas está relacionada com a forma como estas são distribuídas. Existem três formas convencionais de o fazer [15] :

- **Anúncio público da chave pública** - As chaves públicas são, como o nome indica, públicas e podem ser partilhadas com qualquer pessoa, pois os algoritmos de cifra tipicamente utilizados são conhecidos globalmente. O problema desta forma de distribuição é que não há forma de verificar quem é que está efetivamente a partilhar a chave. Desta forma qualquer pessoa pode partilhar uma chave pública fazendo-se passar por outra.

- **Repositório disponível ao público** - Um repositório à responsabilidade de uma entidade de confiança, que guarde e distribua as chaves públicas, garante uma maior segurança. Cada participante regista nestes repositórios uma chave pública que é guardada em conjunto com o nome do participante. Uma desvantagem desta solução é que cada vez que é necessário a chave pública é necessário aceder ao repositório.

- **Certificados da Chave pública** - Uma alternativa é os participantes utilizarem certificados para trocar a chave pública. Estes certificados são compostos pela chave pública, um identificador do dono da chave e um bloco assinado por quem emitiu o certificado, que normalmente são autoridades certificadoras. As autoridades certificadoras por norma são ou agências do governo ou instituições em que as pessoas em geral confiam. Estes certificados, quando são emitidos, vão assinados e os utilizadores usam a chave pública da autoridade certificadora para os verificarem.

Em relação às chaves privadas, existe uma grande preocupação em mantê-las privadas, para tal temos de ter em consideração principalmente como é feito o seu armazenamento e utilização. Uma solução muito comum é o uso de *USB tokens* ou *Smartcards*, como o cartão de cidadão ou passaporte eletrónico português para as armazenar. Quem possui estes dispositivos sabe que só os deve utilizar em ambientes seguros.

2.2.1 Chave Móvel Digital (CMD)

A Chave Móvel Digital (CMD) desenvolvida pela AMA é um sistema de autenticação eletrónica que complementa o cartão de cidadão. Este sistema funciona com a associação do Número de Identificação Civil (NIC) ou o número de passaporte a um número de telemóvel, um *pin* definido pelo cidadão e um código único que é enviado para o número de telemóvel associado. Este código único é gerado a cada utilização, tornado assim a utilização da CMD mais segura.

De forma a poder utilizar esta funcionalidade, o cidadão deve efetuar o registo da sua CMD. Este registo pode ser feito de três formas:

- Presencialmente, para cidadãos portadores de cartão de cidadão ou passaporte, nos espaços do cidadão ou espaços empresa. A forma com é feito este registo está esquematizado na Figura 2.8 e é descrita da seguinte forma: O utilizador deve dirigir-se a um dos espaços mencionados e fazer o pedido da CMD (A), estes pedidos são respondidos pelo *backoffice* da CMD (B), que faz a ligação ao servidor com *Hardware Security Module (HSM)* (C). Este tipo de servidor tipicamente guarda e gere as chaves digitais para uma autenticação segura. O servidor guarda os dados de forma segura numa base de dados (D). Com esta opção o cidadão fica com a CMD ativa no mesmo momento.

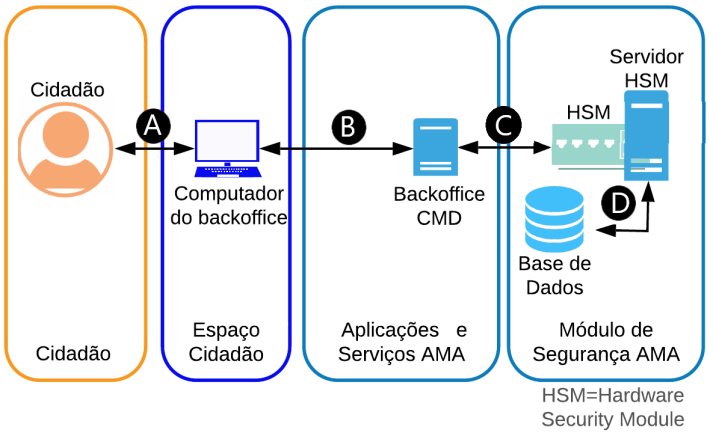

Figura 2.8: Obter a CMD presencialmente

- Online, através do portal "Autenticacao.gov.pt". Este método só pode ser utilizado por portadores de leitor de cartões, cartão de cidadão e código *pin* de autenticação. Com esta opção o cidadão fica com a CMD ativa no mesmo momento. A forma com é feito este registo está esquematizado na Figura 2.9 e é descrita da seguinte forma: O cidadão utiliza o seu computador pessoal (A) para através deste aceder ao portal da CMD (B) que faz a ligação ao servidor com *Hardware security module (HSM)* (C). O servidor guarda os dados na base de dados (D) e para completar o processo faz também um pedido de envio do código único ao utilizador pela *Gateway de SMS* (E). O cidadão recebe o código no seu telemóvel (F) e introduz no portal da CMD o código que recebeu (G,A) e o processo fica completo.

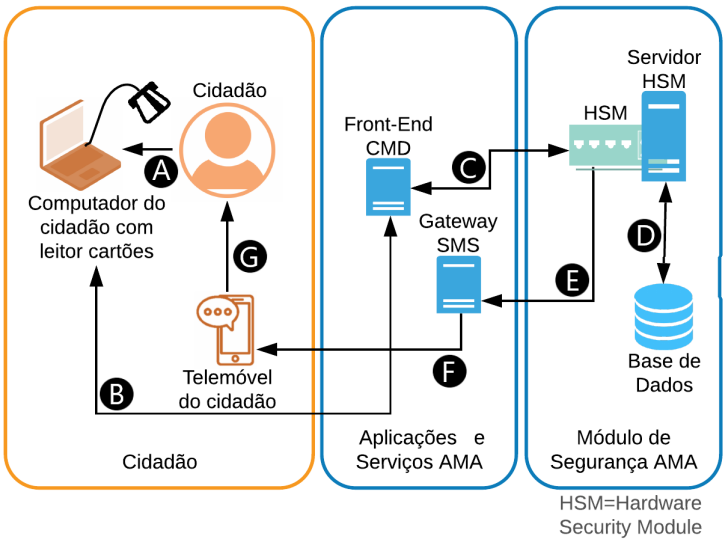

Figura 2.9: Obter a CMD online com o cartão de cidadão

- Online, através do portal das finanças. Para tal o cidadão deve possuir o seu número de identificação fiscal e a senha de acesso ao portal das finanças. A forma com é feito este registo está esquematizado na Figura 2.10 e é descrita da seguinte forma: O cidadão utiliza o seu computador pessoal (A) para aceder ao portal ”Autenticacao.gov.pt”(B), seleciona a opção para registar a CMD através do portal das finanças e é redirecionado para o mesmo (C). Depois de inserir as suas credenciais é enviado um email para confirmar o pedido de registo (D,E). Uma vez que o cidadão confirme através do link enviado, por email, que quer fazer o registo da CMD o portal das finanças comunica com o backoffice da CMD (F), que faz a ligação ao servidor com *Hardware security module (HSM)* (G), e guarda os dados na base de dados (H). De forma a completar o processo é enviada uma carta pela entidade responsável pelo portal das finanças ao cidadão (I). O cidadão terá de esperar 1 a 5 dias úteis até receber uma carta com um *pin* provisório (J). A partir deste momento, para ativar a CMD, o cidadão terá de através do seu computador (A) aceder ao portal ”Autenticacao.gov.pt”(B) e alterar o seu *pin* provisório por um à sua escolha. De modo a completar essa ação, faz o login com o seu número e *pin* provisório, recebe um código no telemóvel que insere no site para verificar a sua identidade (K,L,M,N). Uma vez autenticado, o cidadão deve proceder à alteração. Neste momento, a aplicação da AMA comunica com o servidor com *Hardware security module (HSM)* (k) para fazer a alteração na base de dados (H), ficando assim o processo de registo completo.

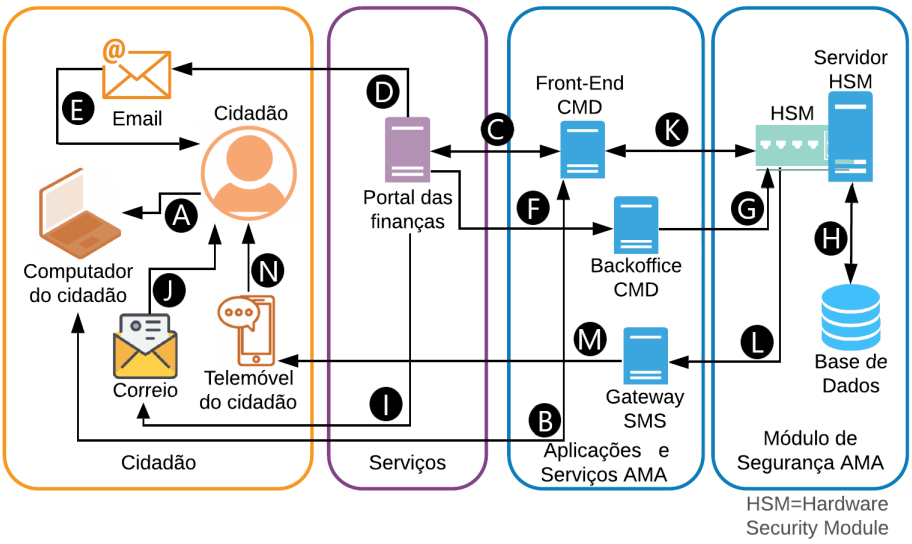

Figura 2.10: Obter a CMD online no portal das finanças

Uma vez que o cidadão tenha a CMD ativa, pode utilizá-la tanto para se autenticar em todos os sites que utilizem esta ferramenta como para assinar documentos digitalmente.

2.2.2 Assinaturas com a Chave Móvel Digital (CMD)

As assinaturas com a CMD surgiram para substituir as assinaturas com o cartão de cidadão por serem mais práticas, uma vez que a maior parte das pessoas não possui um leitor de cartões. Dada a dificuldade em garantir a segurança das chaves privadas descrita acima, a AMA optou por ter uma forma de criar as assinaturas digitais um pouco diferente da convencional.

As assinaturas com a CMD são reconhecidas tanto a nível nacional como europeu, constando na "Trusted List Browser" (`https://webgate.ec.europa.eu/tl-browser/#/tl/PT`). Para tal, têm de obedecer a diversas regras que constam no documento "POL#16 - Política CMD de assinatura qualificada"[3].

Apresenta-se agora um exemplo do funcionamento das assinaturas digitais com a CMD que tem associada a Figura 2.11. Para além deste exemplo, existem outras formas de colocar em funcionamento estas assinaturas. Uma variante deste processo, seria por exemplo, uma aplicação web que teria o *browser* como interveniente.

Supondo que existe um indivíduo 'A' que quer obter um documento assinado digitalmente e uma entidade 'B' que certifique a assinatura, os passos seguintes resumem sucintamente essa operação:

0. O indivíduo 'A' deve cifrar as suas credenciais com a chave pública de 'B' e enviá-las para a entidade 'B'
1. O indivíduo deve ainda escolher o documento e enviar a sua síntese juntamente com a credenciais cifradas.
2. A entidade 'B' envia um código único e temporário para o telemóvel do indivíduo 'A'.
3. O indivíduo 'A' envia o mesmo código à entidade 'B' de forma a que esta possa validar que é o próprio a fazer o pedido de assinatura.
4. A entidade 'B' deve autenticar o indivíduo 'A' e utilizar a chave privada deste para cifrar a síntese do documento que recebeu ficando assim com a assinatura.
5. Uma vez que a assinatura esteja feita, a entidade 'B' envia-a ao indivíduo 'A'. Este junta a assinatura com o documento e fica assim com o documento assinado.
6. Adicionalmente o indivíduo 'A' pode pedir o seu certificado à entidade 'B' para o inserir no documento facilitando assim a posterior validação da assinatura

A arquitetura das assinaturas feitas com a Chave Móvel Digital (CMD) é composta por 2 componentes:

- **Aplicações e serviços**
  Esta componente suporta a interação com o cidadão.
- **Segurança**
  Esta componente é a responsável por todas as operações criptográficas e pela gestão dos certificados. A assinatura digital obriga a que o certificado utilizado seja assinado por uma entidade de certificação reconhecida.

A comunicação com as componentes é feita através de HTTPS com autenticação e mensagens SOAP. O serviço de assinatura digital é composto por três operações principais:

- A operação ***GetCertificate*** que consiste na obtenção do certificado do cidadão que será utilizado para incluir no documento assinado.

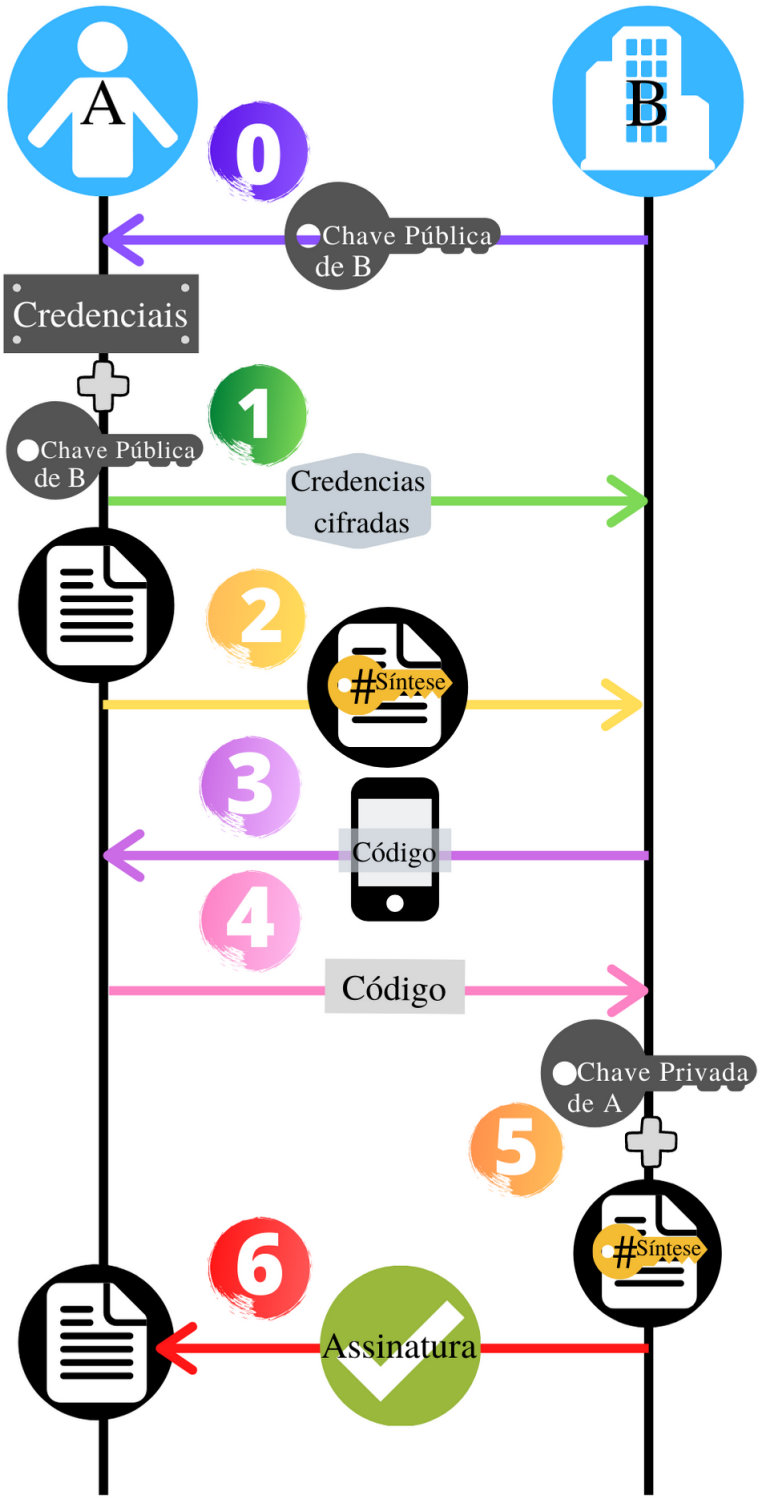

Figura 2.11: Processo de criação da assinatura digital com a CMD em 6 passos

- A operação ***SCMDSign***, que recebe a síntese do documento que se pretende assinar e faz o envio do Código *One Time Password (OTP)* para o telemóvel ou email do utilizador. Quando se quer assinar múltiplos documentos é necessário utilizar a operação ***SCMDlMultipleSign*** que é muito semelhante à anterior mas recebe uma lista de sínteses em vez de receber apenas uma.

- A operação ***ValidateOtp*** que valida o código enviado e devolve a síntese assinada ou a lista de sínteses assinadas.

2.3 Portable Document Format (PDF)

Os ficheiros PDF foram propostos pela Adobe Systems em 1992. A ideia do Dr. John Warnock, cofundador da Adobe, seria que este tipo de documentos preservassem sempre a aparência original, e esta característica faz com que este seja o tipo de documentos ideal para ser assinado digitalmente. Estes documentos atendem aos padrões ISO32000 [8] e têm uma estrutura muito rica, de forma a assegurar que mantêm a aparência.

2.3.1 Estrutura PDF

Os documentos PDF podem incluir texto, imagens, links web, etc. De uma forma geral estes documentos são compostos por 4 partes principais [10]. São elas:

- Cabeçalho - É a primeira linha do ficheiro e apresenta o número da versão.

- Corpo - Aqui é guardado o conteúdo do documento, ou seja, todos os objetos que o utilizador visualiza. Estes objetos são por exemplo, os textos, as imagens e os outros elementos de multimédia que possam existir.

- Tabela xref - Esta tabela, também conhecida por tabela cruzada, contém as referências para todos os objetos contidos no documento. A tabela é constituída por:

  - A *keyword* "xref"que se encontra na primeira linha.
  - Dois números, o primeiro representa o índice em que os objetos começam e o segundo representa a quantidade de entradas na tabela.
  - As restantes linhas são compostas por 3 partes:
    * O offset de 10 dígitos do objeto a começar no início do ficheiro.
    * Um número de geração de 5 dígitos. Este número serve para indicar qual a geração atual do objeto, cada vez que o objeto é apagado e reutilizado é dado um novo número de geração.
    * Uma *flag* com um "n" ou um "f" que representa se o objeto é utilizado ou não, respetivamente. Um objeto que seja apagado permanece no ficheiro com a flag "f".

Se algum byte for alterado, todos os apontadores ficarão errados e o ficheiro ficará corrompido.

- *Trailer* - Estão aqui as referências para a tabela cruzada (xref). É aqui que se especifica como é que a aplicação deve ler o documento PDF. Qualquer aplicação deve começar a ler um ficheiro PDF pelo fim. A última linha do ficheiro é "%%EOF".

É possível adicionar ou atualizar dados num ficheiro PDF. Para tal utiliza-se a atualização incremental [17]. Esta atualização é feita em três passos:

- É criada uma nova secção do corpo atualizada, depois do marcador "%%EOF". Os objetos novos ou atualizados são adicionados nesta zona do ficheiro.
- De seguida é criada uma nova secção de tabela xref que só contém referências para os objetos novos ou que tenham sido atualizados.
- Por fim, é adicionado um novo *trailer* que contém referências para todos os objetos, tanto os antigos como os novos e os atualizados.

Na Figura 2.12 pode-se observar as diferenças entre um ficheiro original e um ficheiro que tenha sofrido uma atualização incremental.

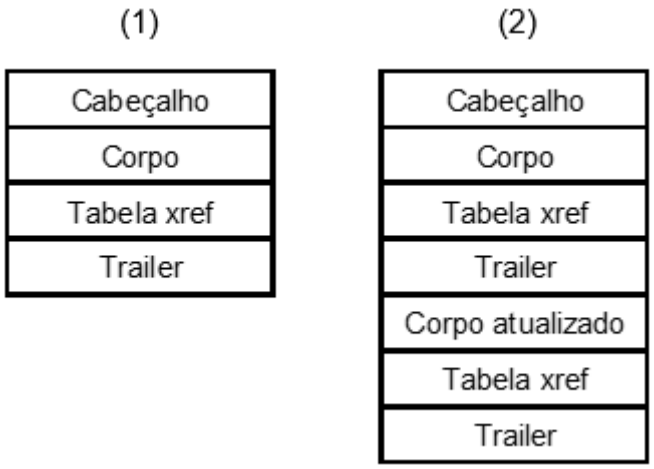

Figura 2.12: Estrutura de um ficheiro PDF no estado original(1) e depois de ser atualizado incrementalmente(2)

2.3.2 Assinaturas digitais em ficheiros PDF

Na secção anterior é mencionado que no corpo do documento estão representados os objetos. Existem bastantes tipos de objetos, porém nas assinaturas digitais o tipo mais importante são os dicionários. Na Figura 2.13 temos um dicionário de assinaturas que contém diversos elementos, sendo apenas obrigatório ter *Filter*, *Contents* e *ByteRange*. Cada um dos elementos deve conter dados diferentes [17]. São eles:

- ***Filter*** **-** especifica o *handler* prefencial que deve ser utilizado para verificar a assinatura.
- ***Contents*** **-** contém um *byte string* com a assinatura.

- ***ByteRange*** **-** especifica o *byte range* em que a síntese é calculada.
- ***SubFilter*** **-** especifica o *encoding* utilizado na assinatura. Pode conter também *handlers* alternativos para validar a assinatura.
- **M -** contém a data e hora a que a assinatura foi processada.

Existem ainda elementos em que é o assinante que preenche, tal como o seu nome (*Name*), a razão da assinatura (*Reason*), a informação de contacto (*ContactInfo*), a localização (*Location*), etc.

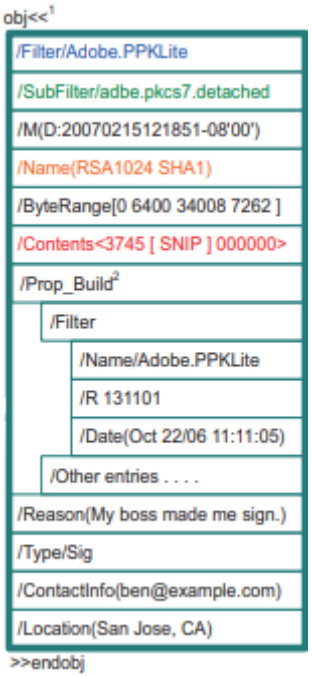

Figura 2.13: objeto Signature Dictionary
Fonte: Digital Signatures in a PDF [7]

Para assinar documentos digitalmente existem vários padrões [4], tais como:

- XAdES: Este padrão é recomendado para assinar documentos com o formato XML, permite múltiplas assinaturas e que estas sejam processadas automaticamente.
- PAdES: Este padrão é recomendado para assinar documentos PDF e permite que a assinatura seja visível no documento.
- CAdES: Este padrão, apesar de poder ser utilizado em documentos XML e PDF, é recomendado para os outros tipos de documentos que precisem de ser assinados. Permite múltiplas assinaturas.

2.4 Sumário

Este capítulo apresenta um estudo teórico dos conceitos fundamentais para a realização do projeto. Ao examinar o funcionamento das plataformas *low-code* é possível compreender a importância destas. Na reitoria da Universidade de Lisboa, o Joget Workflow é uma plataforma muito utilizada. Esta plataforma é constituída por diversos *plugins*.

A Universidade de Lisboa é uma instituição pela qual passam inúmeros documentos PDF que têm de ser assinados, como tal teria muita utilidade para esta possuir um *plugin* que permitisse fazer as assinaturas digitais. Este capítulo permite ainda compreender como funcionam as assinaturas digitais pondo foco nas assinaturas feitas com a CMD.

O capítulo seguinte apresenta o trabalho realizado e tem por base os conceitos explicados.

Capítulo 3

Análise e desenho

No presente capítulo é descrito o trabalho de análise e desenho no contexto deste projeto. Primeiramente é apresentada a análise dos requisitos funcionais e não funcionais, seguindo-se uma descrição da arquitetura e desenho da solução proposta.

3.1 Análise de requisitos

Nesta secção são apresentados requisitos funcionais e não funcionais identificados para o artefacto que queremos criar para o Joget. É ainda apresentado um caso de uso com base na análise feita.

3.1.1 Requisitos funcionais e não funcionais

- **Assinatura digital -** O sistema deve suportar a assinatura digital de documentos.
- **Escolha do documento -** O utilizador deve ser capaz de escolher o documento que pretende assinar de forma inequívoca.
- **Visualização do documento escolhido -** O utilizador deve conseguir visualizar o documento que escolheu assinar estando assim a aplicação em conformidade com a política *WYSIWYS (What You See Is What You Sign)*.
- **Assinatura personalizada -** O utilizador deve poder personalizar a forma como o documento é assinado e ainda decidir se este deve ter uma assinatura visível ou não.
- **Assinatura visível opcional -** O sistema deve ser capaz de colocar uma assinatura visível no documento PDF, caso o utilizador o escolha.
- **Livre de *hardware* -** Não deve ser necessário a utilização de *hardware* adicional, como por exemplo leitores de cartões.
- **Livre de *software* adicional -** Não deve ser necessário o utilizador instalar *software* novo.
- **Abranger todos os utilizadores -** Todos os utilizadores das aplicações criadas a partir do Joget devem conseguir assinar digitalmente documentos.
- **Sem custo para o utilizador -** A assinatura digital não deve ter um custo monetário associado.

- **Segurança dos dados -** A aplicação tem de garantir a segurança dos dados inseridos pelo utilizador. Os dados sensíveis do utilizador não podem ser guardados localmente.
- **Segurança da assinatura -** A aplicação tem de garantir que se mantêm as propriedades de segurança asseguradas pelas assinaturas digitais. Assim como garantir que os elementos de segurança como as credenciais do utilizador não são divulgados nem disponibilizados a terceiros. A comunicação entre o cliente e o servidor Joget deve ser segura e autenticada.

3.1.2 Caso de uso

A partir do levantamento de requisitos foi definido um caso de uso.

Nome: Assinar Digitalmente Documentos
Ator Principal: Utilizador da aplicação criada no Joget Workflow
Cenário Principal de sucesso:

1. O utilizador escolhe o documento PDF que pretende assinar.
2. A aplicação apresenta ao utilizador o documento que foi inserido de modo a que este possa verificar visualmente se era efetivamente aquele documento que queria assinar.
3. O utilizador confirma que pretende assinar aquele documento.
4. A aplicação apresenta uma interface em que o utilizador deve inserir os seus dados relacionados com a assinatura.
5. O utilizador escolhe se a sua assinatura será visível ou não, e insere os seus dados relacionados com a assinatura.
6. A aplicação confirma a identidade do utilizador e produz uma assinatura para o documento em questão, uma vez que o documento esteja assinado é guardado com um nome diferente junto do documento original.

Diagrama de sequência: Figura 3.1

3.2 Arquitetura e desenho da solução

Apresenta-se nesta secção a arquitetura e o desenho da solução, que tiveram na sua origem o funcionamento da Chave Móvel Digital (CMD) e o levantamento de requisitos feito.

3.2.1 Arquitetura da Solução

A arquitetura da solução teve por base o funcionamento da CMD. Na Figura 3.2 temos representada a ligação entre os diversos componentes. Estes componentes estão divididos entre o Joget Workflow e a AMA. A aplicação criada no Joget Workflow é composta pelo cliente e pelo servidor, sendo o servidor que faz a comunicação com a AMA. No entanto, os componentes da AMA são geridos pela própria. A aplicação criada consegue apenas comunicar com o Servidor do módulo de segurança.

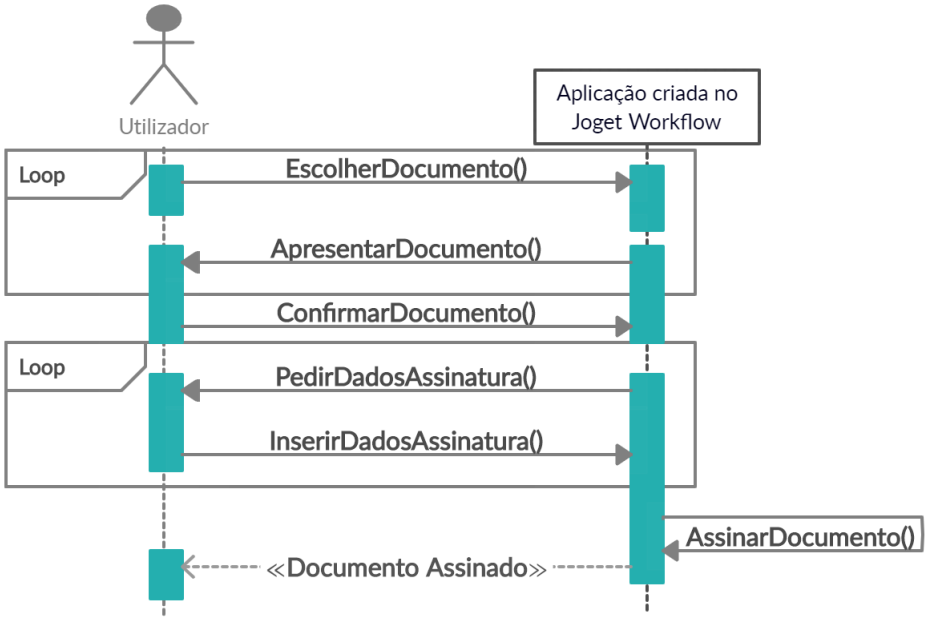

Figura 3.1: Diagrama de sequência do caso de uso Assinar Digitalmente Documentos

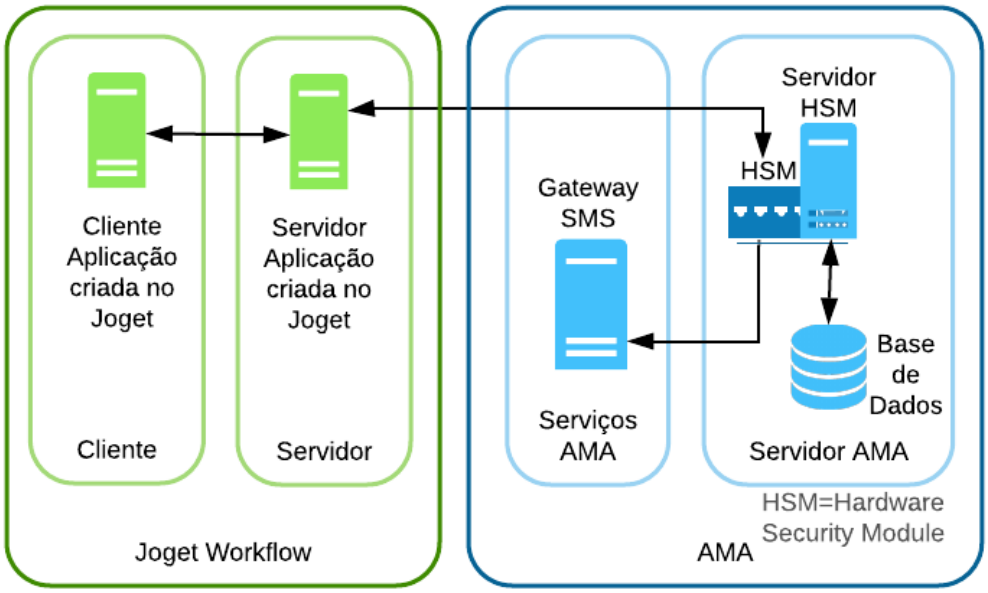

Figura 3.2: Arquitetura da solução - Ligação entre diversos componentes

Listam-se de seguida as funções de cada um dos componentes.

- O **cliente da aplicação criada no joget** serve de interface ao utilizador. É no cliente que se obtêm os dados essencias do utilizador. Este componente comunica apenas com o servidor da aplicação.
- O **servidor da aplicação criada no joget** é onde são processados os pedidos provenientes do cliente. Este servidor comunica com o servidor do módulo de segurança da AMA de modo a obter os resultados das operações de assinatura que depois de processados são enviados ao cliente da aplicação. É também o servidor da aplicação que coloca a assinatura no documento. Como foi mencionado, este componente comunica com o cliente da aplicação e com o servidor da AMA.

- O **servidor do módulo de segurança da AMA** recebe os pedidos do servidor da aplicação, executa as operações relativas à assinatura digital e devolve ao servidor da aplicação o resultado dessas operações. Este servidor verifica ainda a identidade do utilizador recorrendo à base de dados da AMA e ao gateway de SMS. Este componente comunica com o servidor da aplicação criada no joget, com a base de dados da AMA e com o gateway de SMS.

- A **base de dados da AMA** é onde estão guardados os dados relacionados com a CMD do utilizador, tais como o certificado do utilizador, o seu nome, número de identificação, entre outros. Este componente comunica com o servidor da AMA.

- O **gateway de SMS** tem a função de enviar ao utilizador uma mensagem para o telemóvel com o código de verificação, usualmente chamado *One Time Password (OTP)*. Este componente comunica com servidor da AMA e com o telemóvel do utilizador.

3.2.2 Desenho da Solução

Esta sub-secção começa por apresentar um diagrama de sequência, na Figura 3.3. Este diagrama representa graficamente as interações entre os diversos componentes.

Para acompanhar este diagrama temos as Figuras 3.4, 3.5, 3.6, 3.7, 3.8 e 3.9. Estas figuras representam uma possível interface gráfica com que o utilizador irá interagir.

O fluxo apresentado no diagrama de sequência começa quando o utilizador escolhe um documento para ser assinado, e tem associada a Figura 3.4. Em alternativa, os documentos podem ser produzidos no Joget e, por esse motivo já constarem no *workflow*, como exemplificado na Figura 3.5

Este documento é guardado e depois apresentado ao utilizador para que este confirme se é efetivamente este o documento que pretende assinar. A representação gráfica para o utilizador encontra-se na Figura 3.6.

No momento em que o utilizador confirma o documento que pretende assinar, é pedido que insira os seus dados relativos à CMD, o seu número de telemóvel e o pin. Além destes dados o utilizador tem ainda a opção de escolher se pretende obter uma assinatura visível e em que zona do documento é que esta se posicionará, como se pode observar na Figura 3.7. Com estes dados são realizadas duas operações do serviço de assinatura digital da AMA, as operações **getCertificate** e **SCMDSign**. Com estas operações obtemos o certificado do cidadão que será inserido no ficheiro para mais tarde ser possível verificar a sua validade. Com a segunda operação começamos o processo de criação da assinatura e é enviada uma mensagem ao utilizador com o seu código temporário. Caso as operações falhem pelos dados não estarem corretos o utilizador é notificado e pode inseri-los novamente.

Ao ser enviado o código ao utilizador a aplicação pede ao mesmo que insira o código que recebeu no seu telemóvel, tal como representado na Figura 3.8. Com este código podemos realizar a última operação da assinatura da AMA. A operação **validateOTP**, quando concluída com sucesso, envia a assinatura ao servidor da aplicação. É neste momento que se coloca a assinatura no documento e este é guardado junto ao original. Caso a operação não seja realizada com sucesso, o utilizador é informado pela aplicação que inseriu mal o código e tem oportunidade de o inserir novamente.

De modo a finalizar este fluxo, é apresentado ao utilizador o documento assinado, como representado na Figura 3.9. O documento que é apresentado foi guardado junto do original com um nome diferente.

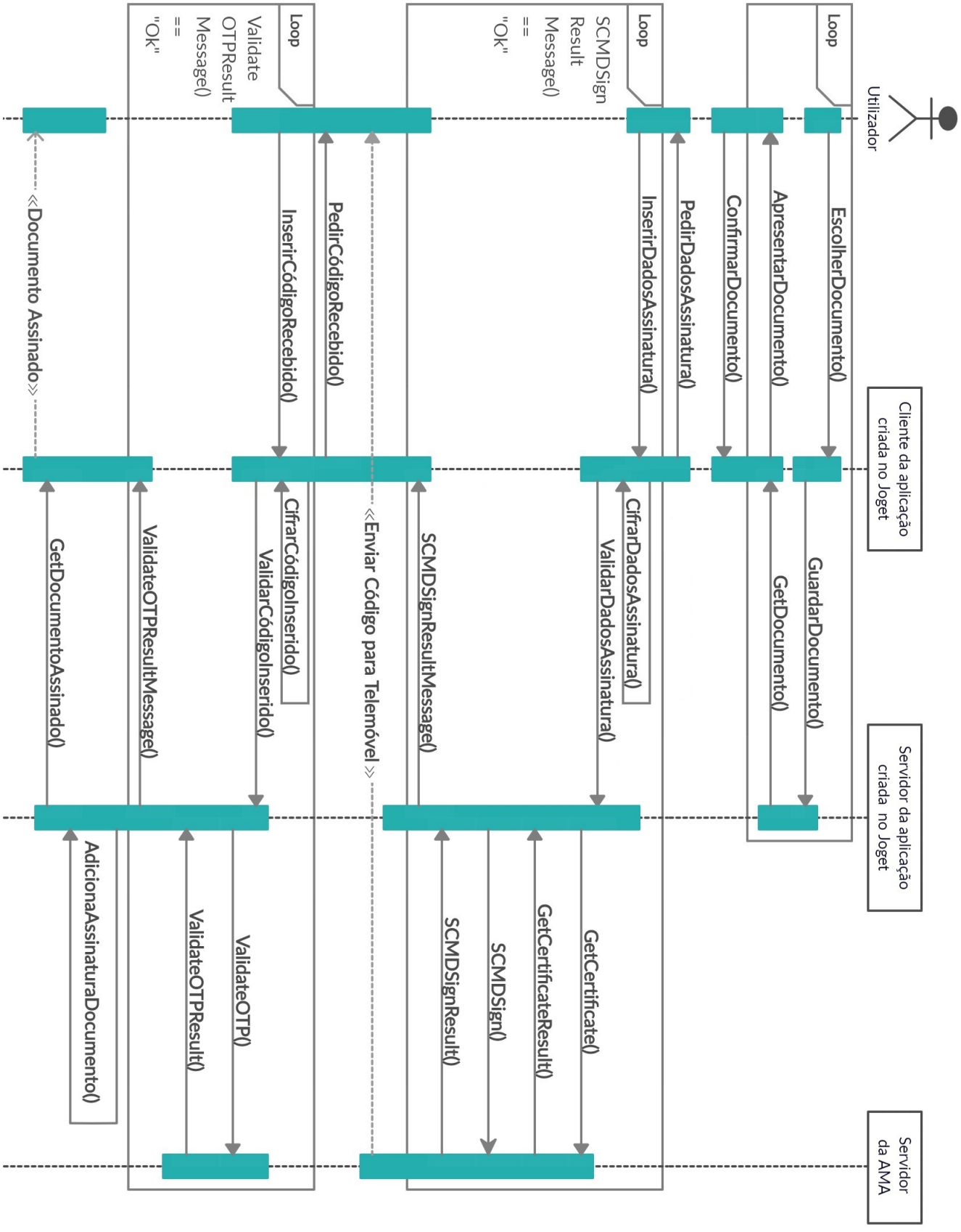

Figura 3.3: Diagrama de sequência da solução

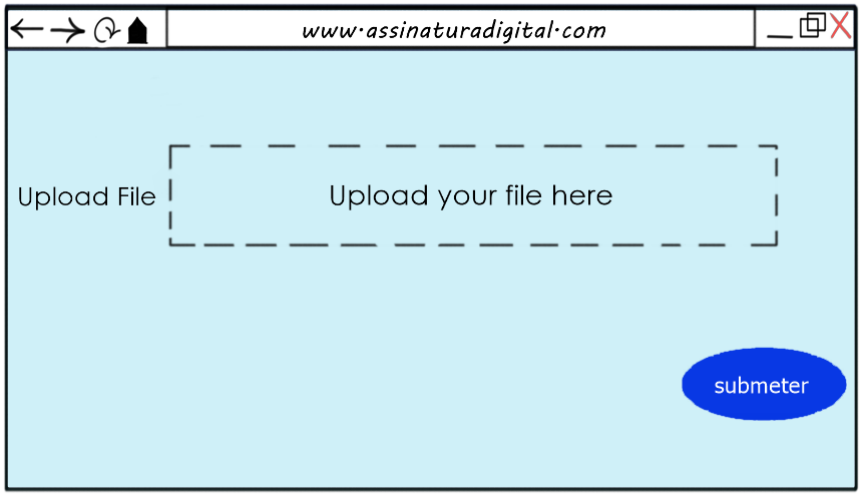

Figura 3.4: Possível Desenho da solução - Escolha do Documento

Figura 3.5: Possível Desenho da solução - Documento gerado no workflow

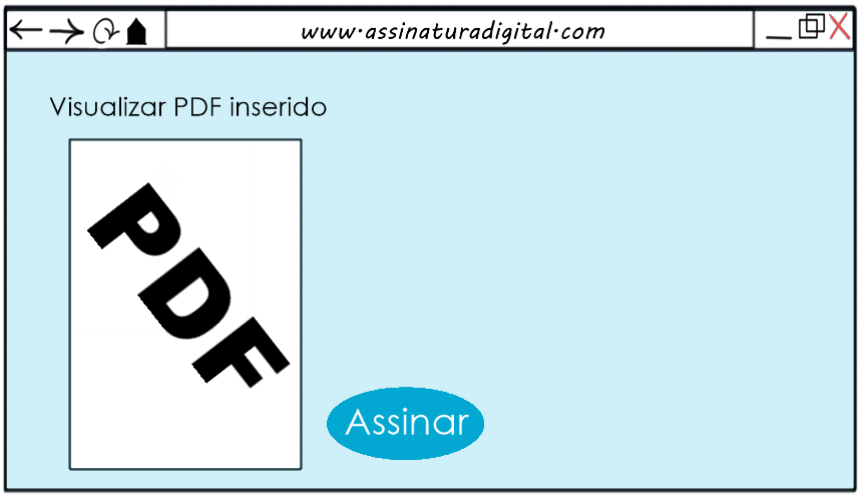

Figura 3.6: Possível Desenho da solução - Confirmação do Documento

Figura 3.7: Possível Desenho da solução - Inserção dos dados da CMD e opções da assinatura

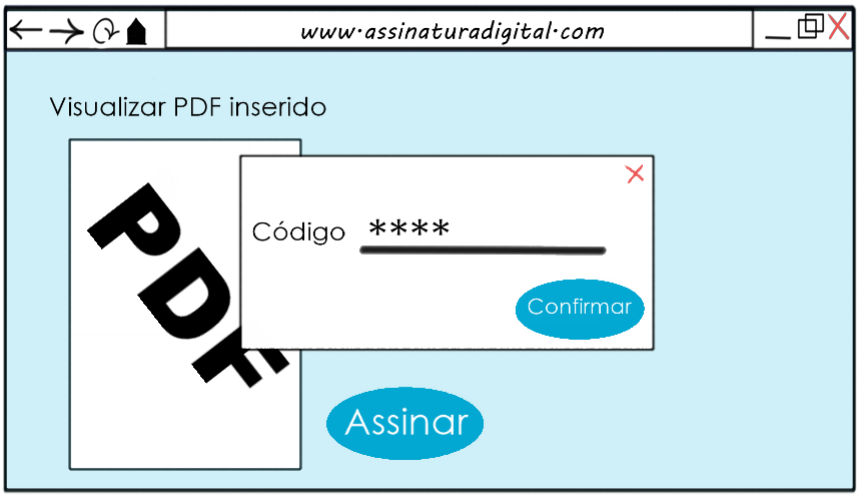

Figura 3.8: Possível Desenho da solução - Inserção do código recebido no telemóvel

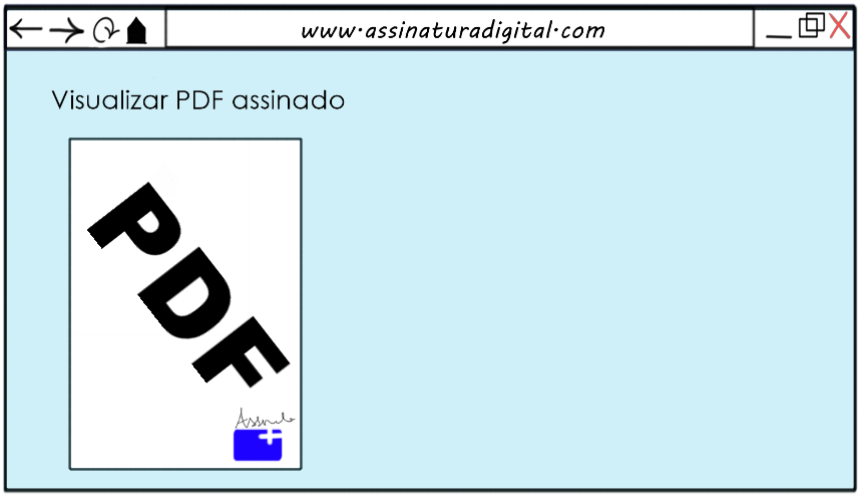

Figura 3.9: Possível Desenho da solução - Visualização do documento assinado

3.3 Sumário

No início do presente capítulo apresenta-se uma análise dos requisitos funcionais e não funcionais e o caso de uso. De seguida, encontramos a arquitetura e o desenho da solução que inclui uma figura que ilustra a forma como os diversos componentes comunicam, um diagrama de sequência que clarifica o funcionamento da aplicação e uma possível apresentação da aplicação no Joget.

No capítulo seguinte é descrita a forma como foi implementada a solução.

Capítulo 4

Implementação

Este capítulo tem como objetivo explicar a forma como foi implementada a solução. Esta implementação teve por base o desenho e arquitetura apresentados anteriormente.

A implementação deste artefacto dividiu-se entre as suas componentes de cliente e servidor da aplicação. A componente de cliente suporta as interações do utilizador com a aplicação. As principais operações pelas quais esta componente está responsável são a apresentação do documento que se vai assinar e a recolha dos dados necessários para se criar a assinatura. Adicionalmente, a componente de servidor da aplicação processa os dados vindos do cliente com a finalidade de colocar uma assinatura num documento PDF. Descreve-se ainda a forma como foi desenvolvido o artefacto complementar.

4.1 Cliente da aplicação

O cliente da aplicação contém os serviços que impliquem uma interação com o utilizador. É nesta componente que se desenvolve a apresentação do artefacto. As suas principais funcionalidades são mostrar ao utilizador o PDF que será assinado, de modo a que este possa confirmar que é o correto, pedir os dados relacionados com a assinatura ao utilizador e enviá-los para o servidor da aplicação.

4.1.1 Apresentação do artefacto Joget

Os artefactos do Joget são compostos por uma pasta *"templates"* onde se encontra o ficheiro ".ftl". Este ficheiro é utilizado para gerar dinamicamente o código HTML do plugin e é escrito de acordo com a linguagem FreeMarker Template Language (FTL). Esta linguagem de programação contém elementos HTML, CSS e JavaScript. A sua maior vantagem é conseguir colocar elementos vindos do código Java diretamente no código que é gerado. Pode-se observar como isso acontece na Listagem 4.1 que mostra como se foi buscar o valor da variável "url" na classe ftl.

```
1 (...)
2 @Override
3 public String renderTemplate(FormData formData, Map dataModel) {
4     String template = "pdfsigncmd.ftl";
5
6     (...)
7     dataModel.put("urlDoc", getUrl(formData));
8     (...)
```

```
9
10      String html = FormUtil.generateElementHtml(this, formData, template, dataModel);
11      return html;
12  }
```

Listagem 4.1: Excerto da classe PdfSignCMD.java - passar um elemento da classe Java para o ftl.

A apresentação do artefacto é composta por duas partes principais. A primeira é onde o utilizador consegue visualizar o PDF que pretende assinar e a segunda é onde o utilizador insere os dados necessários para realizar a assinatura.

Para a primeira parte foi utilizada a biblioteca PDF.JS que é uma biblioteca JavaScript open-source desenvolvida pela Mozilla. Esta foi criada com o intuito de facilitar a leitura de documentos PDF no browser. A biblioteca é composta por código JavaScript que se coloca no cliente da aplicação e vai permitir a visualização dos documentos PDF num elemento HTML <canvas>. É bastante utilizada devido à simplicidade das suas dependências e ao facto dos seus elementos HTML e CSS serem facilmente customizados. Pode-se observar na Listagem 4.2 a criação do objeto ”canvas” que vai ser utilizado para se visualizar o documento PDF.

```
1  (...)
2  <canvas id="the-canvas"></canvas>
3
4  <script type="text/javascript">
5      var url = "${urlDoc}";
6
7      // Loaded via <script> tag, create shortcut to access PDF.js exports.
8      var pdfjsLib = window['pdfjs-dist/build/pdf'];
9
10     // The workerSrc property shall be specified.
11     pdfjsLib.GlobalWorkerOptions.workerSrc = '${request.contextPath}/plugin/org.joget.plugin.
       enterprise.PdfSignCMD/js/pdfsign/pdf.worker.js';
12
13     var pdfDoc = null,pageNum = 1,pageRendering = false, pageNumPending = null, scale = 0.8,
       canvas = document.getElementById('the-canvas'),ctx = canvas.getContext('2d');
14
15     /**
16      * Get page info from document, resize canvas accordingly, and render page.
17      * @param num Page number.
18      */
19     function renderPage(num) {
20       pageRendering = true;
21       // Using promise to fetch the page
22       pdfDoc.getPage(num).then(function(page) {
23         var viewport = page.getViewport({scale: scale});
24         canvas.height = viewport.height;
25         canvas.width = viewport.width;
26
27         // Render PDF page into canvas context
28         var renderContext = {
29           canvasContext: ctx,
30           viewport: viewport
31         };
32         var renderTask = page.render(renderContext);
33
34         // Wait for rendering to finish
35         renderTask.promise.then(function() {
36           pageRendering = false;
37           if (pageNumPending !== null) {
```

```
38              // New page rendering is pending
39              renderPage(pageNumPending);
40              pageNumPending = null;
41            }
42          });
43        });
44
45        // Update page counters
46        document.getElementById('page_num').textContent = num;
47      }
48  (...)
```

Listagem 4.2: Excerto da classe PdfSignCMD.ftl - criação do objeto canvas

Para a segunda parte foi criado um botão que despoleta um elemento de <dialog> em que o utilizador deve preencher o seu número de telemóvel e pin associados à CMD, e no qual pode ainda escolher onde fica a assinatura, colocar o motivo e local da assinatura, como se pode visualizar na Listagem 4.3. Este primeiro <dialog> tem ainda um botão para o utilizador passar para o elemento de <dialog> seguinte, no qual o utilizador coloca o código OTP que recebeu no telemóvel.

```
1  (...)
2  <script>
3          var signDocumentButton = document.getElementById('signDocumentButton');
4          var signDialog = document.getElementById('signDialog');
5
6          signDocumentButton.addEventListener('click', function() {
7              signDialog.showModal();
8          });
9  </script>
10 (...)
```

Listagem 4.3: Excerto da classe PdfSignCMD.ftl - despoletar elemento dialog.

4.1.2 Ligação entre o cliente e o servidor da aplicação

Como é visível na Figura 3.3, no processo de criação da assinatura existem dois momentos em que o cliente da aplicação envia pedidos ao servidor da aplicação. Em ambos os pedidos são enviados um conjunto de dados base relativos à aplicação e dados inseridos pelo utilizador.

Os dados base relativos à aplicação são o *id*, a *password* e o *username* atribuídos pela AMA, o *endpoint* da AMA e os dados específicos da aplicação criada como o *id*, a versão, a tabela da base de dados associada, o nome do documento que será assinado, entre outros.

No primeiro pedido são enviados para o servidor os dados essenciais cifrados para a criação da assinatura como o número e o pin do utilizador associado à CMD. A forma como estes dados são cifrados encontra-se na Listagem 4.4 e utiliza a chave pública da AMA fornecida no certificado X.509. É também envida informação sobre se a assinatura é visível ou não. No caso da assinatura ser visível, são enviados também a página e a zona da página em que a assinatura será colocada e, opcionalmente, o local e o motivo da assinatura. A resposta do servidor para o cliente inclui o certificado e o *processId* que devem ser utilizados na próxima função.

```
1  (function($){
2      $.fn.extend({
```

```
3     pdfsigncmd : function(args){
4             var encrypt = new JSEncrypt();
5             var pubKey = decodeURIComponent(data.pubKey);
6             encrypt.setPublicKey(pubKey);
7     (...)
8         var phoneNumberPlace = data.signDialog.find('#_phoneNumber_');
9         var phoneNumber = "+351"+phoneNumberPlace.val();
10        var encrypedtPhone = encrypt.encrypt(phoneNumberPlace);
11        phoneNumberPlace.val('')
12    (...)
```

Listagem 4.4: Excerto da classe jquery.pdfcmdsign.js - cifra do número de telemóvel.

No segundo pedido, para além das informações base, é enviado o código OTP cifrado que o utilizador recebeu no seu telemóvel e inseriu na aplicação. Neste caso, a resposta do servidor será só para informar o utilizador que o documento se encontra assinado ou, caso isto não tenha sido possível, informar o utilizador que houve um problema e o documento não ficou assinado.

De forma a criar uma ligação entre o cliente e o servidor foi utilizada a função Ajax da biblioteca jQuery do JavaScript. Esta função permite realizar um pedido HTTP assíncrono, ou seja, com um pedido Ajax, a página Web consegue receber novas informações do servidor sem ser necessário recarregar a página toda, tornando assim mais iterativa a página em questão.

Com o intuito de tornar esta ligação mais segura é colocado nos endereços utilizados para fazer o pedido entre o cliente e o servidor um *nonce*. O *Nonce* é um número aleatório que só pode ser utilizado uma vez. A palavra inglesa tem origem na junção das palavras *Number* e *Once*. O *nonce* é normalmente utilizado para aumentar a segurança num processo de autenticação. Utilizando um *nonce* conseguimos garantir que cada chamada *ajax* vai ser feita com um conjunto de parâmetros definidos e apenas com esses. Neste caso é importante pois são esses parâmetros que contêm informação sobre que documento é que vai ser assinado e onde é que se deve escrever na base de dados. Desta forma, garantimos que os parâmetros são os correctos. Isto impede que um utilizador com um *nonce* válido assine um documento diferente ou de ler/escrever num documento de um qualquer campo da base de dados.

4.2 Servidor da aplicação

O servidor da aplicação tem como objetivo receber os dados vindos do cliente e processá-los de modo a poder colocar uma assinatura no documento PDF. Isto é feito com o auxílio dos métodos da AMA e de bibliotecas utilizadas para manipular documentos PDF.

Neste projeto foi criado um plugin, no Joget Workflow, do tipo *"Form Field Element"* como tal a classe Java estende a classe *"Element"* e implementa as classes *"FormBuilderPaletteElement"* e *"PluginWebSupport"*.

Como mencionado na secção 2.1.2, existem duas arquiteturas sobre as quais se pode desenvolver um plugin. Neste caso foi utilizada a ***Standard Java Plugin***, o que facilitou o acesso às dependências necessárias facilitando consequentemente o desenvolvimento do artefacto. Com este tipo de arquitetura o artefacto encontra-se junto dos artefactos base do Joget que sejam do mesmo tipo, neste caso do tipo *"Form Field Element"*.

A estrutura base deste plugin é igual à estrutura de um projeto Maven em que temos o ficheiro ".pom" onde se colocam as dependências necessárias para o projeto e dois *packages* diferentes, o *source*

e o *resources*.

O *package resources* foi utilizado para criar o cliente da aplicação como explicado na secção 4.1.

O *package source* contém o código Java do artefacto, responsável pelo processamento dos dados vindos do cliente da aplicação, pela comunicação com a AMA e pela manipulação do documento PDF para que este contenha uma assinatura conforme detalhado nas subsecções seguintes. É nesta classe que se faz uso da biblioteca Digital Signature Service (DSS) e do SCMDService, a *api* disponibilizada pela AMA.

A biblioteca DSS, desenvolvida pelo Connecting Europe Facility (CEF), foi criada com o intuito de oferecer uma infraestrutura genérica e reutilizável para qualquer projeto europeu, facilitando assim o desenvolvimento de serviços públicos digitais. As principais funcionalidades da biblioteca são adicionar assinaturas (visíveis ou não) a um documento e validar essas assinaturas. Esta biblioteca é disponibilizada através de um repositório Maven. Relativamente aos padrões de assinaturas digitais mencionados na sub-secção 2.3.2 a biblioteca DSS fornece vários. No contexto do projeto foi escolhido o padrão PDF Advanced Electronic Signatures (PAdES) .

4.2.1 Processamento dos dados

Para se utilizar o artefacto desenvolvido é necessário haver um documento PDF que se pretenda assinar. Este documento deve estar presente no contexto da aplicação. Para tal deve ser utilizado o plugin *"File Upload"* ou deve ser gerado um documento dentro da aplicação.

Com o intuito de posteriormente conseguirmos manipular documentos PDF foi necessário primeiro compreender como o Joget guarda os documentos. Os documentos são guardados numa pasta no servidor e o seu nome é guardado na base de dados. A tabela da base de dados em que se regista o nome do documento tem o mesmo nome que a pasta onde o documento é guardado no servidor. Dentro desta pasta existe outra cujo nome é igual ao *"id"* do processo. O *"id"* do processo corresponde também à chave primária da tabela. Dentro desta última pasta encontramos o documento guardado.

Resumindo para obter o documento é necessário sabermos o nome da tabela da base de dados, a chave primária ou o *"id"* do processo, assim como o nome do *"field"* do elemento em que o ficheiro foi guardado, como se pode observar na Listagem 4.5.

```
1 (...)
2 String url = "";
3 url =request.getContextPath() + "/web/client/app/" + appDef.getAppId() + "/" + appDef.
     getVersion().toString() + "/form/download/" + sourceForm.getPropertyString(FormUtil.
     PROPERTY_ID) + "/" + primaryKeyValue + "/" + encodedFileName+ ".";
4 (...)
```

Listagem 4.5: Excerto da classe PdfSignCMD.java - localizar documento.

Como mencionado anteriormente, é necessário o utilizador introduzir os seus dados relacionados com a CMD e tomar algumas decisões sobre como irá ficar a assinatura. Estes dados são introduzidos no cliente da aplicação e são enviados para o servidor utilizando duas chamadas ajax, descritas anteriormente. É durante estas chamadas ajax que são realizadas as operações da AMA descritas em seguida.

Na primeira chamada ajax são realizadas as operações ***GetCertificate*** e ***SCMDSign*** que devolvem respetivamente o certificado do cidadão e o *processId* que são utilizados na operação ***ValidateOtp***. Por

este motivo estas operações são realizadas durante a primeira chamada ajax e o seu resultado é enviado no pedido da segunda chamada ajax. Na última operação é ainda enviado ao utilizador o seu código OTP para o telemóvel. Na segunda chamada ajax é realizada a operação ***ValidateOtp*** que valida o código inserido pelo utilizador e devolve a assinatura que deve ser colocada no documento.

4.2.2 Manipulação de documentos PDF

A manipulação de documentos PDF é feita antes das operações da AMA e no final de modo a colocar a assinatura no documento.

No início é necessário criar a síntese do documento e criar os parâmetros que irão fazer parte da assinatura. Isto é feito durante a primeira chamada ajax com o auxílio das bibliotecas PDFBox e DSS.

Na segunda chamada ajax a assinatura é colocada no documento PDF com o auxílio da biblioteca DSS.

Os documentos PDF podem ser assinados com assinaturas visíveis e invisíveis. Isto significa que podem ou não ter a assinatura escrita no documento. Uma pessoa consegue saber se um documento PDF contém uma assinatura invisível se o abrir num programa como o "Adobe Reader". As assinaturas visíveis são feitas exatamente da mesma forma, levam apenas mais uma imagem e um pequeno texto.

No projeto, quando queremos ter uma assinatura invisível é só seguir os passos descritos nas secções seguintes e no final criar um objeto "SignatureValue" para inserir a assinatura. Porém, se quisermos uma assinatura visível, temos de criar os objetos "SignatureImageParameters" e "SignatureImageTextParameters" para serem adicionados aos parâmetros básicos da assinatura. Esta adição de parâmetros tem de acontecer antes da operação SCMDService, caso isto não aconteça a síntese do documento não irá estar correta.

4.2.3 Comunicação com a AMA

De forma a começar o desenvolvimento do projeto foi necessário pedir à AMA a documentação relacionada com a assinatura de documentos com a CMD. Foi fornecido um documento com a especificação dos serviços de assinatura, um certificado (*X.509*), um *applicationId* para que a AMA consiga identificar qual é a aplicação com que está a comunicar, credenciais para a aplicação e um *WSDL* através do qual se gerou as diversas classes que constituem o *SCMDService*.

O *SCMDService* é o serviço que nos vai permitir estabelecer a comunicação com a AMA. Este serviço requer a implementação de comunicação *HTTPS* com autenticação básica (nome de utilizador e palavra-passe) e mensagens *SOAP*. As credenciais utilizadas para esta autenticação foram também fornecidas pela AMA. O *SCMDService* fornece 3 operações principais como mencionado na subsecção 2.2.2. A Figura 4.1 representa essas operações assim como os seus *input* e *output*.

A AMA fornece dois *endpoints* para a implementação deste serviço, sendo um deles o de produção e o outro de pré-produção. Este projeto foi desenvolvido utilizando o de pré-produção. Visto que as CMD de produção não são válidas neste ambiente, foi necessário criar uma CMD de pré produção. A criação desta chave é feita em `https://pprwww.autenticacao.gov.pt/web/guest/login/pedido-cmd` ou num balcão exatamente como referido na subsecção 2.2.1.

De forma a obtermos a assinatura para colocar no documento PDF temos de implementar três

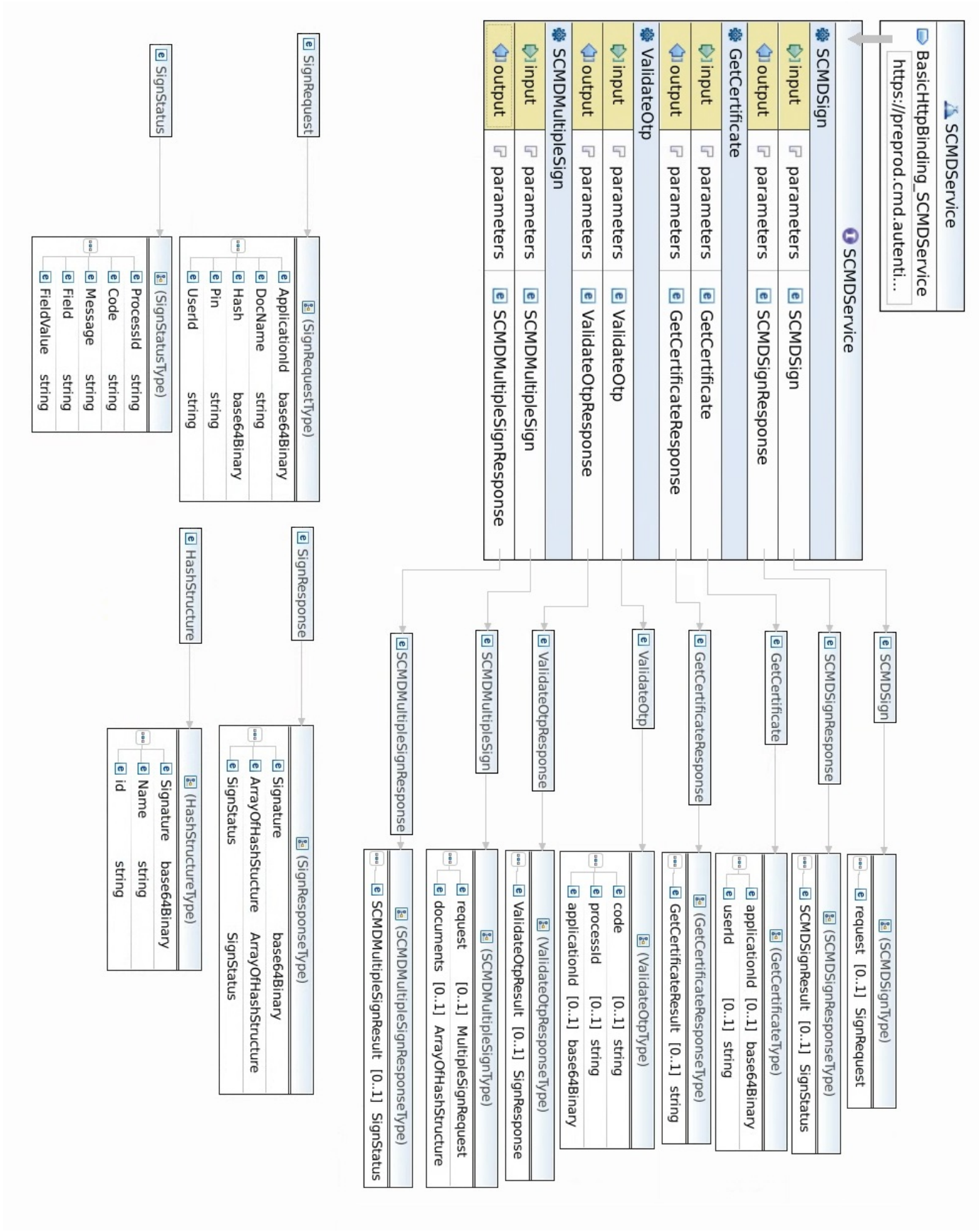

Figura 4.1: Classes presentes no WSDL

operações. Como mencionado anteriormente para esse efeito foi utilizada a biblioteca DSS, com o padrão PAdES e a biblioteca PDFBox para manipular os documentos PDF.

GetCertificate

A primeira operação é a ”GetCertificate” que devolve o certificado do cidadão. Este certificado é utilizado mais tarde quando é inserido no documento PDF e tem informações relativas ao cidadão, tais como, o nome e número de identificação civil, os quais serão úteis para construir uma assinatura visível. Na Listagem 4.6 pode-se observar esta operação.

Para realizar esta operação é necessário enviar o *applicationId* fornecido previamente e o número de telemóvel inserido pelo utilizador na aplicação, já cifrado no cliente da aplicação.

```
1  (...)
2  //OPERACAO--getCertificate
3  String certificate = getCert(cmdService, phoneNumber, appId_AMA);
4  (...)
5  public String getCert(SCMDService cmdService, String numero, String appId) {
6          try {
7              String userIdChiphered = numero;
8              byte[] byte_app = (appId.getBytes(StandardCharsets.UTF_8));
9
10             return cmdService.getCertificate(byte_app, userIdChiphered);
11         } catch (Exception e) {
12             e.printStackTrace();
13         }
14         return null;
15     }
16 (...)
```

Listagem 4.6: Excerto da classe PdfSignCMD.java - Operação GetCertificate.

SCMDSign

De forma a realizar esta operação temos primeiro de criar a síntese do documento que se pretende assinar, para tal é criado um objeto do tipo ”DSSDocument” utilizado para manusear o documento. De seguida é criado o objeto ”PAdESSignatureParameters” no qual se inserem os parâmetros relativos à assinatura, como o certificado que obtivemos na operação anterior, a cadeia de assinatura (presente no certificado), o nível da assinatura, o algoritmo de síntese, a data, o local e o motivo da assinatura. Nesta fase utilizamos os objetos relativos ao documento e aos parâmetros da assinatura como argumentos do método ”getDataToSign” depois de inicializar o serviço ”PAdESService”. Desta forma, obtém-se o objeto do qual temos de fazer a síntese. A síntese é feita de acordo com o documento das especificações, que faz referência ao documento [13]. Feita a síntese esta é concatenada bit a bit com o prefixo (que mais uma vez se encontra no documento das especificações), e é enviada no método SCMDSign.

No método SCMDSign para além da síntese envia-se também o nome do documento, o ”applicationId”, o número e o *pin* do utilizador cifrados. Na Listagem 4.7 pode-se observar esta operação.

```
1  (...)
2  //OPERACAO--SCMDSign
3  SignStatus scmdSign = getSignHashResult(cmdService, phoneNumber, pin, dataToSign, fileName,
       appId_AMA);
4  String pid = scmdSign.getProcessId();
5  (...)
```

```
6  public SignStatus getSignHashResult(SCMDService cmdService, String numero, String
       pinUtilizador,
7              ToBeSigned dataToSign, String nomeDocumento, String appId) {
8          try {
9              String applicationId = appId;
10             byte[] byte_app = (applicationId.getBytes(StandardCharsets.UTF_8));
11             SignRequest request = new ObjectFactory().createSignRequest();
12             request.setApplicationId(byte_app);
13
14             String nomeDoc = nomeDocumento;
15             request.setDocName(new ObjectFactory().createSignRequestDocName(nomeDoc));
16             // Criar Hash
17             MessageDigest md = MessageDigest.getInstance(DIGEST_ALGORITHM);
18             byte[] hash = md.digest(dataToSign.getBytes());
19
20             byte[] sha256SigPrefix = { 0x30, 0x31, 0x30, 0x0d, 0x06, 0x09, 0x60, (byte) 0x86,
       0x48, 0x01, 0x65, 0x03,
21                     0x04, 0x02, 0x01, 0x05, 0x00, 0x04, 0x20 };
22             byte[] c = new byte[sha256SigPrefix.length + hash.length];
23             System.arraycopy(sha256SigPrefix, 0, c, 0, sha256SigPrefix.length);
24             System.arraycopy(hash, 0, c, sha256SigPrefix.length, hash.length);
25             request.setHash(c);
26             request.setPin(pinUtilizador);
27             request.setUserId(numero);
28             SignStatus scmdSign = cmdService.scmdSign(request);
29
30             return scmdSign;
31         } catch (NoSuchAlgorithmException e) {
32             e.printStackTrace();
33         } catch (Exception e) {
34             e.printStackTrace();
35         }
36         return null;
37     }
38 (...)
```

Listagem 4.7: Excerto da classe PdfSignCMD.java - Operação SCMDSign.

ValidateOTP

Sendo a operação final, é esta que devolve a assinatura. O que esta operação faz é enviar ao servidor o código que o utilizador recebeu no telemóvel ou no email de forma a que seja possível verificar se é efetivamente a própria pessoa a querer assinar. Nesta operação é necessário enviar o ”processId” que veio em resposta à operação anterior. Para além disto, é enviado também o *pin* que o utilizador inseriu devidamente cifrado. Na Listagem 4.8 pode-se observar esta operação.

```
1  (...)
2  //OPERACAO--ValidateOTP
3  SignResponse signResponse = getSignResponse(cmdService, pid, otp, appId_AMA);
4  byte[] signature = signResponse.getSignature();
5  (...)
6  public SignResponse getSignResponse(SCMDService cmdService, String processId, String code,
       String appId) {
7          try {
8              String applicationId = appId;
9              byte[] byte_app = (applicationId.getBytes(StandardCharsets.UTF_8));
10
11             String otp = code;
12             SignResponse validateOtpResponse = cmdService.validateOtp(otp, processId, byte_app
       );
```

```
13
14              return validateOtpResponse;
15          } catch (Exception e) {
16              e.printStackTrace();
17          }
18          return null;
19      }
20  (...)
```

Listagem 4.8: Excerto da classe PdfSignCMD.java - Operação ValidateOTP.

Depois destas três operações, é colocada a assinatura no PDF, conforme detalhado na secção seguinte.

4.3 Apresentação do documento assinado

Para se poder visualizar os documentos assinados na aplicação foi desenvolvido um artefacto complementar. Este artefacto permite ainda que o utilizador faça o *download* do documento. Este artefacto encontra-se dividido igualmente em duas componentes, o cliente e o servidor.

No cliente o artefacto utiliza também a biblioteca PDF.JS e faz uso do mesmo código desenvolvido para o plugin de assinatura, sendo que se acrescentou apenas a opção de *download* do documento. Para tal, é utilizada a linguagem FTL que nos permite colocar um elemento vindo do código Java presente no servidor, este elemento é a localização do ficheiro que está a ser apresentado e utiliza-se um elemento $<href>$ com o atributo *download*, como se pode observar na Listagem 4.9.

```
1 (...)
2 <menu>
3         <a class="btn" href="${nomeDocumento}" download>@@pdfview.label.download@@</a>
4 </menu>
5 (...)
```

Listagem 4.9: Excerto da classe ViewPdfFile.ftl - Download do documento.

O servidor neste caso serve apenas para se encontrar o documento assinado. Como este é guardado da mesma forma que o Joget já guardava os ficheiros, foi reutilizado o código do artefacto de assinatura.

4.4 Sumário

Este capítulo resume a forma como foi implementada a solução e explica ainda a solução desenvolvida para ser possível visualizar o documento assinado na aplicação. O artefacto está dividido em duas componentes o cliente e o servidor da aplicação. Sendo que no cliente estão implementados os serviços relacionados com a apresentação do artefacto e é onde se faz a ligação com o componente do servidor. Por sua vez o servidor é onde se processam os dados vindos do cliente, onde se manipula o PDF e onde são executadas as três operações da AMA.

Capítulo 5

Prova de conceito e validação

No presente capítulo são descritas a forma como foi feita a prova de conceito e a validação do artefacto criado. São apresentadas a aplicação SIADAP 3, que faz uso do artefacto, a forma como o artefacto é utilizado e os resultados do inquérito SUS feito a alguns funcionários da reitoria da Universidade de Lisboa que se disponibilizaram para testar esta funcionalidade da aplicação.

5.1 Prova de conceito - Aplicação SIADAP 3

O SIADAP é um sistema integrado de gestão e avaliação do desempenho na administração pública. Este sistema é divido em três componentes. A primeira destina-se a avaliar os serviços da administração pública, a segunda componente é destinada à avaliação dos dirigente e a terceira à avaliação dos trabalhadores. A aplicação que está a ser desenvolvida pela reitoria da Universidade de Lisboa, à data da implementação deste artefacto, concretiza apenas da terceira componente.

5.1.1 *workflow* da aplicação

O sistema de avaliação dos trabalhadores (SIADAP 3) tem carácter bienal, querendo com isto dizer que o sistema é utilizado para fazer a avaliação do desempenho durante os dois anos civis anteriores [6].

Como mencionado anteriormente, a aplicação ainda não está finalizada, pelo que neste momento, o *workflow* contempla apenas três intervenientes, o Avaliado, o Avaliador e o Direção Recursos Humanos (DRH). O Avaliado pode ser qualquer trabalhador dos serviços centrais da reitoria da Universidade de Lisboa das carreiras de Técnico Superior, Assistente Técnico, Assistente Operacional, Especialista de Informática, Técnico de Informática e Coordenador Técnico. O Avaliador é a chefia direta do avaliado. Contudo, na ausência deste, poderá ser a chefia de nível superior. O DRH é composto pelos trabalhadores deste departamento que estejam envolvidos na gestão do SIADAP 3.

A Figura 5.1 representa o fluxo implementado. Na figura as caixas azuis representam os momentos em que são assinados documentos.

O processo inicia-se com registo de todos os utilizadores e a criação do período e processo de avaliação, isto é feito pelo DRH. A primeira fase da avaliação é a definição de objetivos por parte do Avaliador, de seguida passa para o Avaliado de forma a que este possa confirmar e aceitar os objetivos propostos. Caso não sejam aceites, há uma reformulação, isto acontece até que os objetivos sejam aceites. No momento em que são aceites os objetivos, ambas as partes assinam o documento.

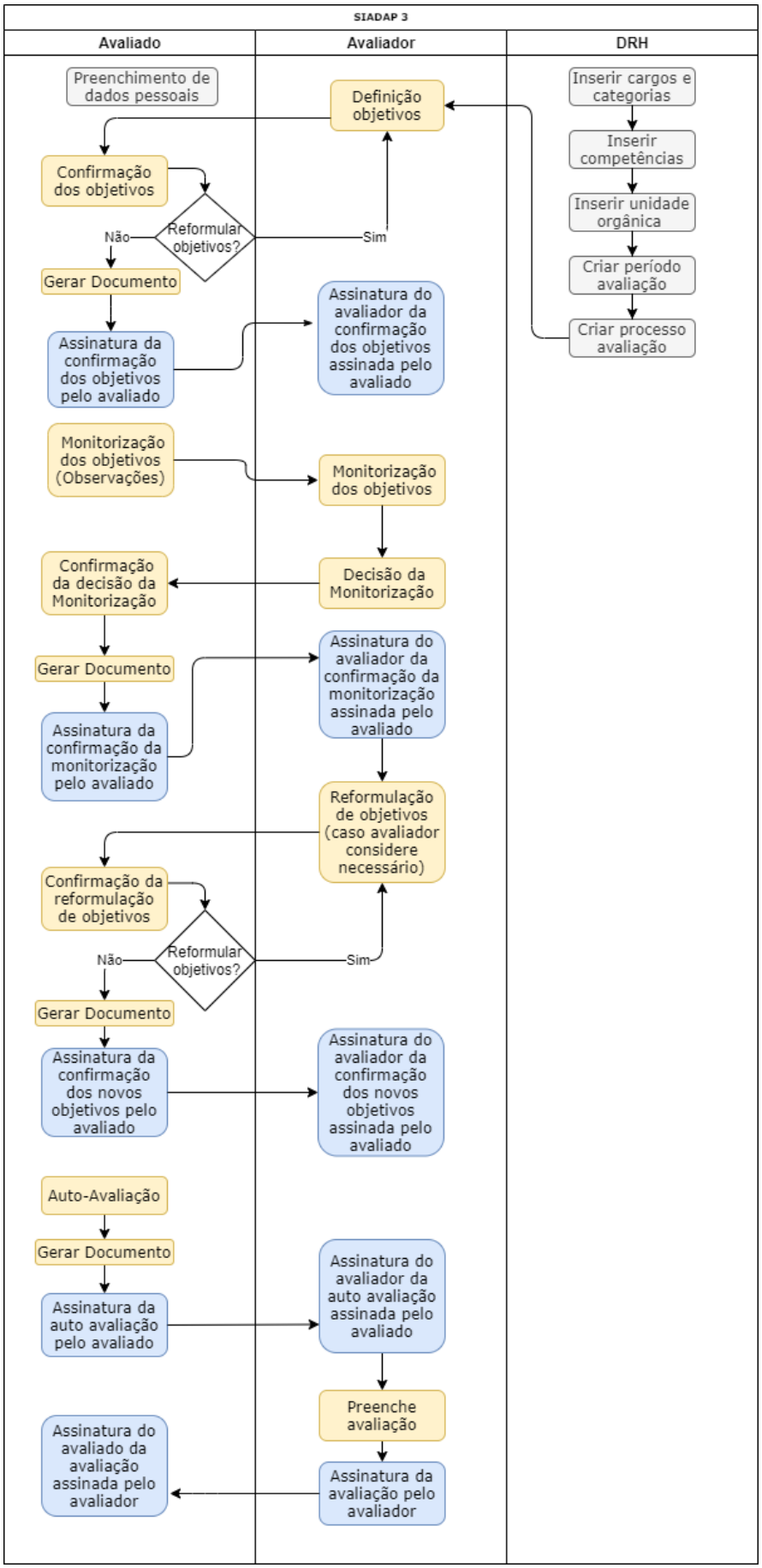

Figura 5.1: Workflow da aplicação SIADAP neste momento

Passado um ano da definição dos objetivos, é feita a monitorização dos mesmos. Nesta fase, o avaliado insere as suas observações sobre os objetivos e submete essas observações para o avaliador tomar conhecimento. O avaliador vê as observações e decide se haverá uma redefinição dos objetivos iniciais ou não. O avaliado toma conhecimento desta decisão e ambos assinam o documento. Caso haja um redefinição de objetos continua-se o fluxo, que é igual ao do passo anterior. Caso contrário, avança-se para o passo seguinte, a auto-avaliação (que só acontece passado um ano).

A auto-avaliação é feita pelo avaliado, assinada e será passada ao avaliador para este tomar conhecimento e assinar. De seguida o avaliador preenche a avaliação, assina-a e passa ao avaliado para este tomar conhecimento e assinar. A aplicação desenvolvida contempla apenas estes passos. Porém no futuro haverão mais fases que envolverão outros intervenientes.

5.1.2 Exemplo de utilização do artefacto na aplicação

O artefacto desenvolvido foi integrado na aplicação. De modo a exemplificar a forma como este é utilizado e sem divulgar dados sensíveis dos utilizadores, criou-se um processo de avaliação de um sujeito fictício chamado ”Ana” cuja chefia direta é um sujeito fictício chamado ”Bruno”.

A Figura 5.2 apresenta o ecrã inicial. Neste exemplo a ”Ana” quer assinar a sua ficha de auto-avaliação. Para tal seleciona o seu processo de avaliação e carrega no botão ”Ver Ficha”. Neste momento é gerado um PDF com a auto-avaliação feita pela ”Ana”. Este documento fica guardado na aplicação, como se pode visualizar na Figura 5.3.

Com o documento gerado a ”Ana” carrega em ”Assinar Documento” e vai para o ecrã apresentado na Figura 5.4. Neste ecrã pode pré-visualizar o documento gerado. Quando carrega no botão ”Assinar Documento” aparece o *pop-up* presente na Figura 5.5.

Neste *pop-up* a ”Ana” deve colocar os seu dados relativos à CMD (número de telemóvel e o pin), pode adicionar dados relativos ao local e motivo da assinatura e deve ainda decidir em que zona do documento será apresentada a assinatura. Uma vez que os dados estejam preenchidos, a ”Ana” carrega em confirmar e aparece o *pop-up* presente na Figura 5.6.

Neste *pop-up* a ”Ana” insere o código que recebeu no telemóvel e confirma. No final aparece um alerta de que o documento foi assinado, como na Figura 5.7. Se a ”Ana” voltar à página anterior pode visualizar que o documento assinado ficou guardado na aplicação (Figura 5.8).

Uma vez que o documento esteja assinado, acabou o fluxo que diz respeito à assinatura dos documentos com a CMD. No entanto, para o utilizador conseguir visualizar e fazer download do documento que assinou, foi desenvolvido um novo artefacto. Para fazer uso deste, a ”Ana” seleciona novamente o processo e escolhe a opção ”Ver Documento Assinado”que a redirecionará para um ecrã como o apresentado na Figura 5.9.

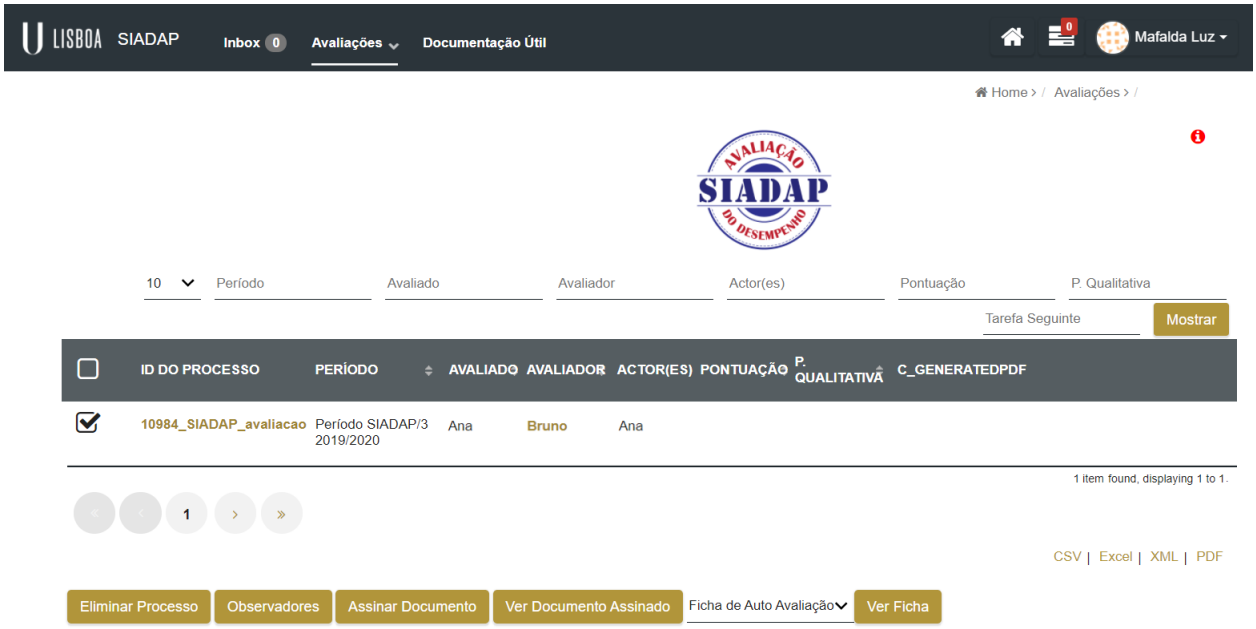

Figura 5.2: Fluxo assinatura da Auto-Avaliação na aplicação SIADAP3 - Ecrã inicial

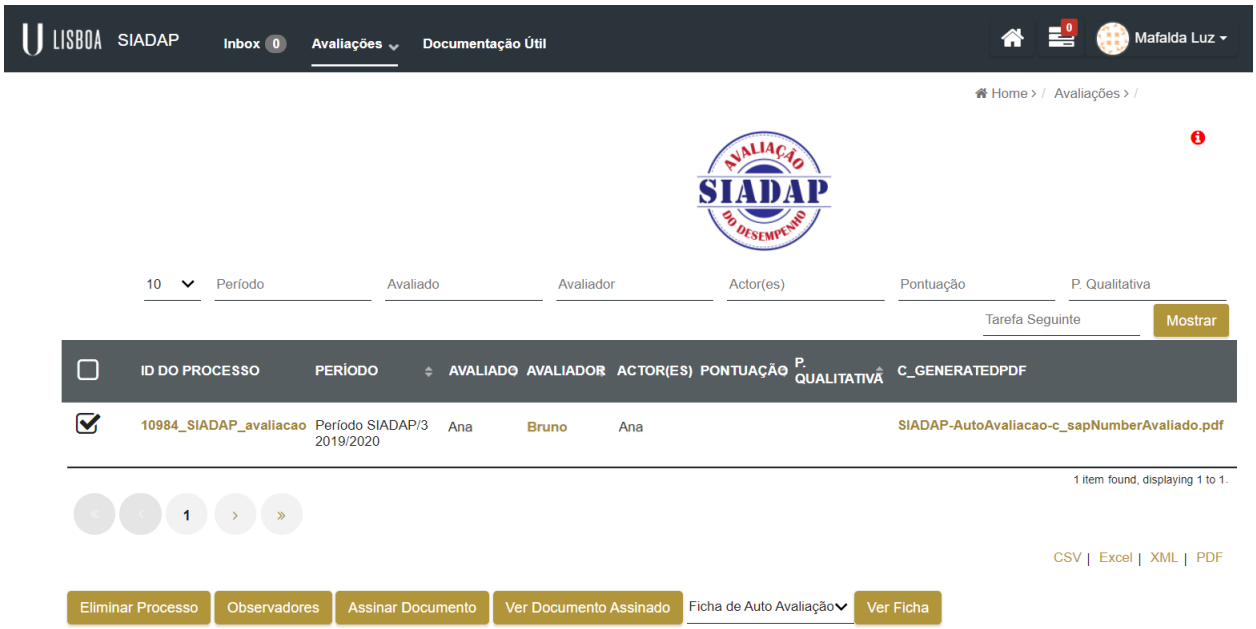

Figura 5.3: Fluxo assinatura da Auto-Avaliação na aplicação SIADAP3 - Ecrã Inicial com documento gerado

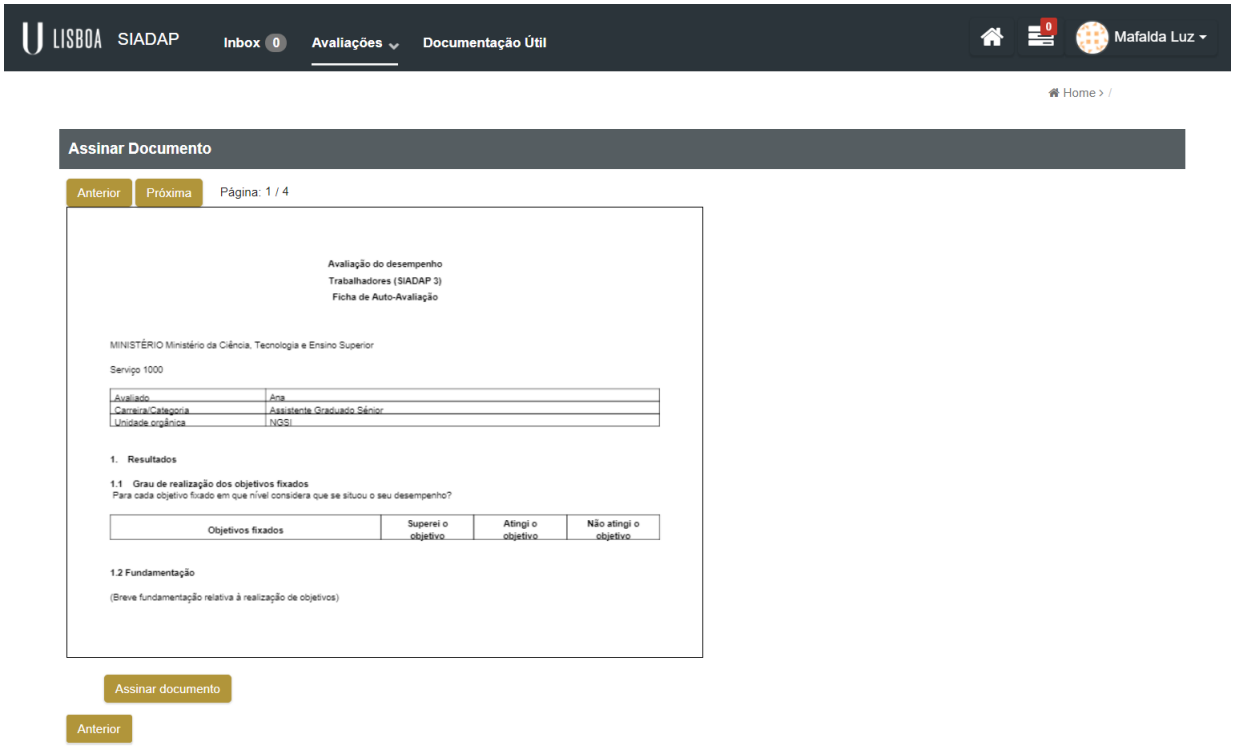

Figura 5.4: Fluxo assinatura da Auto-Avaliação na aplicação SIADAP3 - Ecrã Assinatura parte 1 de 4

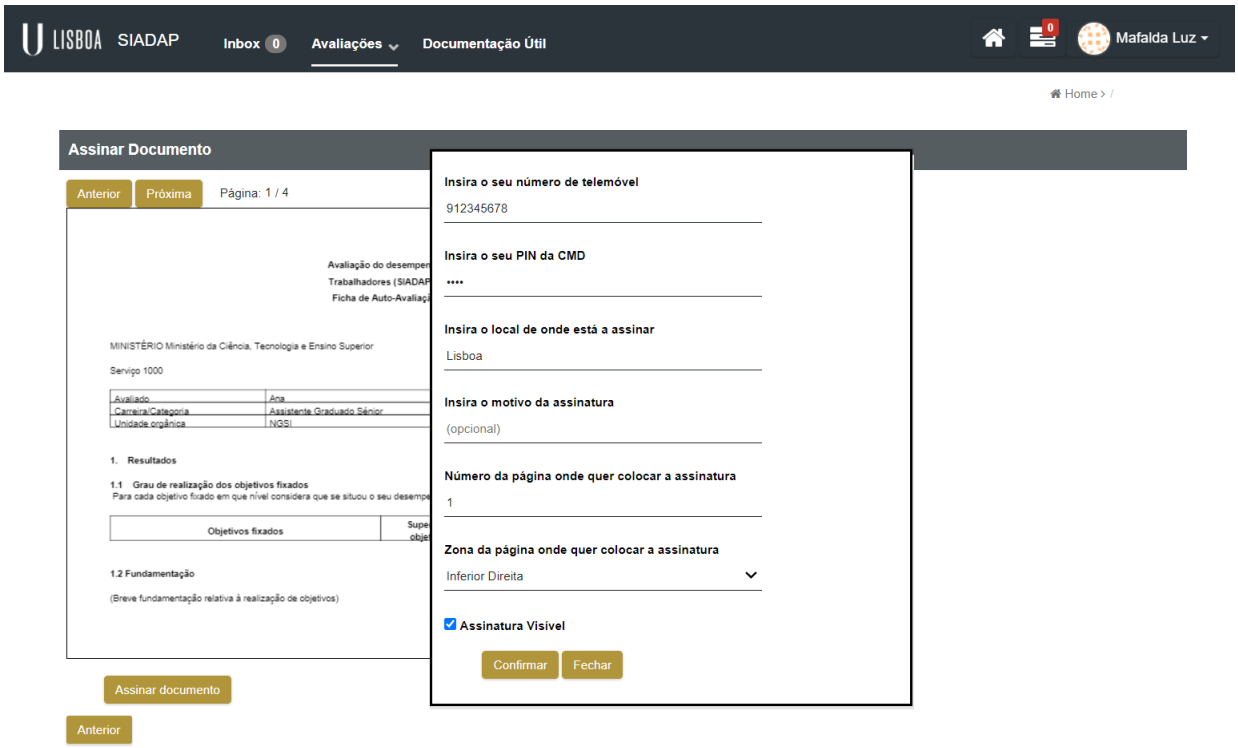

Figura 5.5: Fluxo assinatura da Auto-Avaliação na aplicação SIADAP3 - Ecrã Assinatura parte 2 de 4

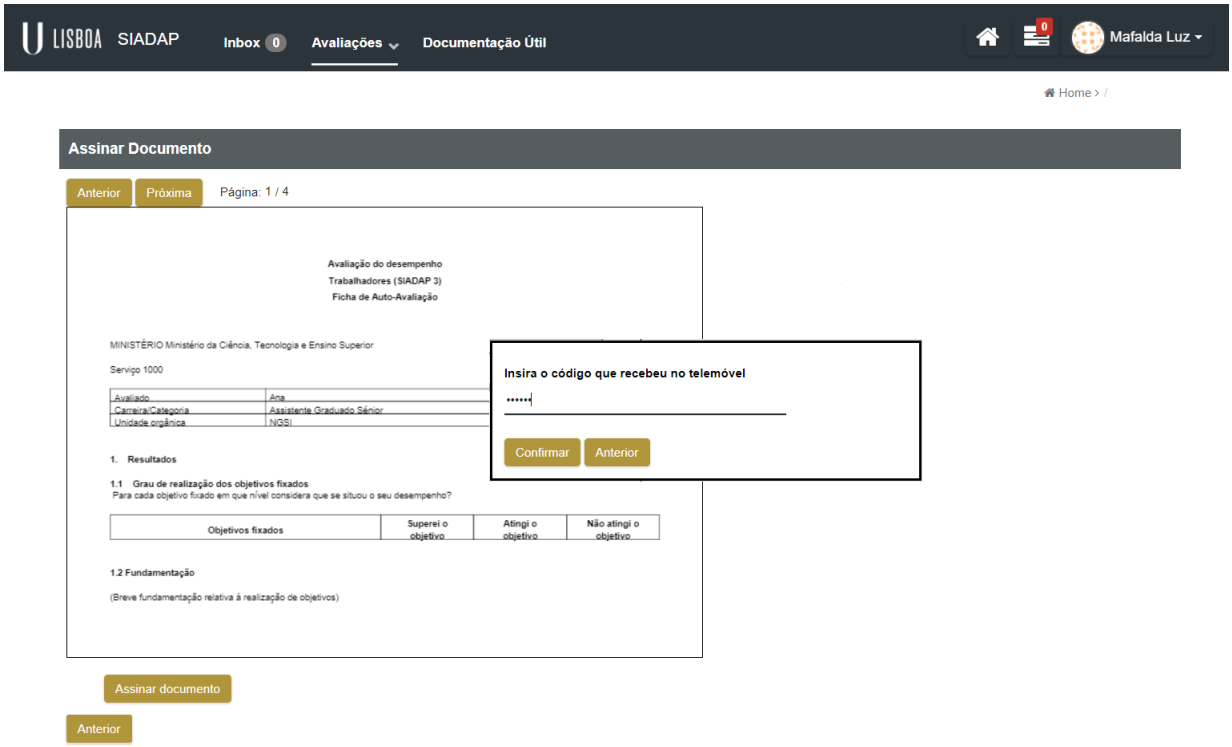

Figura 5.6: Fluxo assinatura da Auto-Avaliação na aplicação SIADAP3 - Ecrã Assinatura parte 3 de 4

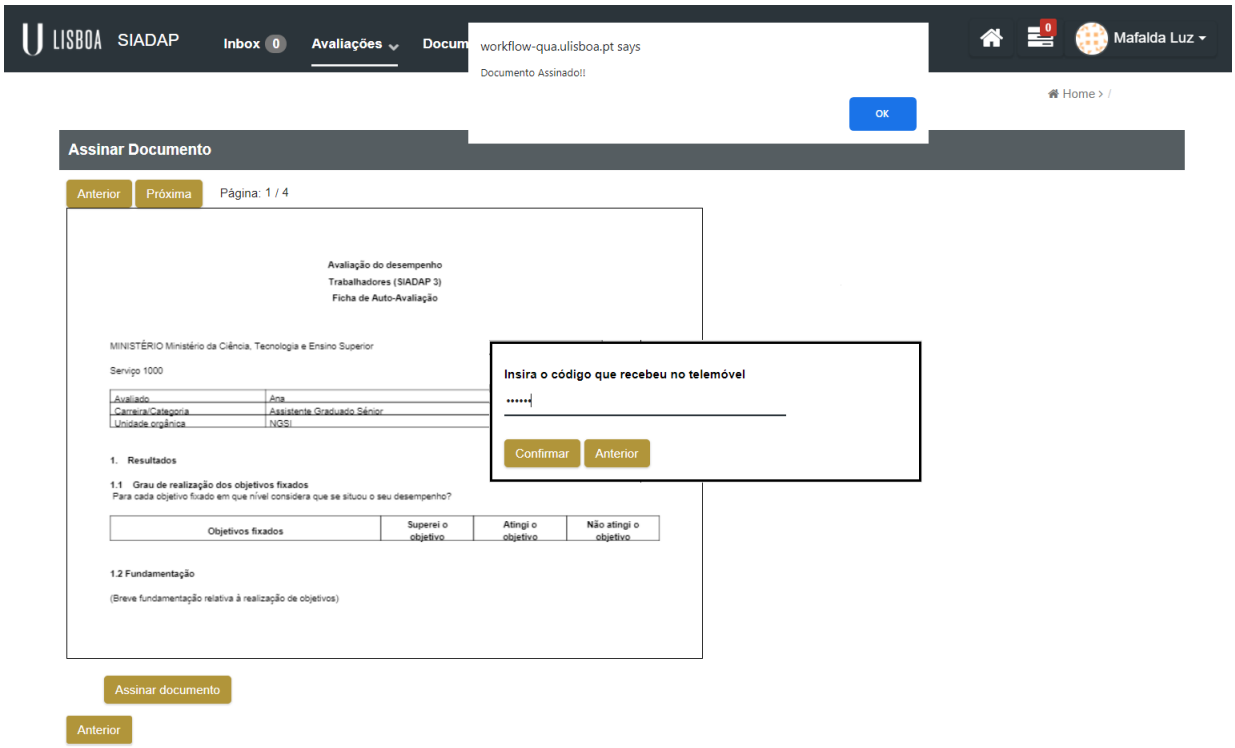

Figura 5.7: Fluxo assinatura da Auto-Avaliação na aplicação SIADAP3 - Ecrã Assinatura parte 4 de 4

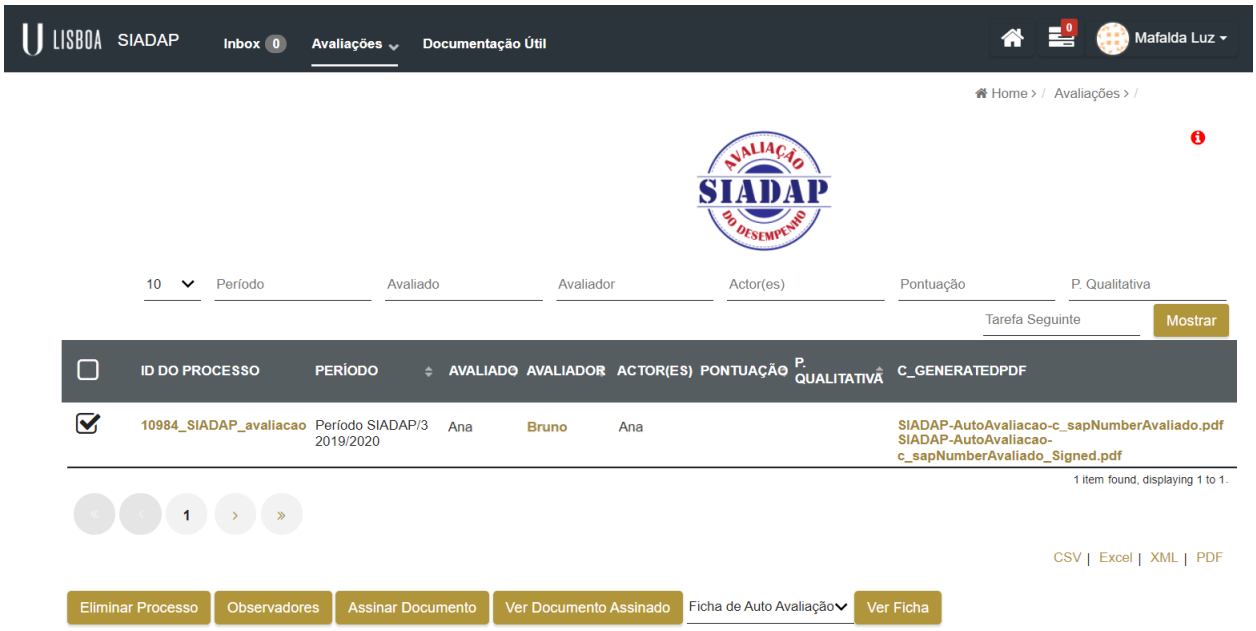

Figura 5.8: Fluxo assinatura da Auto-Avaliação na aplicação SIADAP3 - Ecrã Inicial com documento assinado

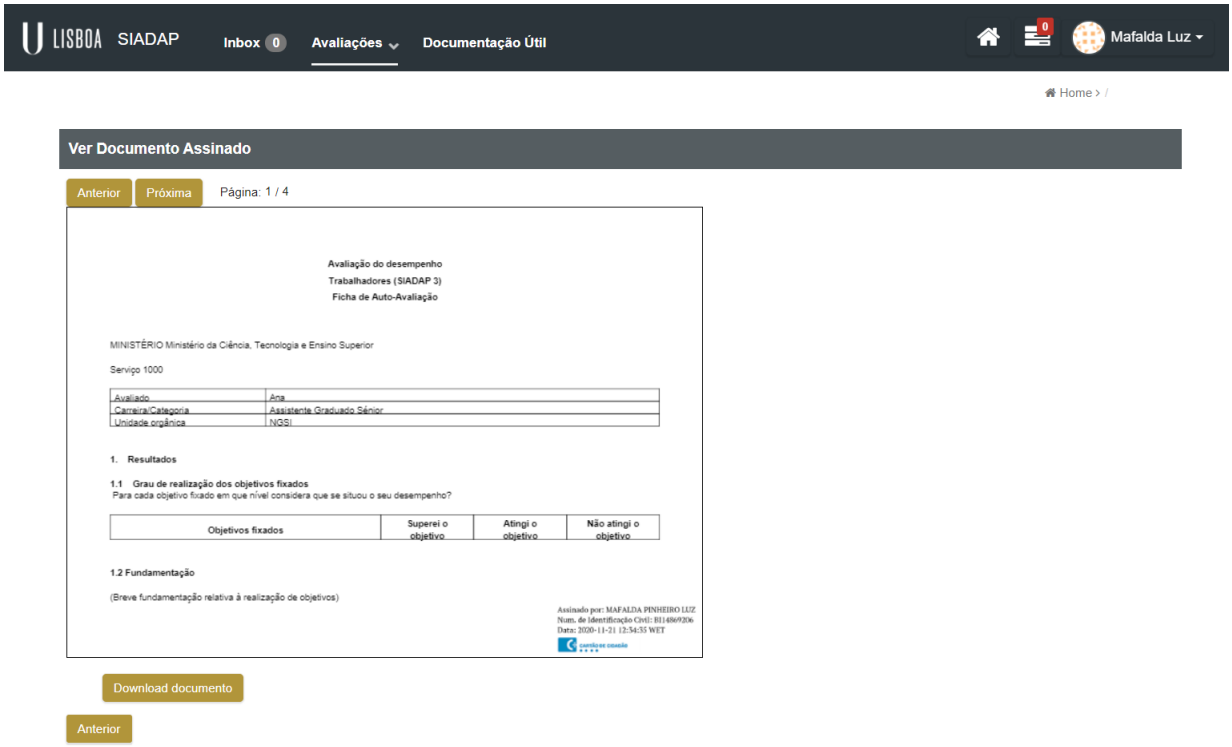

Figura 5.9: Fluxo assinatura da Auto-Avaliação na aplicação SIADAP3 - Ecrã visualizar assinatura

5.2 Validação e questionário SUS

A validação do artefacto criado foi feita através de testes de usabilidade. Pediu-se a 10 colaboradores da reitoria da Universidade de Lisboa que na aplicação que está a ser desenvolvida do SIADAP 3, testassem a funcionalidade de assinar os documentos com a CMD e de seguida preenchessem um questionário. Dos dez colaboradores, oito testaram na qualidade de Avaliados e dois na qualidade de Avaliadores. Foram dadas instruções de como criar a CMD de pré-produção e do seu funcionamento. Foram ainda criados os processos de avaliação para os 8 colaboradores.

O questionário feito encontra-se no Anexo A. Neste questionário utilizou-se o SUS, uma escala numérica utilizada para medir a usabilidade. O questionário é composto por 10 perguntas *standard* cada uma com 5 opções de resposta, em que 1 corresponde a "Discordo fortemente"e 5 a "Concordo plenamente". Após serem obtidos os resultados calculou-se a pontuação da seguinte forma:

1. Nas respostas às perguntas ímpares (1,3,5,7 e 9) subtraiu-se 1 à pontuação atribuída na resposta. (pontuação da pergunta = resposta - 1)

2. Nas respostas às perguntas pares (2,4,6,8 e 10) subtraiu-se a 5 a pontuação atribuída na resposta. (pontuação da pergunta = 5 - resposta)

3. Por fim foram somados esses resultados e multiplicados por 2,5.

4. De modo a obter uma pontuação final, foi feita uma média desses resultados de todos os questionários.

O resultado final varia entre 0 e 100. Este resultado é a pontuação obtida na escala de usabilidade, não corresponde a uma percentagem. De acordo com os estudos feitos, ao longo dos mais de 30 anos de utilização desta escala, um resultado acima de 68 é considerado acima da média [14].

Os resultados obtidos neste questionário relativamente ao artefacto desenvolvido encontram-se na tabela seguinte:

Tabela 5.1: Resultados obtidos no questionário

	Q.1.	Q.2.	Q.3.	Q.4.	Q.5.	Q.6.	Q.7.	Q.8.	Q.9.	Q.10.	Comentários (opcional):	SUS raw score	SUS final score
Avaliado 1	4	2	4	2	3	2	4	2	3	2		28	70
Avaliado 2	4	1	5	2	4	1	5	4	4	1		33	82,5
Avaliado 3	3	2	4	1	4	2	4	2	4	1		31	77,5
Avaliado 4	3	2	4	2	4	2	2	3	3	2		25	62,5
Avaliador 1	5	1	5	1	5	2	5	1	4	1	Excelente	38	95
Avaliado 5	5	2	4	1	5	1	4	1	5	1		37	92,5
Avaliado 6	4	3	4	1	4	3	5	1	5	1		33	82,5
Avaliado 7	5	1	5	1	5	1	5	1	5	1		40	100
Avaliado 8	5	2	5	1	4	1	4	1	4	2	Seria mais fácil poder arrastar a assinatura para o sítio do documento para o qual se pretende assinar	35	87,5
Avaliador 2	4	2	5	1	4	2	5	1	4	1		35	87,5
												Média	83,75

O resultado obtido na escala SUS foi de 83,75. Este pode ser considerado um bom resultado. Nos comentários foi ainda deixada a sugestão de implementar uma interface gráfica de modo a ficar ainda mais interativa a escolha da zona onde se coloca a assinatura no documento.

5.3 Sumário

Este capítulo apresenta a prova de conceito e a validação do artefacto. A prova de conceito foi feita através da implementação do artefacto de assinatura na aplicação SIADAP3. Explicou-se como funciona essa aplicação que está a ser desenvolvida pela reitoria da Universidade de Lisboa e foi dado um exemplo de como seria utilizado o artefacto nesta aplicação. Por fim encontra-se uma descrição da validação feita através da escala de usabilidade SUS.

Capítulo 6

Conclusões e trabalho futuro

O Joget Workflow é uma plataforma bastante utilizada pela reitoria da Universidade de Lisboa. Esta plataforma permite a criação de aplicações web e móveis por parte de utilizadores inexperientes em programação.

A Universidade de Lisboa é uma instituição pela qual passam inúmeros documentos que precisam de ser assinados. Foi por este motivo desenvolvido um artefacto que pode ser utilizado em todas as aplicações criadas no joget, que permite assinar documentos PDF com a CMD.

Com este novo artefacto passam a ser poupados bastantes recursos, não se gasta papel, deixa de ser necessário investir em *hardware* para se fazer assinaturas digitais e passa a ser um processo mais rápido para quem tem de assinar documentos.

O artefacto desenvolvido foi implementado na aplicação do SIADAP3 que está a ser desenvolvida na reitoria da Universidade de Lisboa. Com o intuito de se avaliar a usabilidade foi pedido a dez funcionários que testassem a funcionalidade de assinar documentos com a CMD na aplicação e que preenchessem um questionário. Com base no questionário obteve-se uma pontuação de 83,75 na escala SUS. Este é um bom resultado e mostra que os utilizadores conseguiram assinar documentos PDF com facilidade.

Como trabalho futuro prevê-se a mudança do end-point de pré-produção para produção. Para tal será necessário contactar a AMA de modo a que eles possam verificar se a forma como a solução foi implementada cumpre todos os requisitos e possam posteriormente fornecer credenciais para a aplicação.

Ainda como trabalho futuro de forma a complementar o artefacto desenvolvido podem ser adicionadas as funcionalidades de assinar múltiplos documentos de uma vez e adicionar-se uma interface gráfica para quando o utilizador está a escolher a zona do documento em que quer assinar pré-visualizar logo como irá ficar.

Bibliografia

[1] AMA. Condições gerais de utilização do serviço scmd (POL#8). https://www.autenticacao.gov.pt/documents/10179/958335/POL_08_CondGerUtil_signed_signed.pdf/, 2018. (Acedido em 20/11/2019).

[2] AMA. Condições gerais de utilização do serviço scmd (POL#1). https://www.autenticacao.gov.pt/documents/10179/615532/POL%2301.DPO_signed_signed1.pdf/42fafe8f-60b0-4fae-b728-ef8b47d63e4a, 2019. (Acedido em 20/11/2019).

[3] AMA. Política cmd de assinatura qualificada (POL#16). https://www.autenticacao.gov.pt/documents/10179/615532/POL%2316.PolAssQual_signed_signed1.pdf/46c5e79b-d8a2-467c-8bd6-bb1927212ac7, 2019. (Acedido em 20/11/2019).

[4] Hrvoje Brzica, Boris Herceg, and Hrvoje Stančić. Long-term preservation of validity of electronically signed records. https://www.researchgate.net/publication/332752410_Long-term_Preservation_of_Validity_of_Electronically_Signed_Records, 2013. (Acedido em 14/07/2020).

[5] Ministério da Ciência e da Tecnologia. Diário da república n.º 178/1999, 1º suplemento, série i-a de 1999-08-02. https://data.dre.pt/eli/dec-lei/290-d/1999/08/02/p/dre/pt/html. (Acedido em 13/03/2020).

[6] dgaep-Direção-geral da Administração e do Emprego Público. avaliação do desempenho dos trabalhadores (siadap 3). https://www.dgaep.gov.pt/index.cfm?OBJID=866C1963-A5D3-40B3-A20E-3E20FEDD2A47. (Acedido em 21/12/2020).

[7] Adobe Inc. Digital signatures in a pdf. https://www.adobe.com/devnet-docs/acrobatetk/tools/DigSigDC/Acrobat_DigitalSignatures_in_PDF.pdf. (Acedido em 017/01/2019).

[8] Adobe Inc. Pdf. três letras que continuam mudando o mundo. https://acrobat.adobe.com/pt/pt/acrobat/about-adobe-pdf.html. (Acedido em 09/01/2019).

[9] Adobe Inc. Transform business processes with electronic and digital signatures. https://support.office.com/pt-pt/article/Assinaturas-digitais-e-certificados-8186cd15-e7ac-4a16-8597-22bd163e8e96, 2018. (Acedido em 17/01/2019).

[10] Dejan Lukan. Pdf file format: Basic structure. https://resources.infosecinstitute.com/pdf-file-format-basic-structure/#gref. (Acedido em 13/01/2019).

[11] Microsoft. Assinaturas digitais e certificados. https://support.office.com/pt-pt/article/Assinaturas-digitais-e-certificados-8186cd15-e7ac-4a16-8597-22bd163e (Acedido em 09/12/2019).

[12] Owen Ong. Knowledge Base for v6 introduction to plugin architecture. https://dev.joget.org/community/display/KBv6/Introduction+to+Plugin+Architecture, 2018. (Acedido em 05/12/2019).

[13] Burt Kaliski; Jakob Jonsson; Andreas Rusch. Pkcs 1: Rsa cryptography specifications. https://tools.ietf.org/html/rfc8017#page-45, 2016. (Acedido em 16/08/2020).

[14] Jeff Sauro. 5 ways to interpret a sus score. https://measuringu.com/interpret-sus-score/. (Acedido em 4/01/2020).

[15] William Stallings. *Cryptography and network security: principles and practice*. Pearson, 3 edition, 2017.

[16] ULisboa. Sobre nós ulisboa. https://www.ulisboa.pt/sobre-nos#video. (Acedido em 14/08/2020).

[17] Karsten Meyer zu Selhausen. Security of pdf signatures. https://www.pdf-insecurity.org/download/DIGITALVERSION_KMeyerZuSelhausen_SecurityOfPDFSignatures_2018-11-25.pdf. (Acedido em 20/02/2019).

Abreviaturas

AMA Agência para a Modernização Administrativa, I.P

CEF Connecting Europe Facility

CMD Chave Móvel Digital

DRH Direção Recursos Humanos

DSS Digital Signature Service

FTL FreeMarker Template Language

NIC Número de Identificação Civil

OTP One Time Password

PAdES PDF Advanced Electronic Signatures

PDF Portable Document Format

SIADAP Subsistema de Avaliação do Desempenho da Administração Pública

SUS System Usability Scale

Apêndice A

Questionário SUS

*Obrigatório

System Usability Scale

Para cada uma das seguintes afirmações, escolha a opção que melhor descreve as suas reações de hoje à assinatura de documentos feita com a Chave Móvel Digital (CMD) na aplicação do SIADAP

1. Nome do participante (Primeiro e Último) *

_____

2. 1. Gostaria de usar esta funcionalidade frequentemente. *

Marcar apenas uma oval.

	1	2	3	4	5	
Discordo fortemente	◯	◯	◯	◯	◯	Concordo plenamente

3. 2. A funcionalidade é desnecessariamente complexa. *

Marcar apenas uma oval.

	1	2	3	4	5	
Discordo fortemente	◯	◯	◯	◯	◯	Concordo plenamente

4. 3. A funcionalidade é fácil de usar. *

Marcar apenas uma oval.

	1	2	3	4	5	
Discordo fortemente	◯	◯	◯	◯	◯	Concordo plenamente

5. 4. Necessito de ajuda de um técnico para conseguir usar a funcionalidade. *

Marcar apenas uma oval.

	1	2	3	4	5	
Discordo fortemente	◯	◯	◯	◯	◯	Concordo plenamente

6. 5. As várias funções da funcionalidade estão bem integradas. *

Marcar apenas uma oval.

	1	2	3	4	5	
Discordo fortemente	◯	◯	◯	◯	◯	Concordo plenamente

7. 6. A funcionalidade apresenta demasiadas inconsistências. *

Marcar apenas uma oval.

	1	2	3	4	5	
Discordo fortemente	◯	◯	◯	◯	◯	Concordo plenamente

8. 7. As pessoas irão aprender a usar a funcionalidade muito rapidamente. *

Marcar apenas uma oval.

	1	2	3	4	5	
Discordo fortemente	◯	◯	◯	◯	◯	Concordo plenamente

9. 8. Muitas das funcionalidades da assinatura digital não se percebe bem para que servem. *

Marcar apenas uma oval.

	1	2	3	4	5	
Discordo fortemente	◯	◯	◯	◯	◯	Concordo plenamente

10. 9. Senti-me confiante e tranquilo/a a usar a funcionalidade. *

Marcar apenas uma oval.

	1	2	3	4	5	
Discordo fortemente	◯	◯	◯	◯	◯	Concordo plenamente

11. 10. Tive de aprender demasiadas coisas antes de começar a usar a funcionalidade. *

Marcar apenas uma oval.

	1	2	3	4	5	
Discordo fortemente	◯	◯	◯	◯	◯	Concordo plenamente

12. Comentários (opcional):