

UNIVERSIDADE DE LISBOA
FACULDADE DE CIÊNCIAS
DEPARTAMENTO



INTEGRAÇÃO DE SERVIÇOS DE ASSINATURA DIGITAL, COM BASE EM REGRAS DE INTEROPERABILIDADE

Ricardo Filipe Nunes Madeira

Mestrado em Engenharia Informática

Trabalho de Projeto orientado por:
Prof. Doutor Pedro Alexandre de Mourão Antunes

Agradecimentos

Serve esta secção para agradecer às pessoas que tiveram especial importância no desenvolvimento deste projeto.

Primeiramente, quero agradecer à SIAG, empresa onde realizei o projeto, que me forneceu todas as condições para a realização do estágio. Destacam-se o Engenheiro Carlos da Silva Alves, por depositar a sua confiança em mim, no cumprimento de objetivos profissionais e pessoais, assim como o responsável pela área de desenvolvimento da SIAG, o Engenheiro Diogo André Mota dos Reis, cuja apoio e suporte incondicional, só pode ser superado pela sua capacidade de guiar e ensinar a ser um melhor profissional da área.

Quero também agradecer ao Orientador Interno Professor Pedro Alexandre de Mourão Antunes, cuja visão e conhecimento sobre pesquisa e desenvolvimento baseado em *Design Science*, arquitetou a escrita desta tese.

Também, agradeço à minha família, por sustentar, com sacrifícios e paciência, a formação académica e pessoal que levou ao início da minha carreira profissional.

Por último, um agradecimento geral a todos os fóruns que me auxiliaram na realização deste projeto na obtenção de recursos, entre os quais o (inevitável) Stack Overflow e Portugal-a-Programar, e aos colaboradores da AMA e Multicert, responsáveis pelo apoio à integração de serviços, cuja ajuda foi inestimável.

Dedicatória.

Resumo

Num mundo cada vez mais tecnológico, a desburocratização de processos administrativos torna-se imperativa, e o papel das entidades que disponibilizam *web services* que aceleram esta modernização, crucial. Esta modernização da Administração Pública, através de tecnologias de informação, é denominado de e-Gov. Entidades como a AMA e a Multicert contribuíram para o desenvolvimento do e-Gov, através da integração de serviços que facilitam e melhoram o processo de assinatura digital de documentos, por parte de cidadãos e representantes de empresas.

O propósito desta dissertação é explicar como é feita a integração de *web services* de assinatura digital, de diferentes fornecedores, cumprindo as regras de interoperabilidade especificadas para uma correta integração, no contexto de um sistema ERP - sistema composto por diversos módulos em inter-comunicação. O primeiro serviço integrado é a instalação de um SDK, com um conjunto de funcionalidades de acesso ao Cartão de Cidadão, entre as quais a assinatura digital de documentos. O segundo serviço integrado foi o uso de Chave Móvel Digital, utilizando SOAP para a comunicação com a AMA. E o último serviço integrado serve para a assinatura de documentos fiscalmente relevantes, utilizando um selo qualificado certificado pela Multicert, através de REST.

Para a integração destes *web services*, é utilizada a metodologia DSRM, para desenvolvimento e análise do trabalho realizado, com base em princípios, práticas e procedimentos próprios.

Este estudo demonstra quais os passos necessários para uma correta integração de vários *web services* para assinatura de documentos. Também, apresenta uma análise das diversas integrações, com base num conjunto de critérios, como assinatura digital, nível de segurança, desenvolvimento, cliente e servidor.

Palavras-chave: modernização administrativa, web services, REST, SOAP, SDK

Abstract

In an increasingly technological world, the debureaucratization of administrative processes becomes imperative, and the role of entities that make available *web services* that accelerate this modernization is crucial. This modernization of the Public Administration, through information technologies, is called e-Gov. Entities such as AMA and Multicert contribute to the development of e-Gov, through the integration of services that facilitate and improve the process of digital document submission, by citizens and company representatives.

The purpose of this dissertation is to explain how it is possible to integrate digital web services from different providers, complying with the interoperability rules specified for a correct integration, not in the context of an ERP system - a system made up of various modules in inter-communication. The first integrated service is the installation of an SDK, with a set of functionalities for access to the City Card, among which is the digital assignment of documents. The second integrated service was the use of Chave Móvel Digital, using SOAP for communication with AMA. The last integrated service is for the submission of fiscally relevant documents, using a qualified seal certified by Multicert, through REST. For the integration of these *web services*, the DSRM methodology is used, for the development and analysis of the work carried out, based on its own principles, practices and procedures.

This study demonstrates the necessary steps for a correct integration of various *web services* for document submission. It also presents an analysis of the various integrations, based on a set of criteria, such as digital enrollment, security level, development, client and server.

Keywords: administrative modernization, web services, REST, SOAP, SDK

Índice

1	Introdução	1
1.1	Contexto	1
1.2	Definição do problema	2
1.3	Abordagem	2
2	Enquadramento	5
2.1	Enquadramento Conceptual	5
2.1.1	Certificação digital de documentos	5
2.1.2	Cartão de Cidadão	9
2.1.3	Chave Móvel Digital	10
2.1.4	Web services	11
2.2	Enquadramento institucional	17
2.2.1	SIAG	17
2.2.2	AMA	19
2.2.3	Multicert	21
3	Metodologia de trabalho	25
3.1	Design Science	25
3.2	Desenvolvimento baseado em Design Science	25
3.3	Metodologia DSRM	26
4	Trabalho realizado	31
4.1	Iteração 1: Integração de SDK do <i>middleware</i> do Cartão de Cidadão	31
4.1.1	Identificação do Problema	31
4.1.2	Definição de objetivos	31
4.1.3	<i>Design</i> e desenvolvimento	32
4.1.4	Demonstração	40
4.1.5	Avaliação	41
4.1.6	Comunicação	42
4.2	Iteração 2: Integração de serviço Chave Móvel Digital	43
4.2.1	Identificação do Problema	43
4.2.2	Definição de objetivos	43
4.2.3	<i>Design</i> e desenvolvimento	45

4.2.4	Demonstração	54
4.2.5	Avaliação	56
4.2.6	Comunicação	57
4.3	Iteração 3: Integração de Sign'Stash	58
4.3.1	Identificação do Problema	58
4.3.2	Definição de objetivos	58
4.3.3	<i>Design</i> e desenvolvimento	58
4.3.4	Demonstração	69
4.3.5	Avaliação	69
4.3.6	Comunicação	70
5	Resultados	73
6	Considerações finais	79
	Bibliografia	86
	Anexos	89
7.1	Tecnologias	89
7.2	Legislação aplicável da Chave Móvel Digital	94
7.3	Figuras, Listagens e Tabelas	95

Lista de Figuras

2	Enquadramento	5
2.1	Modalidades de assinaturas eletrônicas	6
2.2	Processo de assinatura digital de um documento	8
2.3	Estrutura de uma mensagem SOAP	16
2.4	Arquitetura do sistema	18
2.5	Arquitetura tecnológica baseada em 3 camadas	19
3	Metodologia de trabalho	25
3.1	Modelo de vistas 4+1	27
4	Trabalho realizado	31
	Iteração 1: Integração de SDK do <i>middleware</i> versão 3 do Cartão de Cidadão	31
4.1	Vista geral da arquitetura proposta	33
4.2	Arquitetura técnica da solução: cliente-servidor	34
4.3	Arquitetura técnica do módulo SiagPrimeFaces:JavaServer Faces	35
4.4	Arquitetura técnica do módulo SiagSignature: Servidor Jetty	35
4.5	Diagrama de sequência - Assinatura digital com Cartão de Cidadão	36
4.6	Estrutura SiagPrimeFaces.war	37
4.7	Diagrama de componentes	38
4.8	Diagrama de pacotes	38
4.9	Interface para parametrizar assinatura - Cartão de Cidadão	40
4.10	Interface para inserir PIN de assinatura - Cartão de Cidadão	41
	Iteração 2: Integração de serviço Chave Móvel Digital	43
4.11	Vista geral da arquitetura da solução	45
4.12	Diagrama de caso de uso - Assinatura digital com Chave Móvel Digital	46
4.13	Arquitetura técnica da solução: cliente-servidor	47
4.14	Arquitetura técnica do módulo SiagCMD	48
4.15	Diagrama de sequência - Assinar documento com Chave Móvel Digital	49
4.16	Diagrama de pacotes	51
4.17	Interface para parametrizar assinatura - Chave Móvel Digital	54
4.18	Interface de inserção do OTP - Chave Móvel Digital	55
	Iteração 3: Integração de Sign'Stash	58

4.19	Vista geral da arquitetura da solução	59
4.20	Diagrama de caso de uso - Assinatura digital de faturação eletrónica	60
4.21	Arquitetura técnica da solução: cliente-servidor	61
4.22	Arquitetura técnica do módulo Siag	62
4.23	Diagrama de sequência - Assinar fatura electrónica	63
4.24	Obtenção de certificado SSL/TLS de servidor	65
4.25	Diagrama de pacotes	65
4.26	Diagrama de pacote dto	66
4.27	Interface UI para imprimir fatura - Sign'Stash	69
5	Resultados	73
5.1	Comparação critério Assinatura digital	73
5.2	Comparação critério Nível de Segurança	74
5.3	Comparação critério Desenvolvimento	75
5.4	Comparação critério Cliente	76
5.5	Comparação critério Servidor	77
7	Anexos	89
7.1	Arquitectura JavaServer Faces	90
7.2	Fluxo de autenticação OAuth2	93
7.3	Diagrama de caso de uso - Assinar documento usando Cartão de Cidadão	96
7.4	Interface para parametrizar assinatura - Cartão de Cidadão	99
7.5	Introdução de PIN para assinatura com Chave Móvel Digital	106
7.6	Interface para parametrizar assinatura - Chave Móvel Digital	107
7.7	Interface de inserção do OTP - Chave Móvel Digital	108
7.8	Assinatura protótipo por Chave Móvel Digital	108

Lista de Tabelas

2	Enquadramento	5
2.1	Comparação entre XML e JSON	14
7	Anexos	89
7.1	Cenário de Modificabilidade	95
7.2	Cenário de Portabilidade	95
7.3	Cenário de Usabilidade	96
7.4	Cenário - Integrar SDK do <i>middleware</i>	97
7.5	Cenário - Assinar documento com Cartão de Cidadão	97
7.6	Descrição de componentes	97
7.7	Descrição de pacotes contidos no SiagSignature	98
7.8	Descrição de pacotes contidos no SiagPrimefaces	98
7.9	Cenário de Legalidade	100
7.10	Cenário de Interoperabilidade	100
7.11	Cenário de Usabilidade	101
7.12	Realização caso de uso - Assinar documento com Chave Móvel Digital	101
7.13	Descrição de pacotes contidos no cmd	102
7.14	Operação 1 - GetCertificate	103
7.15	Operação 2 - SCMDSign	104
7.16	Operação 3 - ValidateOtp	105
7.17	Cenário de Interoperabilidade	109
7.18	Realização caso de uso - Assinar fatura eletrónica	109
7.19	Descrição de pacote broker contido em multicert	110
7.20	Descrição de pacote dto contido em multicert	110

Acrónimos

- AMA** Agência para a Modernização Administrativa, I.P. ix, 19, 20, 31–33, 35–37, 39, 41–47, 50–53, 55–57, 73–77
- API** Application Programming Interface. 12, 35, 39, 41, 58, 74
- CMD** Chave Móvel Digital. 10, 11, 20, 43, 44, 50, 56, 74
- DSRM** Design Science Research Methodology. 2, 26, 31
- ERP** Enterprise Resource Planning. 17, 32, 64
- GNS** Gabinete Nacional de Segurança. 22, 70
- HTTP** Hypertext Transfer Protocol. 12, 14–16, 32, 34, 35, 39, 46, 58, 59, 61, 64–69, 74, 76
- JNLP** Java Network Launching Protocol. 42
- JSON** JavaScript Object Notation. 12–14, 16, 34, 38, 40, 58, 61, 65–69, 76
- JWS** Java Web Start. 42, 75
- OTP** One-Time-Password. xi, 43, 46, 50, 53, 55, 75, 76
- REST** Representational State Transfer. 2, 3, 12–14, 21, 58, 61, 65, 74, 75
- RSA** Rivest-Shamir-Adleman. 46, 50, 52, 75
- SDK** Software Development Kit. ix, xi, 2, 20, 31–33, 35–37, 39, 41, 57, 74, 76
- SIAG** Sistemas Integrados de Apoio à Gestão, S.A. 3, 17, 19, 29, 31–34, 37, 38, 43, 45, 46, 50, 51, 56–59, 61, 63, 64, 66, 69, 73–77
- SOAP** Simple Object Access Protocol. xi, 2, 3, 12, 15, 16, 44, 46, 47, 52, 56, 67, 74, 75
- SSL** Secure Sockets Layer. 46, 58, 61, 64, 69
- TLS** Transport Layer Security. 46, 58, 64, 69
- WSDL** Web Services Description Language. 15, 43, 51, 75
- XML** Extensible Markup Language. xiii, 12–16, 37, 43, 46, 52, 56, 69, 70, 73, 74

Capítulo 1

Introdução

1.1 Contexto

Pode-se considerar que desde o século XX, houve duas ondas de inovação de Tecnologias de Informação e Comunicação: A primeira está associada ao desenvolvimento das tecnologias analógicas e da computação de grande porte (caracterizada por computadores de grande porte e linguagens de programação linear). A segunda, relacionada com a revolução digital (associada à microcomputação e a linguagens de programação orientada a objetos) caracterizada pela otimização da transferência de dados por todo o mundo.

Apesar de advirem em alturas diferentes, estas ondas de inovação foram tratadas como uma ferramenta auxiliar à gestão das diversas áreas do Estado, nomeadamente a Administração Pública¹. Esta resistência no uso de novas tecnologias tornou-a mais burocrática, menos tecnológica, menos próxima dos cidadãos, menos eficiente e eficaz.

Recentemente, têm sido aplicadas reformas administrativas pautadas pela substituição dos modelos tradicionais burocráticos para um modelo de Nova Gestão Pública, caracterizado por conceitos como gestão e melhoria de qualidade, descentralização, desregulamentação e eficiência. Este modelo acaba por evoluir para um modelo que abrange os cidadãos, mais tecnológico e inovador, denominado *e-government* (Carvalho, 2020, pág. 9).

O desenvolvimento do *e-government*, ou *e-Gov*, está associado à modernização da Administração Pública pelo uso das Tecnologias de Informação e Comunicação, na melhoria da eficiência dos processos operacionais e administrativos dos governos. O termo *e-Gov* é um termo genérico para os serviços baseados na *web* proporcionados pelas Entidades e serviços da Administração Pública. Com o *e-Gov*, a Administração Pública usa as tecnologias de informação, e em particular, a Internet para suportar as operações do Estado, envolver os cidadãos e prestar serviços públicos. O *e-Gov* permite melhorar os processos da Administração Pública, aumentando a eficiência, a transparência, a disponibilidade de serviços com maior qualidade e melhor acesso à informação, por parte dos cidadãos (Almeida, 2020, pág. 5–17).

Um dos programas-bandeira de *e-government* em Portugal é o SIMPLEX, que nos últimos anos, promoveu diversas alterações significativas com impacto na vida dos indivíduos, entidades e na própria Administração Pública.

¹Pode ser definida como o poder de gestão do Estado, que se manifesta no poder de regulamentar, tributar e fiscalizar, através dos seus órgãos e outras instituições, tendo em vista a prossecução do serviço público

Em 2006, este programa foi lançado para ir de encontro às expectativas e necessidades dos cidadãos. O SIMPLEX integra medidas que facilitam a relação das pessoas, empresas e outras organizações com o Estado. Facilita o acesso entre as partes envolvidas aos serviços públicos, seja por via presencial ou digital, de forma a tratar as diversas necessidades associadas aos eventos de vida. (SIMPLEX, 2022)

1.2 Definição do problema

Para empresas que desenvolvem *software* para Entidades do Estado, o *e-Gov* facilita a interação entre clientes e prestadores de serviços, desburocratizando os processos dos cidadãos, aumentando a interoperabilidade dos sistemas e aumentando a transparência e agilidade dos serviços prestados. Para se manterem competitivas e ágeis no mercado, esta tem de seguir as tendências e práticas de desenvolvimento de *software* mais atuais. Contudo, nem sempre é fácil acompanhar a evolução da tecnologia, nomeadamente quando esta cresce a um ritmo exponencial. Este problema agrava-se quando uma empresa está integrada numa área, como a da Administração Pública, em que só recentemente é que recebeu investimento do Estado para se modernizar, usando *software open-source*². Um dos problemas mais prementes e burocratizados, atualmente, dos utilizadores das Entidades do Estado, é a assinatura digital de documentos.

Um cliente que, por exemplo, queira autorizar um orçamento ou marcação de férias, ou que queira assinar um contrato, tem duas possibilidades: imprime o documento e assina-o, fazendo a sua rubrica no lugar indicado para a assinatura; ou recorre a um *software* externo como o Adobe Acrobat ou Autenticacao.gov, para assinar esse documento. E se o mesmo quiser assinar um documento, com poderes institucionais da entidade para o qual trabalha?

Assim, surge o problema e razão para a realização deste projeto: **Como é que se integra serviços de assinatura de outras entidades, com protocolos e arquiteturas diferentes, de forma ágil e adaptável, em sistemas usados por Instituições do Estado?**

1.3 Abordagem

A proposta deste projeto é a integração de serviços, através de comunicação com SDKs, estilos arquiteturais como REST e protocolos como SOAP, em comunicação com os restantes sistemas, mediado por um serviço de mensagens, denominado Java Message Service. Desta forma, os serviços podem ser integrados de forma ágil e independente dos outros módulos, seguindo princípios de interoperabilidade específicos, para a comunicação dos vários componentes.

O projeto seguiu a metodologia DSRM, que incorpora princípios, práticas e procedimentos, no desenvolvimento de artefactos para resolver um problema.

O problema é complexo e, por isso, envolve o uso de várias tecnologias e perspetivas de resolução do mesmo. A assinatura de um documento pode ser feita de várias formas e, por isso, decidi abordar este problema em várias iterações, cada uma contribuindo com uma perspetiva de solução, com recurso a diversas tecnologias e integrações de arquiteturas.

1. A primeira iteração aborda a resolução da integração de um SDK de um middleware que

²https://www.rtp.pt/noticias/economia/agencia-para-a-modernizacao-administrativa-vai-promover-software-livre_n93948

disponibiliza a um sistema um conjunto de bibliotecas necessárias para aceder às funcionalidades eletrónicas do Cartão de Cidadão.

2. A segunda iteração exemplifica como é possível integrar o serviço da Chave Móvel Digital por meio do protocolo SOAP;
3. A última iteração atesta como é pode ser integrado REST como estilo arquitetural, para a comunicação com uma entidade externa privada, para a assinatura de documentos, através de um selo qualificado.

O projeto foi realizado no âmbito de um estágio, entre os meses de Novembro de 2021 a Agosto de 2022, na empresa SIAG.

Capítulo 2

Enquadramento

2.1 Enquadramento Conceptual

2.1.1 Certificação digital de documentos

Em 1999, com a aprovação do Decreto-Lei n.º 290-D/99, de 2 de Agosto, “dá-se, em Portugal, o primeiro passo no sentido da consagração legal das assinaturas eletrónicas, acolhendo-se, designadamente, as soluções avançadas no quadro da União Europeia, na proposta de diretiva do Parlamento Europeu e do Conselho, relativa a um quadro legal comunitário para as assinaturas eletrónicas. (...) As redes eletrónicas abertas, como a Internet, têm assumido uma importância crescente na vida quotidiana dos cidadãos e dos agentes económicos, proporcionando uma teia de relações comerciais globais. Para aproveitar da melhor forma estas oportunidades, urge criar um ambiente seguro para a autenticação eletrónica. Na realidade, as comunicações e o comércio eletrónicos exigem assinaturas eletrónicas e serviços a elas associados que permitam a autenticação eletrónica dos dados. As assinaturas eletrónicas possibilitam ao utente de dados enviados eletronicamente que verifique a sua origem (autenticação), bem como se os dados foram alterados (integridade). Em matéria de assinatura eletrónica, o presente diploma assenta no modelo tecnológico ora prevalecente: a assinatura digital produzida através de técnicas criptográficas. Como se depreende dos estudos disponíveis sobre tecnologias de assinaturas digitais baseadas na criptografia de chaves públicas, a assinatura digital constitui, neste momento, a técnica mais reconhecida de assinatura eletrónica, apresentando o mais elevado grau de segurança para as trocas de dados em redes abertas. E é esta constatação do estado da tecnologia que tem levado as experiências legislativas estrangeiras a privilegiar esta forma de assinatura eletrónica.”

A certificação digital é um conjunto de técnicas e processos que propiciam maior segurança às comunicações e transações eletrónicas. A certificação digital identifica pessoas e empresas no mundo digital, comprovando a sua identidade, permite aceder a serviços eletrónicos e assinar documentos eletronicamente com a possibilidade de autenticidade e integridade dos dados. Além destas vantagens, a certificação pode ser usada também como garantia de sigilo e privacidade de sites, controlo de acesso a aplicações, assinatura de formulários, identificação de remetentes, assinatura de mensagens e impossibilidade de repúdio¹ (Zunino, 2017, pág. 68).

¹Rejeição de autoria

Tipos de assinatura electrónica

Pelo Decreto-Lei n.º62/2003 de 3 de Abril, é possível contextualizar e compreender os tipos de assinatura digital que existem e o que as caracterizam.

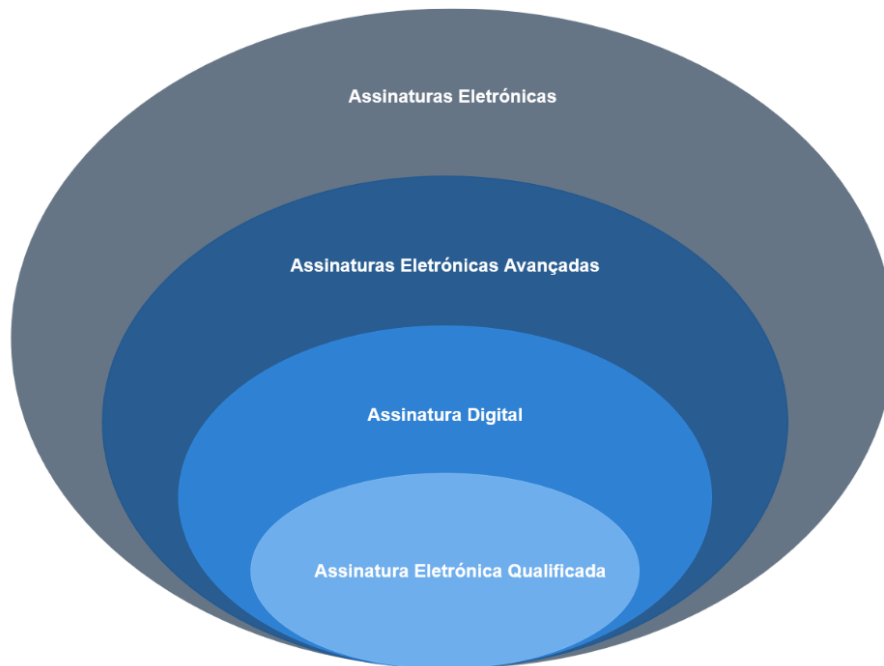


Figura 2.1: Modalidades de assinaturas eletrónicas (Pena, 2020, pág. 7)

De acordo com o Artigo 2º do decreto anteriormente referenciado, a **assinatura eletrónica** é a mais abrangente, que corresponde “ao resultado de um processamento eletrónico de dados suscetível de constituir objecto de direito individual e exclusivo e de ser utilizado para dar a conhecer a autoria de um documento electrónico”. É uma assinatura simples que consiste numa assinatura constituída pelo nome do assinante, como forma de identificação, semelhante a um e-mail.

A **assinatura eletrónica avançada** corresponde à assinatura eletrónica que preenche determinados requisitos, sendo esses requisitos os seguintes:

- “Identifica de forma unívoca o titular como autor do documento”;
- “A sua aposição ao documento depende apenas da vontade do titular”;
- “É criada com meios que o titular pode manter sob seu controlo exclusivo”;
- “A sua conexão com o documento permite detetar toda e qualquer alteração superveniente do conteúdo deste”;

A **assinatura digital** é uma “modalidade de assinatura eletrónica avançada baseada em sistema criptográfico assimétrico composto de um algoritmo ou série de algoritmos, mediante o qual é gerado um par de chaves assimétricas exclusivas e interdependentes, uma das quais privada e outra pública, e que permite ao titular usar a chave privada para declarar a autoria do documento electrónico ao qual a assinatura é aposta e concordância com o seu conteúdo e ao destinatário usar a chave pública para verificar se a assinatura foi criada mediante o uso da correspondente chave privada e se o documento electrónico foi alterado depois de aposta a assinatura”.

Uma **assinatura eletrónica qualificada** é uma assinatura digital ou outro tipo de assinatura eletrónica avançada que satisfaça o requisito de segurança idêntica à da assinatura digital, sendo a assinatura eletrónica qualificada baseada num certificado qualificado e criado através de um dispositivo seguro de criação de assinatura. Aspectos importantes a considerar neste tipo de assinatura são o facto de a mesma ser certificada por uma entidade, o que por sua vez traz uma maior garantia da segurança na utilização pelo seu utilizador ou cidadão, sendo que preserva a autenticidade e veracidade (Pena, 2020, pág. 27).

Criptografia assimétrica

Para a segurança e total funcionamento da assinatura digital, os certificados utilizam criptografia para cifrar e decifrar as assinaturas, sendo utilizado dois tipos de chaves no seu processo: chave pública e privada.

A chave pública pode ser acedida e conhecida por qualquer entidade. É um elemento do par de chaves assimétricas destinado a ser divulgado, com o qual se verifica a assinatura digital aposta no documento eletrónico pelo titular do par de chaves assimétricas, ou se cifra um documento eletrónico a transmitir ao titular do mesmo par de chaves (Ministério da Ciência e da Tecnologia, 1999).

A chave privada é um elemento do par de chaves assimétricas destinado a ser conhecido apenas pelo seu titular, mediante o qual se põe a assinatura digital no documento eletrónico, ou se decifra um documento eletrónico previamente cifrado com a correspondente chave pública (Ministério da Ciência e da Tecnologia, 1999).

Autoridade certificadora

Uma Autoridade Certificadora (AC) é uma entidade, pública ou privada, responsável por emitir, distribuir, renovar, revogar e gerir certificados digitais. Tem a responsabilidade de verificar se o titular do certificado possui a chave privada que corresponde à chave pública que faz parte do certificado e criar e assinar digitalmente o certificado do assinante, em que o certificado emitido pela AC representa a declaração da identidade do titular, que possui um par único de chaves (pública/privada) (Zunino, 2017, pág. 71).

Validade legal de documentos

A aposição de uma assinatura digital a um documento equivale à assinatura autógrafa de um documento em papel e cria a presunção de que:

- A pessoa que põe a assinatura é o titular desta ou é representante, com poderes bastantes, da pessoa coletiva titular da assinatura digital;
- A assinatura foi aposta com a intenção de assinar o documento eletrónico;
- O documento eletrónico não sofreu alteração desde que lhe foi aposta a assinatura digital, sempre que seja utilizada para verificação uma chave pública contida em certificado válido emitido por entidade certificadora.

A assinatura digital deve referir-se inequivocamente a uma só pessoa singular ou coletiva e ao documento ao qual é aposta. Para a aposição de assinatura digital deve utilizar-se uma chave privada cuja correspondente chave pública conste de certificado válido, emitido por entidade certificadora,

credenciada nos termos deste diploma, e que, na data da aposição da assinatura digital não se encontre suspenso ou revogado por decisão da entidade certificadora, e cujo prazo de validade não tenha terminado (Ministério da Ciência e da Tecnologia, 1999).

Validade legal de documentos fiscais

De acordo com o Decreto-Lei Decreto-Lei n.º 28/2019, de 15 de fevereiro, Artigo 12º, "As faturas e demais documentos fiscalmente relevantes podem, mediante aceitação pelo destinatário, ser emitidos por via eletrónica. (...) considera-se garantida a autenticidade da origem e a integridade do conteúdo dos documentos emitidos por via eletrónica se adotado, nomeadamente, um dos seguintes procedimentos: a) Aposição de uma assinatura eletrónica qualificada nos termos legais; b) Aposição de um selo eletrónico qualificado, nos termos do Regulamento (UE) n.º 910/2014, do Parlamento Europeu e do Conselho, de 23 de julho de 2014; c) Utilização de um sistema de intercâmbio eletrónico de dados, desde que os respetivos emitentes e destinatários outorguem um acordo que siga as condições jurídicas do «Acordo tipo EDI europeu», aprovado pela Recomendação n.º 1994/820/CE, da Comissão, de 19 de outubro." Assim, para uma fatura eletrónica ser assinada digitalmente, esta tem de ser assinada por um selo eletrónico qualificado, emitido por uma Autoridade Credenciada.

Processo de assinatura digital

O processo de assinatura digital de um documento consiste num conjunto de passos, que se encontra esquematizado na Figura 2.2:

1. Geração de um "resumo" do documento, utilizando uma função *hash*. É uma função matemática que gera uma mensagem de tamanho pré-definido, calculada com base no documento e têm a propriedade de, para qualquer mudança no documento, gerar mensagens diferentes. A probabilidade de documentos diferentes gerarem mensagens iguais é muito baixa;
2. De seguida, o "resumo" do documento é encriptado usando a chave privada do assinante;
3. o resultado da cifra do "resumo" é associado à mensagem original, produzindo um novo documento: o documento assinado digitalmente;
4. No final, resulta o documento assinado digitalmente, composto pela mensagem inicial e a cifra do "resumo".



Figura 2.2: Processo de assinatura digital de um documento (Subramanya and Yi, 2006)

Comparações com assinatura formal

Num modelo mais tradicional e antiquado, documentos são assinados e reconhecidos num cartório por uma entidade oficial do Estado. Um cartório é responsável pela organização técnica e administrativa com o objetivo de garantir a autenticidade, segurança e eficácia de atos jurídicos. Para uma assinatura ser reconhecida no cartório, é necessário uma firma ter cadastro associado no mesmo. Este reconhecimento é feito através de dois modelos: Por semelhança, em que a assinatura é comparada pela semelhança; Por autenticidade, em que o signatário tem de ir pessoalmente ao cartório assinar o documento (Zunino, 2017, pág. 72).

2.1.2 Cartão de Cidadão

“O Cartão de Cidadão é um documento de cidadania que permite ao cidadão identificar-se de forma segura. Para além de um documento de identificação físico, o Cartão de Cidadão é um documento eletrónico que possibilita a realização de várias operações sem necessidade de interação presencial.” É o documento oficial de cidadania português, desenvolvido no âmbito do programa XVII Governo Constitucional, como projeto de modernização.

A 5 de fevereiro de 2007 foi publicada a Lei nº7/2007, que cria o Cartão de Cidadão e rege a sua emissão e utilização. A criação do Cartão de cidadão conduziu à substituição de cinco outros cartões, são eles: o bilhete de identidade, cartão de eleitor, cartão de contribuinte, cartão de beneficiário da Segurança Social e cartão de utente do Serviço Nacional de Saúde, de forma a evitar a dispersão de suportes físicos sem reduzir os números identificadores afetos a cada cidadão (Pena, 2020, pág. 92).

Características e funcionalidades eletrónicas do Cartão de Cidadão

O Cartão de Cidadão tem um formato *smartcard*. dado que possui um microcomputador embebido também designado de *chip*. Alguma informação como por exemplo, a data e local de emissão do documento, estado civil e o número de eleitor apenas pode ser consultada através do seu chip com a introdução de códigos de acesso (Pena, 2020, pág. 85).

Do ponto de vista eletrónico o *smartcard* do Cartão de Cidadão permite (Pena, 2020, pág. 85):

- Guardar informação pessoal para validação informática interna da identidade do titular;
- Guardar informação privada. Informação privada é informação que o titular pode usar, mas não conhecer ou divulgar. Concretamente, esta informação é constituída por três chaves criptográficas:
 1. Uma chave simétrica de autenticação do titular;
 2. Uma chave privada, que serve para autenticar o titular;
 3. Uma chave privada, que serve para produzir assinaturas digitais do titular;
- Guardar informação reservada. Informação reservada é informação que o titular conhece, mas que apenas disponibiliza de forma fidedigna, via *smartcard*, a quem desejar ou a quem tiver autorização para a obter, independentemente da vontade do titular;

- Guardar informação pública de grande dimensão, não memorizável por humanos. Esta informação é constituída pela fotografia do titular e por certificados X.509v3 de chaves públicas do titular, chaves essas que podem ser usadas para autenticar o titular ou a assinatura digital;
- Guardar toda a informação do titular observável no cartão de cidadão (fotografia, nome, data de nascimento, os diversos números de identificação, validade do cartão, etc. ..).
- Ao titular provar a sua identidade perante terceiros através de autenticação eletrónica. Sendo que, nenhuma autoridade ou entidade pública ou privada pode exigir para efeitos de identificação qualquer outro dado pessoal do titular de cartão de cidadão que não seja facultado pelos respetivos meios eletrónicos;
- Ao titular autenticar de forma unívoca através de uma assinatura eletrónica qualificada a sua qualidade de autor de um documento eletrónico, de onde decorre que nenhuma autoridade ou entidade pública ou privada pode recusar o valor probatório da assinatura eletrónica aposta pelo cidadão num documento eletrónico.

Processo de assinatura de documento com recurso a Cartão de Cidadão

De acordo com o *website*², para assinar documentos com Cartão de Cidadão é necessário:

- Ter a assinatura digital do cartão ativada;
- Leitor de cartões *smartcard*;
- Código PIN de assinatura;

Também inclui também várias funcionalidades avançadas de assinatura, tais como:

- Assinatura de vários documentos ao mesmo tempo
- Assinatura com atributos profissionais;
- Assinatura com opção de Selo Temporal, Motivo ou Localização.

No contexto deste projeto, só a assinatura com motivo e localização é contemplada.

2.1.3 Chave Móvel Digital

Em 2014, a Assembleia da República aprovou³ um meio de autenticação dos cidadãos “alternativo e voluntário”, chamado Chave Móvel Digital.

A CMD surgiu como um meio simples e seguro de autenticação dos cidadãos em portais e *websites* da Administração Pública na Internet.

O primeiro objetivo da Chave Móvel Digital era agilizar a relação das pessoas com os serviços públicos e, assim, permitir que fossem usados online. À medida que este sistema de autenticação foi evoluindo, também se tornou mais abrangente. Atualmente, a lei⁴ diz que a Chave Móvel Digital pode ser usada nos portais online das entidades públicas e também de algumas entidades privadas, assim como assinar documentos.

²<https://www.autenticacao.gov.pt/cartao-cidadao/assinatura-digital>

³Lei n.º 37/2014, de 26 de junho

⁴Diário da República n.º 54/2018, Série I de 2018-03-16

Com a Chave Móvel Digital é possível assinar documentos digitais com a mesma validade de uma assinatura à mão. Sendo que para tal é necessário ter a Chave Móvel Digital ativada, ter a assinatura digital da CMD ativada e código PIN de assinatura da CMD (que pode ser diferente do código PIN da CMD).

De acordo com o Governo da República Portuguesa, existem 200 protocolos assinados, 375 portais e sistemas públicos e privados a recorrer à Chave Móvel Digital como ferramenta de autenticação ou para a realização dos seus serviços, o que representa já um universo de 379 portais e sistemas integrados com a CMD. Também, a Chave Móvel Digital é reconhecida pela União Europeia como um meio de identificação eletrónica com o nível de segurança «Elevado», e que preenche os mais elevados padrões de segurança europeus, tendo recebido ao longo dos anos diversas distinções, nacionais e internacionais (Governo da República Portuguesa, 2021).

Processo de assinatura de documento com recurso a Chave Móvel Digital

De acordo com o *website*⁵, para assinar documentos com Chave Móvel Digital é necessário:

- Ter a Chave Móvel Digital ativada;
- Ter a assinatura digital da CMD ativada;
- Código PIN de assinatura da CMD.

Também inclui também várias funcionalidades avançadas de assinatura, semelhantemente às do Cartão de Cidadão.

No contexto deste projeto, só a assinatura com motivo e localização é contemplada.

2.1.4 Web services

Um *web service* é um contrato estabelecido entre um cliente e um fornecedor. É utilizado para transferir dados através de protocolos de comunicação para diferentes plataformas, independentemente das linguagens de programação utilizadas nessas plataformas. A utilização de *web services* tem vários benefícios tanto a nível tecnológico, como a nível do negócio. Entre os quais se destaca:

Integração de informação e sistemas: uma vez que o funcionamento do *web service* necessita apenas de uma linguagem intermédia que garanta a comunicação entre a linguagem do *web service* e o sistema que faz o pedido ao mesmo, a comunicação entre sistemas e aplicações é bastante simplificada. Com um *web service* é possível trocar informação entre dois sistemas, sem necessidade de recolher informação detalhada sobre o funcionamento de cada sistema. Os *web services* permitem ligar qualquer tipo de sistema, independentemente das plataformas (Windows, Linux, entre outras) e linguagens de programação (Java, Perl, Python, etc.) utilizadas;

Reutilização de código: um *web service* pode ser utilizado por várias plataformas com diferentes objetivos de negócio. O código do *web service* é feito uma vez e pode ser utilizado vezes sem conta por diferentes aplicações;

⁵<https://www.autenticacao.gov.pt/cmd-assinatura>

Redução do tempo de desenvolvimento: é mais rápido desenvolver com *web services*, porque os sistemas não são totalmente construídos a partir do zero e facilmente são incluídas novas funcionalidades;

Maior segurança: o *web service* evita que se comunique diretamente com a base de dados. Assim, a segurança do sistema que fornece os dados está salvaguardada;

Redução de custos: Com a utilização de *web services* não é necessário criar aplicações à medida para a integração de dados, algo que pode ser bastante caro. Os *web services* tiram partido de protocolos e da infraestrutura Web já existente na organização, requerendo por isso pouco investimento.

Numa organização coexistem várias aplicações que organizam e trocam dados, muitas vezes, de formas distintas. Assim, nem sempre garantem a comunicação entre sistemas. Por outro lado, é cada vez mais comum a necessidade de trocar dados entre diferentes sistemas, seja dentro de uma organização ou entre organizações.

É necessário uma linguagem intermédia que garanta a comunicação entre a linguagem do *web service* e o sistema que faz o pedido ao *web service*. Para tal, existem arquiteturas de comunicação como API, estilos arquiteturais como REST (Representational State Transfer) e protocolos como SOAP (Simple Object Access Protocol).

REST

Hoje em dia, são muito poucos os projetos ou aplicações que não usem APIs REST para a criação de serviços. Empresas como Twitter, YouTube, Facebook, utilizam este tipo de APIs para alimentar os seus negócios. Sem a existência de APIs REST, não seria possível as empresas crescerem horizontalmente, através da integração de serviços e funcionalidades, tornando REST um estilo arquitetural muito desejado no desenvolvimento web (devmedia, 2013).

REST é um estilo arquitetural em que uma interface serve de intermediário entre sistemas, e a comunicação seja feita por HTTP para obter e gerar dados, em qualquer formato como XML e JSON. Um *web service* que implemente este padrão é denominado de RESTful e tem como características:

Arquitetura cliente-servidor: REST é composto por clientes, servidores e recursos. Comunica através de HTTP;

Interface uniforme: A interface tem de ser definida uniformemente, de forma a que a informação seja transferida corretamente;

Organização em camadas: As interações entre cliente e servidor podem ser mediadas por camadas adicionais que oferecem recursos como balanceamento de carga, *cache* ou segurança.

Stateless: Não mantém estado entre pedidos. O que significa que toda a informação necessária tem de estar contida no pedido;

Métodos HTTP: A definição do método HTTP indica qual é o tipo de operação a ser realizada sobre um determinado recurso. Pode ser:

- GET:** Consultar;
- POST:** Criar;
- PUT:** Alterar;
- DELETE:** Eliminar;

Como dito anteriormente, o **formato de representação de dados** na comunicação em REST, pode ser de vários tipos, entre os quais se destacam o XML e JSON.

XML é uma linguagem de marcação que define um conjunto de regras para codificação de documentos. Linguagem de marcação é um conjunto de códigos que podem ser aplicados na leitura de dados ou textos feitos por computadores ou pessoas. A linguagem XML fornece uma plataforma para definir elementos de marcação e gerar uma linguagem personalizada (RockContent, 2018).

De seguida encontra-se um exemplo de um conjunto de empregados e os seus respetivos primeiro e último nome, representados em XML.

Listagem 1: Exemplo XML (W3Schools)

```
1 <employees>
2   <employee>
3     <firstName>John</firstName> <lastName>Doe</lastName>
4   </employee>
5   <employee>
6     <firstName>Anna</firstName> <lastName>Smith</lastName>
7   </employee>
8   <employee>
9     <firstName>Peter</firstName> <lastName>Jones</lastName>
10  </employee>
11 </employees>
```

JSON é um formato de representação de dados baseado na linguagem de programação Javascript. É um formato leve e de fácil leitura e escrita por humanos, assim como é de fácil processamento e geração, por parte de máquinas. Assenta em duas estruturas ⁶:

- Uma coleção de pares chave/valor, considerado um objeto;
- Uma lista ordenada de objetos.

A seguinte listagem representa uma coleção de empregados e os seus respetivos primeiro e último nome.

⁶https://www.w3schools.com/js/js_json_xml.asp

Listagem 2: Exemplo JSON (W3Schools)

```

1 {"employees":[
2   { "firstName":"John", "lastName":"Doe" },
3   { "firstName":"Anna", "lastName":"Smith" },
4   { "firstName":"Peter", "lastName":"Jones" }
5 ]}

```

Serve as listagens 1 e 2 de código para comparar os formatos de representação de dados possíveis para um *web service* RESTful. A escolha do formato de dados é dependente dos requisitos do *software*.

Tabela 2.1: Comparação entre XML e JSON (Cloud, 2021)

XML	JSON
Não suporta vetores	Suporta vetores
Tem tags de início e fim	Não usa tags
Documentos são menos compreensíveis	É mais fácil de ler comparativamente a XML
XML consegue encriptar os dados	JSON não tem mecanismos de segurança
Suporta comentários	Não suporta comentários
Suporta vários tipos de codificação	Só suporta a codificação UTF-8

XML e JSON, como formatos de representação de dados, tem as suas vantagens e desvantagens, que determina a sua utilidade, de acordo com os requisitos da aplicação a desenvolver. Para transferir mensagens com informação complexa, XML é a melhor escolha. JSON é mais adequado a aplicações web dinâmicas, de transmissões de dados de quantidades baixas e a velocidade de transmissão deste formato de dados é maior que a do XML, devido à sua simples estrutura.

O estilo arquitectural REST tem vantagens e desvantagens na sua utilização, tais como:

- + REST é fácil e rápido de programar, comparativamente a outros estilos arquitecturais (Como será explicado na próxima secção);
- + Quando o cliente faz o mesmo pedido por diversas vezes, REST utiliza um sistema de armazenamento intermédio, denominado *cache*. Assim, quando o utilizador repete o pedido, primeiramente é feita uma busca à *cache*, à procura do recurso. Caso seja encontrado, não há necessidade de prosseguir com o pedido para o servidor, reduzindo a latência de pedidos;
- + Oferece uma variedade de formatos de dados;
- HTTP não guarda estado da informação que é passada na comunicação entre cliente e servidor. Assim, qualquer gestão de dados tem de ser tratada do lado do cliente;
- REST é uma arquitetura leve, por isso não é adequada para sistemas mais complexos, devido à necessidade de criar múltiplos *endpoints* para lidar com pedidos;
- Por utilizar métodos HTTP, pedidos (nomeadamente o GET) não podem ser utilizados para buscar dados de grandes dimensões.

Os benefícios que REST tem de usar HTTP, também cria restrições no seu desenvolvimento, na medida em que muitas das limitações do protocolo HTTP tornam-se desvantagens de usar o estilo arquitectural REST.

SOAP

Atualmente, existe um grande número de aplicações suportadas por diversas linguagens de programação. A comunicação entre estas aplicações heterogéneas pode ser complicada de gerir, assim como a complexidade do código.

Uma das formas de combater este problema é o uso de XML, que é uma linguagem intermédia usada para a comunicação entre aplicações. Apesar de qualquer linguagem de programação ser capaz de entender XML, não há uma especificação própria de como a usar.

Este protocolo foi desenhado como o objetivo de usar XML e HTTP (como protocolo de transporte) e ter uma especificação de como os usar, em qualquer linguagem de programação (Guru99, 2022).

SOAP tem especificações próprias do protocolo, tais como:

Manter estado: Este protocolo pode manter estado ou ser *stateless*, conforme o tipo de implementação e necessidade de manter estado;

Segurança de serviços web: Padroniza como é que as mensagens são protegidas e transferidas;

Endereçamento de serviços web: Encapsula informações de roteamento, como metadados em cabeçalhos SOAP, em vez de armazená-las em camadas mais a fundo na rede;

WSDL : É uma linguagem baseada em XML, utilizada para descrever *web services* funcionando como um contrato do serviço.

Uma mensagem SOAP é um documento XML estruturado em:

Envelope: Encapsula o conteúdo da mensagem. Contém 2 elementos, o cabeçalho, que pode ser opcional, e o *body*, que é obrigatório;

Cabeçalho : Este elemento contém os dados de reencaminhamento, indicando ao documento XML a qual cliente é que é dirigido o pedido;

Body : Contém a informação para o recetor final;

fault : É um subelemento do *body*, usado para reportar erros.

O diagrama 2.3 mostra a estrutura de uma mensagem SOAP.

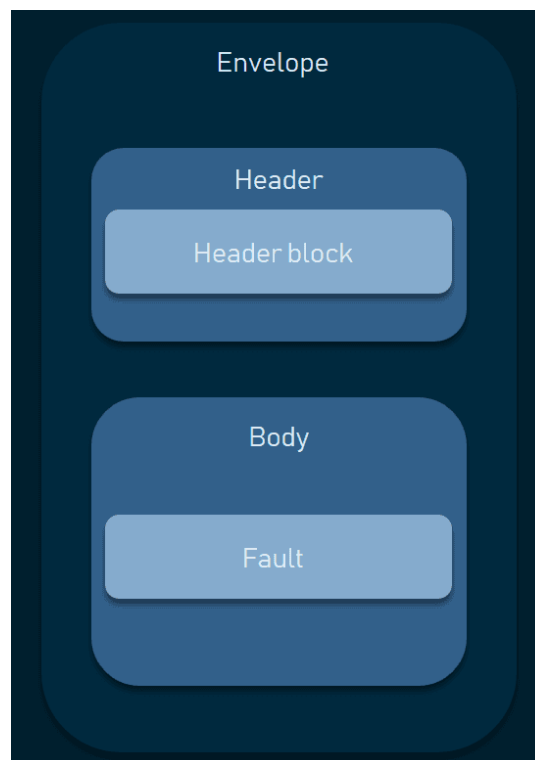


Figura 2.3: Estrutura de uma mensagem SOAP (AltexSoft, 2019)

Este protocolo tem vantagens e desvantagens. Tais como:

- + É independente da plataforma e sistema operativo. SOAP pode ser utilizado sobre diversos protocolos, permitindo a comunicação entre aplicações com diferentes linguagens de programação;
- + SOAP é mais fiável, por ter uma lógica de reentrega de pedido, caso haja algum problema no envio do mesmo;
- + Utiliza HTTP, que é o protocolo mais comum em aplicações web;
- + Devido a configurações muito específicas, a nível de segurança, SOAP tem facilidade de passar *firewalls*;
- O uso deste protocolo torna a comunicação mais lenta. Por usar XML, como forma de representação de dados, que é de fácil leitura por humanos, torna os dados mais complexos e maiores de fazer "parse", comparativamente a outros formatos de representação;
- Apesar de ser flexível, não é tanto como algumas arquiteturas RESTful, que podem utilizar XML, JSON, YAML;
- Mudanças no contrato do lado do servidor, obriga a mudanças do lado do cliente;

2.2 Enquadramento institucional

2.2.1 SIAG

A SIAG é uma empresa privada que desenvolve software para a Administração Pública, há mais de 30 anos. De forma a integrar os diferentes serviços, cada um com as suas responsabilidades, desde gestão de pagamentos, faturação, etc..., segue uma abordagem ERP, em que um só sistema é composto por vários módulos em inter-comunicação. Desta forma, a partir de uma só interface/aplicação, é possível aceder aos recursos e casos de uso de diferentes departamentos de uma empresa. Este estilo de desenvolvimento de software permite a diferentes módulos partilhar e sincronizar dados, expandir negócios e reduzir custos de implementação de diferentes serviços.

Tem como alguns casos de sucesso:

- O SNC-AP, Sistema de Normalização Contabilística para a Administração Pública, que resulta da adaptação das Normas Internacionais de Contabilidade aplicadas ao setor público. A Cinemateca Portuguesa é uma das usufruintes deste sistema moderno⁷.
- A arquitetura flexível e parametrizável do ERP SIAG-AP tem permitido aos organismos que com ele trabalham gerir a mudança para o SNC-AP com um mínimo de esforço, sem surpresas, nem sobressaltos. Desde então, a Universidade de Évora tem vindo a otimizar os seus processos e a consolidar a exploração do sistema SIAG, em diversas vertentes, em especial integrando a gestão académica, a avaliação de desempenho⁸ e a gestão de tempos, com ganhos resultantes muito significativos em termos de eficiência e de eficácia dos serviços⁹.

Sistema e serviços da SIAG

O sistema tecnológico da empresa é composto por diversos módulos de *software*, cada um com as suas responsabilidades e conjunto de operações. Destes, destacam-se:

- **SIAG-ERP** - Base do sistema SIAG. Para além de fornecer as funcionalidades de ERP, também é responsável por gerir e executar algumas operações de baixo nível como o envio de mensagens de correio ou o processamento de operações executadas de forma diferida no sistema;
- **Siag** - Módulo responsável pela integração de serviços externos, persistência de dados, tratamento de dados, entre outros;
- **SiagPrimeFaces** - Módulo responsável pelo *front-end* do sistema. Utiliza a tecnologia JavaServer Faces;
- **SiagSignature** - Conjunto de serviços responsáveis pela assinatura de documentos, utilizando o certificado da SIAG.

A figura 2.4 mostra os módulos-macro que compõem o sistema e como estão organizados.

⁷<https://www.siag.pt/publico/cinemateca-caso-sucesso-completo>

⁸SIADAP é a entidade responsável pela avaliação de serviços, dirigentes e demais trabalhadores

⁹<https://www.siag.pt/publico/uevora-caso-sucesso-completo>

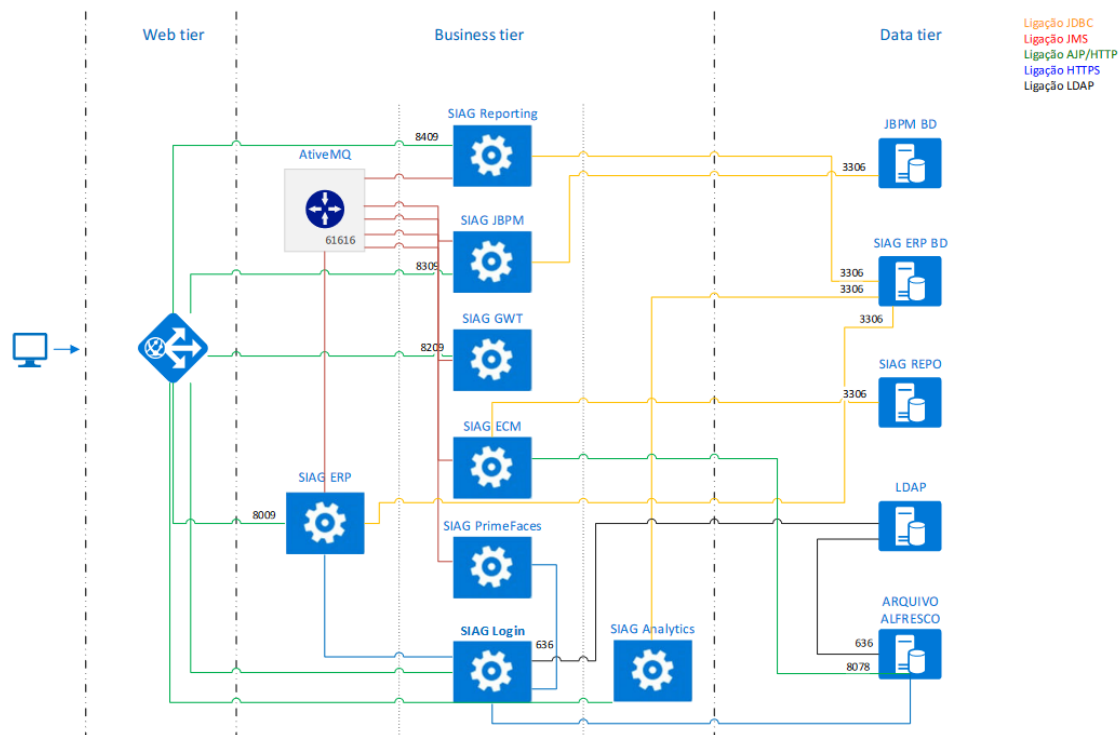


Figura 2.4: Arquitetura do sistema (SIAG, 2020)

Como se pode observar na Figura 2.4, este é um sistema modular baseado numa arquitectura descentralizada em que a comunicação entre os diferentes serviços se realiza principalmente por JMS (Java Message Service). Esta modularidade permite, entre outras vantagens, distribuir os serviços em diferente hardware em função da carga em cada serviço bem como uma parametrização independente da alocação de recursos que cada serviço utiliza. Desta forma é possível nivelar cada serviço à carga expectável e em função das necessidades do momento. A modularidade permite ainda, por exemplo, a utilização de diferentes tecnologias por serviço o que possibilita, entre outras vantagens, a migração faseada por serviço para tecnologias mais recentes, otimizadas para as funções que o serviço realiza. Esta arquitectura é baseada em camadas, em que:

- A **camada web** é usada para controlar e balancear o tráfego de pedidos dos utilizadores com os sistemas aplicativos.
- A **camada aplicacional** com 1 a vários servidores de produção e um de testes.
- A **camada de dados** para dados de produção e dados de teste.

Esta arquitectura encontra-se representada de forma simplificada na figura 2.5.

Na camada aplicacional encontram-se os diferentes serviços desenvolvidos independentes mas que comunicam entre si, predominantemente através de mensagens JMS centralizadas e distribuídas por um *broker* responsável por essa gestão.

JMS é uma especificação que fornece um método comum para programas Java criarem, enviarem, receberem e lerem mensagens. Foi criado para facilitar o desenvolvimento de aplicações empresariais que assincronamente enviam e recebem dados e eventos, e têm dois modelos de comunicação: mensagens em fila de espera (*queue*) e *publish-subscribe*.

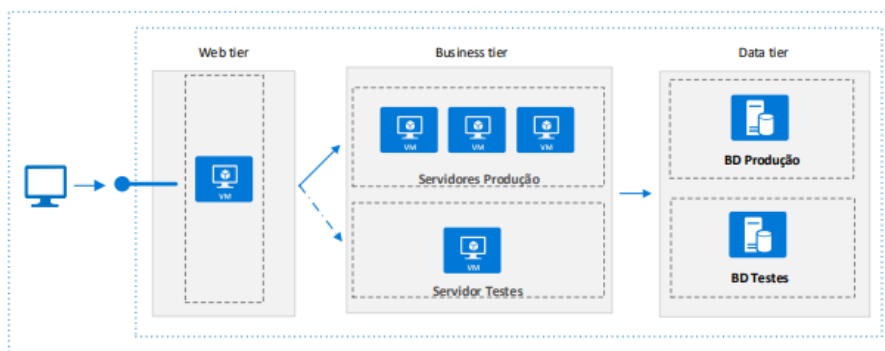


Figura 2.5: Arquitetura tecnológica baseada em 3 camadas (SIAG, 2020)

O *broker* responsável pelas mensagens JMS é o ActiveMQ, que é uma ferramenta para a comunicação entre múltiplos componentes em diferentes servidores ou linguagens de programação. Também garante fiabilidade na comunicação entre produtor/consumidor e escalabilidade horizontal, aumentando o número de serviços, caso o número de mensagens seja demasiado grande. Esta comunicação pode ser observada na figura 2.4.

Deployment do produto da SIAG

Atualmente o produto da SIAG é entregue ao cliente dentro de um ambiente próprio de execução, chamado de *docker container*. É uma plataforma que contém bibliotecas, *software* e regras de *deployment* e segurança muito específicas e necessárias para clientes mais sigilosos. Este *container* contém todas as tecnologias necessárias para a execução do produto da SIAG, de forma a que não seja preciso ir ao cliente instalar uma determinada tecnologia ou versão para a executar. Esta forma de *deployment* de produtos, facilita e automatiza o trabalho de engenheiros de sistemas, assim como isola os acessos ao produto da SIAG, visto que só pode ser feito a partir do ambiente de execução do cliente.

2.2.2 AMA

Em 2007, a AMA foi criada. É um instituto público integrado na administração indireta do Estado. Tem por missão desenvolver, coordenar e avaliar medidas, programas e projetos nas áreas de modernização e simplificação administrativa e regulatória, de administração eletrónica e de distribuição de serviços públicos (ePortugal, 2022). Para isto, produziu os seguintes serviços:

- Uso electrónico do Cartão de Cidadão, que permite:
 - Assinatura digital de documentos;
 - Autenticação em diversas plataformas;
 - Entre outros...
- Chave Móvel Digital
 - Forma simples e segura de autenticação em vários sites públicos e privados, apenas com um número de telemóvel e um PIN de 4 dígitos;
 - Assinatura digital de documentos;
- Integração com a Plataforma de Interoperabilidade;

- É uma plataforma, orientada a serviços, que fornece aos membros da Administração Pública, ferramentas partilhadas para uma melhor comunicação dos mesmos e disponibilização de *web services*;

Serviços da AMA

Uso electrónico do Cartão de Cidadão

Para a integração das funcionalidades electrónicas do Cartão de Cidadão, a AMA desenvolveu um SDK, que consiste num conjunto de bibliotecas utilizadas no acesso às funcionalidades do Cartão de Cidadão. Este SDK foi desenvolvido em C++, sendo suportado por três diferentes sistemas operativos: Windows, Linux e MacOS.

O desenvolvimento de aplicações, que utilizem este SDK, pode ser realizado em C++, Java ou C# (AMA, 2022c).

Este SDK permite a utilização de diversas funcionalidades, tais como:

- Obtenção de dados pessoais do cidadão;
- Verificação e alteração do PIN;
- Assinatura digital de documentos;
- Alteração de morada.

Chave Móvel Digital

A CMD surgiu em 2014 como um meio de autenticação e assinatura digital certificado pelo Estado português. Permite ao utilizador aceder a vários portais públicos ou privados, e assinar documentos digitais, com um único login.

A Chave Móvel Digital associa um número de telemóvel ao número de identificação civil para um cidadão português (AMA, 2021).

A assinatura digital tem a mesma validade legal que uma assinatura à mão, e é válida com qualquer software que lhe permita assinar digitalmente. Possui certificados digitais associados que asseguram a identidade de quem assina um documento digital.

O Estado português garante a certificação de assinaturas digitais realizadas com Cartão de Cidadão ou Chave Móvel Digital (AMA, 2020).

Segundo dados do governo¹⁰, em cada quatro cidadãos utiliza a Chave Móvel Digital de forma efetiva, apesar do número total de cidadãos com CMD ativa ser quatro milhões. O número de utilizadores cresceu de forma gradual desde 2018, tendo um pico de ativações de CMD aquando da pandemia Covid-19.

De acordo com um estudo da Deco Proteste a 1019 cidadãos, 30% dos inquiridos utilizaram a Chave Móvel Digital pela primeira vez para emissão ou renovação de documentos. "Isto diz tudo acerca do papel da pandemia, que foi má em muitos aspetos, mas que impulsionou a utilização dos meios digitais de identificação", segundo Magda Moura Canas, jurista da Deco Proteste.

¹⁰<https://www.dn.pt/sociedade/chave-movel-digital-e-quase-como-mostrar-o-cartao-de-cidadao-a-distancia-15145765.html>

A nível empresarial, as empresas também beneficiaram muito do desenvolvimento deste serviço. A mesma notícia indica que, atualmente já existem, entre públicos e privados, 400 entidades aderentes à Chave Móvel Digital.

De acordo com o Diário de Notícias, a Chave Móvel Digital é resultado do processo de transição digital que tem vindo a desenvolver-se em Portugal, com o objetivo de capacitar os cidadãos, a transformação digital das empresas e do Estado. A jurista da Deco Proteste, no mesmo artigo, considera que "o principal papel da Chave Móvel Digital em todo este projeto será a simplificação e otimização dos processos, a eficiência dos serviços e a diminuição do desperdício de papel, pois são vários os processos que se podem tratar sem qualquer deslocação." Contudo, culpa o governo de ter um papel pouco ativo na divulgação da Chave Móvel Digital e de todas as suas valências no dia a dia dos cidadãos.

Resultado deste estudo, surge a estatística de nos últimos dois anos ter sido evitado cerca de três milhões de idas aos balcões dos serviços disponíveis.

2.2.3 Multicert

A Multicert é uma empresa portuguesa do grupo SIBS especializada em Segurança Digital e de Informação, com uma sólida experiência em sectores como Banca, Sector Financeiro, Saúde, Energia, Administração Pública, e Retalho (Multicert, 2022).

Serviços da Multicert

A Multicert dispõe de vários serviços, porém, no interesse do projeto, há um que se destaca:

Sign'Stash. Este serviço é uma multiplicidade de funcionalidades que permite a integração com sistemas de faturação atuais, envio de faturas eletrónicas, assinatura eletrónica qualificada de documentos e preservação de segurança digital. Tem como características:

- Serviços em formato REST e autenticação OAuth 2 para uma integração simples e rápida;
- Os selos qualificados Multicert oferece garantia de aceitação da sua assinatura qualificada em todo o espaço europeu;
- Assinar documentos em vários formatos digitais, permitindo a utilização noutros fluxos de negócio;
- Entre outras...

Conformidade Legal

A Multicert uma entidade credenciada pelo GNS e integrada na *Trusted List*¹¹. De acordo com o própria, "Os Selos Temporais emitidos pela Multicert estão em conformidade com a legislação, nomeadamente, com os seguintes regulamentos e decretos-lei: Regulamento (UE) n° 910/2014 do Parlamento e do Conselho Europeu - define o requisito dos Estados Membros da UE serem obrigados a manter uma Lista de Confiança com as CAs qualificadas Regulamento de Execução (UE) 2015/1501 da Comissão - define o formato e dados a constar na lista de confiança Decreto-Lei n° 88/2009 - nomeação do organismo em Portugal responsável por manter, atualizar e disponibilizar a Lista de Confiança em Portugal."

¹¹Lista de entidades de confiança, emitida pelo GNS de Portugal

Capítulo 3

Metodologia de trabalho

3.1 Design Science

As ciências naturais, como a biologia ou geologia, descrevem e ensinam como é que os fenómenos naturais ocorrem e interagem com o mundo. Enquanto que as ciências naturais abordam este objetivo através do desenvolvimento e justificação de teorias que expliquem ou prevejam um determinado evento, a *Design Science* aborda este objetivo através da construção de um artefacto inovador que se proponha a resolver um problema (Antunes et al., 2022).

Essencialmente, *Design Science* é um paradigma de resolução de problemas cujo objetivo é produzir um artefacto que deve ser implementado e, posteriormente, avaliado.

3.2 Desenvolvimento baseado em Design Science

O desenvolvimento baseado em *Design Science* é uma abordagem recente, com enfoque na concepção de artefactos. Esta abordagem tem duplo objetivo: desenvolver um artefacto para resolver um problema identificado num determinado contexto; gerar novos conhecimentos técnicos e científicos (Hevner et al., 2004, pág. 84–85). Assim, deve resolver o problema observado, contribuir para futuros desenvolvimentos, avaliar o *design* de uma solução e comunicar os resultados ao público relevante. O conhecimento gerado a partir de um desenvolvimento baseada em *Design Science* informa como um artefacto pode ser melhorado, se é melhor que a atual solução e como é que resolve o problema de forma mais eficiente.

A condução de desenvolvimento baseado em *Design Science* requer correto planeamento e execução que pode ser executado em várias questões. Apesar de haver vários aspetos para o sucesso da mesma, a escolha da metodologia é das decisões mais complicadas que um investigador tem, na medida em que a metodologia dita o tipo de abordagens a tomar, como recolher informação e como chegar ao objetivo do estudo (Thuan et al., 2019).

3.3 Metodologia DSRM

Há diferentes metodologias para se conduzir desenvolvimento em *Design Science*.

A metodologia escolhida para o presente desenvolvimento é a *Design Science Research Methodology*, ou DSRM, que incorpora princípios, práticas e procedimentos para realizar desenvolvimento em *Design Science*. Esta metodologia, de acordo com (Venable et al., 2017), adequa-se melhor a estudos que tem como objetivo o desenvolvimento de um artefacto que resolva o problema proposto e a comunicação do mesmo ao público relacionado.

Este processo é iterativo e sequencial, contudo pode ser iniciado em qualquer etapa da metodologia, conforme a necessidade dos pesquisadores e o contexto da desenvolvimento.

Passo 1: Identificação do problema

Esta etapa é dedicada à definição do problema, assim como a motivação para o desenvolvimento a realizar. Este passo da metodologia serve também para justificar o valor da solução para o atual problema, motivando o público do desenvolvimento a seguir o processo de construção da solução.

Passo 2: Definição de objetivos

Nesta etapa, os objetivos da solução são inferidos a partir do passo anterior, e se são possíveis dentro do contexto do problema. Os objetivos podem ser quantitativos ou qualitativos.

Passo 3: *Design* e desenvolvimento

O propósito deste passo da metodologia DSRM é fornecer uma arquitetura detalhada da solução proposta, com base nos atributos definidos anteriormente. Cada um destes atributos vai ser descrito através de um conjunto de cenários, seguido de uma vista geral da arquitetura do sistema, que inclui várias perspectivas de análise do mesmo e a descrição dos padrões de desenho usados para o desenvolvimento da solução.

De forma a documentar a arquitetura das soluções presentes em cada iteração da solução, foram realizadas vistas técnicas da mesma. A utilização de vistas para explicar em detalhe uma arquitetura, é altamente recomendada por arquitetos de *software*, por trazer um conjunto de benefícios à definição da mesma, como por exemplo:

Separação de responsabilidades - Separar diferentes modelos de um sistema em descrições distintas, mas relacionadas, ajuda no *design*, análise e comunicação das diferentes partes do sistema, focando os *stakeholders* em cada aspeto separadamente;

Gestão de complexidade - Gerir simultaneamente todos os aspetos de um sistema complexo pode resultar numa sobrecarga de trabalho, que uma só pessoa não consegue lidar. Tratando cada aspeto de forma significativa e independente, o arquiteto pode focar a sua atenção em cada aspeto à vez, gerindo a complexidade global do sistema, de forma faseada;

Equipa de desenvolvimento melhor focada - A definição de uma arquitetura de um sistema, através de vistas, é particularmente importante para a equipa de desenvolvimento por as usarem como

fundação para o *design* do sistema. Ao separarem os vários aspetos de um sistema em diferentes vistas, é assegurada a devida construção do mesmo;

Melhor comunicação com os *stakeholders* - A separação de responsabilidades e comunicação entre *stakeholders* pode ser bastante difícil de gerir. Uma vista orientada a determinados aspetos de um sistema podem guiar, de forma fácil e rápida, *stakeholders* para a gestão das suas responsabilidades, e cada vista pode ser utilizada usando linguagem e notação apropriada ao nível de conhecimento e experiência do leitor da mesma.

É recomendado, e conveniente, o uso do modelo de vistas 4+1 para a documentação da arquitetura de um sistema (Kruchten, 1995). Esta forma de visualizar a arquitetura de sistemas de *software* é baseado em 5 vistas/perspetivas de um sistema.

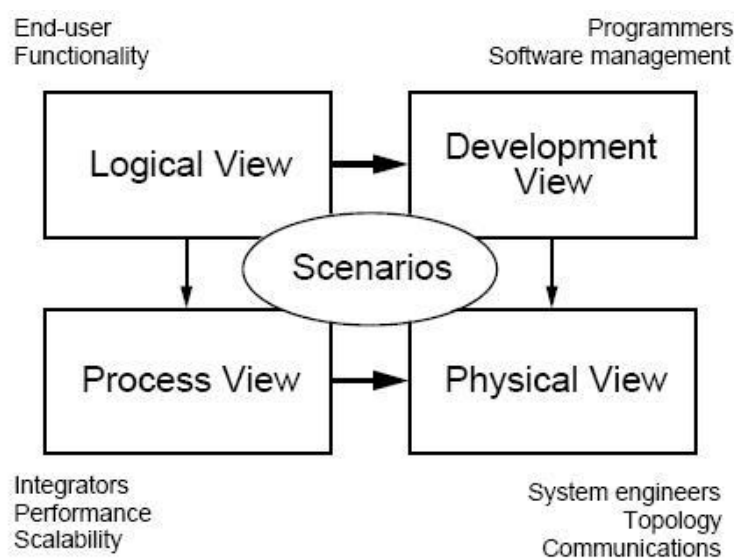


Figura 3.1: Modelo de vistas 4+1 (Kruchten, 1995)

Cenários

Este tipo de vista tem como objetivo capturar as funcionalidades oferecidas aos utilizadores e como é que é a interação entre o sistema e os atores do mesmo, através da apresentação de cenários.

Visão lógica

Este tipo de vista suporta principalmente os requisitos funcionais - O que é que o sistema deve fornecer em termos de serviços aos seus utilizadores. O sistema é decomposto num conjunto de abstrações, retiradas do domínio do problema, na forma de objetos ou classes de objetos. Esta vista explora os princípios de abstração, encapsulamento e herança. Em suma, esta vista descreve como é que o sistema é estruturado em termos de unidades de implementação, como pacotes, classes e *interfaces* e a relação entre elementos, como dependências entre os mesmos. Esta vista é de grande importância para *stakeholders* como o utilizador, cujo interesse é no comportamento do sistema.

Esta vista é obrigatória no Modelo 4+1.

Visão de processo

Este tipo de vista tem em conta requisitos não-funcionais, como *performance* e disponibilidade. Aborda questões como concorrência, distribuição e tolerância de faltas. Esta vista pode ser descrita em vários níveis de abstração, em que cada um trata das suas responsabilidades. Ao nível mais alto, a arquitetura pode ser vista como um conjunto de processos independentes, distribuídos pelo sistema.

Este tipo de vista descreve a estrutura cliente-servidor de um sistema. Enquanto que a Visão lógica descreve como é que um sistema é estruturado em unidades ou módulos, a Vista de processo mostra como é que um sistema é estruturado como um conjunto de elementos em execução e como é que é feita a comunicação dos mesmos.

Stakeholders como "integradores", membros da equipa de desenvolvimento com especial interesse na integração de sistemas, encontram utilidade neste tipo de vistas.

Esta vista é opcional no Modelo 4+1 e pode ser representada através de diagramas de sequência, atividade, entre outros...

Visão física

A arquitectura física tem em conta, principalmente, requisitos não-funcionais como disponibilidade, fiabilidade, *performance*, entre outros... Este tipo de vista retrata a topologia dos componentes de *software*, assim como as ligações físicas entre os mesmos. Tendo em conta que esta vista retrata a disposição dos componentes de *software* do sistema, engenheiros de sistema tem particular interesse nas representações para esta vista. Esta vista é opcional no Modelo 4+1 e pode ser representada através do diagrama de implementação, ou "deployment diagram".

Visão de desenvolvimento

A arquitetura de desenvolvimento tem como foco a organização dos módulos de *software*, no ambiente de desenvolvimento de *software*. As unidades são armazenadas em pequenas bibliotecas ou subsistemas que podem ser desenvolvidos por uma equipa de desenvolvimento. Cada subsistema é organizado numa hierarquia de camadas, cuja comunicação é feita através da definição de *interfaces*.

A vista de desenvolvimento serve como base para alocação de recursos, como esforço e membros da equipa de desenvolvimento, avaliação e planeamento de custos, monitorização do progresso do projeto, reutilização de *software*, portabilidade e segurança. Esta vista é opcional no Modelo 4+1 e pode ser representada através de diagramas de pacotes e componentes.

Passo 4: Demonstração

Nesta etapa, é feita uma demonstração da solução.

Passo 5: Avaliação

Nesta atividade, é feita uma observação e análise da solução. É realizada uma comparação entre os objetivos da solução (Passo 2), e o resultado do uso do artefacto no passo anterior. Dependendo do contexto do problema e do artefacto, várias métricas podem ser usadas para efetuar a avaliação. Podem ser utilizadas métricas quantitativas, como *performance* do sistema, orçamento, inquéritos de satisfação, entre outras... No final desta atividade, os pesquisadores decidem se reiteram o passo 3 para melhorar a solução ou se continuam para a fase seguinte.

No contexto deste projeto, todas as iterações foram alvo de avaliação funcional, na medida em que foram desenvolvidas no âmbito do contexto empresarial. Assim, todo e qualquer artefacto que advém da iteração realizada, foi sujeito a uma avaliação criteriosa e prática, para poder ser integrada no sistema da SIAG.

Passo 6: Comunicação

No final do ciclo, é comunicado o problema e a sua importância, a utilidade e inovação do artefacto, o rigor do *design* da solução, e a sua eficácia, no contexto do problema.

Capítulo 4

Trabalho realizado

A abordagem tomada para a resolução do problema identificado, decorre da identificação de três possíveis integrações de serviços. Para isto, cada uma destas iterações é analisada de acordo com a metodologia DSRM, em que são apresentados os princípios, práticas e procedimentos para o desenvolvimento do artefacto resultante da iteração observada.

4.1 Iteração 1: Integração de SDK do *middleware* do Cartão de Cidadão

4.1.1 Identificação do Problema

Como dito anteriormente, a SIAG desenvolve *software* para entidades da Administração Pública e para os seus respetivos departamentos, como faturação, gestão de recursos humanos, aprovação de orçamentos, etc...

Este problema foi identificado a partir de casos de uso como aprovação de marcação de férias, aprovação de gastos e orçamentos, em que os utilizadores necessitavam de imprimir os documentos e assiná-los fisicamente, ou utilizar aplicações externas, como o Autenticacao.gov¹, para os poder assinar e assim, atribuir validade legal aos mesmos.

4.1.2 Definição de objetivos

Com base na necessidade de uma forma de assinar digitalmente documentos, a SIAG tem restrições quanto ao tipo de *software* a integrar no seu sistema. Algumas destas são:

- Compatibilidade com a linguagem de programação Java 8;
- Ser *software open-source*, de preferência;
- Utilizador assinar através do Cartão de Cidadão;
- Assinar documentos, de acordo com a lei portuguesa.

Segundo estes requisitos, surge o SDK do *middleware* desenvolvido pela AMA.

Este *software* é compatível com linguagens de programação como C++, Java e C#, e sistemas operativos como Windows, Linux e Apple MacOS. Também é atualizado com bastante regularidade² e é flexível quanto à estilização visual da assinatura.

¹A aplicação Autenticacao.gov para computador permite ao cidadão tirar partido das funcionalidades eletrónicas do seu Cartão de Cidadão ou Chave Móvel Digital, desenvolvido por AMA

²<https://github.com/amagovpt/autenticacao.gov/commits/master/pteid-mw-pt>

Assim, a solução proposta é a integração do SDK do *middleware* do Cartão de Cidadão, que permite a aplicações incluir funcionalidades de acesso às utilidades eletrónicas do Cartão de Cidadão. Para além de permitir a assinatura digital de documentos, também oferece uma panóplia de funcionalidades extra, que podem ser utilizadas para autenticação de um utilizador, possibilitando assim ao sistema evoluir, caso seja necessário. Com base nesta diversidade de funcionalidades, deseja-se que a solução permita a criação de *endpoints*, para que cada pedido realizado seja reencaminhado para o *endpoint* responsável pela lógica e processamento do mesmo.

De forma a poder avaliar a viabilidade da solução, são definidos três requisitos não-funcionais:

- **Modificabilidade** - Deve-se à possibilidade de utilização do SDK, desenvolvido pela AMA, para a integração de várias funcionalidades, tais como uso dos dados do Cartão de Cidadão para poder realizar autenticação, mudança de morada, entre outros. Tendo em conta este conjunto de funcionalidades que a biblioteca oferece, deseja-se um sistema flexível e facilmente modificável;
- **Portabilidade** - O produto da SIAG pode ser executado em qualquer sistema operativo e aparelho. Assim, a solução tem de ser capaz funcionar de forma correta, independentemente do ambiente de execução em que se encontre;
- **Usabilidade** - Tornar a *interface* com o utilizador focada na experiência do mesmo, de forma a prevenir erros e ser satisfatória para o utilizador.

4.1.3 Design e desenvolvimento

Atributos qualitativos

As tabelas 7.1, 7.2 e 7.3 descrevem cenários concretos para os três atributos de qualidade que devem estar incluídos na solução.

Vista geral da arquitetura

Para se poder integrar o SDK do *middleware* desenvolvido pela AMA no produto da SIAG, é necessária a execução de um servidor local a correr no ambiente de execução do utilizador, tendo em conta que este *software* tem de comunicar com um leitor de Cartão de Cidadão, para poder assinar um documento. Assim, muito semelhante a aplicações como o Portal das Finanças, que na autenticação por Cartão de Cidadão, exigem a execução do *plugin* Autenticação.Gov, o produto da SIAG permitirá ao utilizador correr um servidor local, neste caso Jetty, que será intermediário entre as funcionalidades oferecidas pelo SIAG ERP e os serviços desenvolvidos pela AMA, contidos no SDK.

O diagrama 4.1 representa os principais componentes da solução proposta no sistema geral e as respetivas ligações. Seguindo a arquitetura cliente-servidor, o lado-cliente é o utilizador a realizar o pedido de assinatura através do produto da SIAG, e o lado-servidor é

A escolha pela tecnologia Jetty deve-se ao facto de ser um servidor HTTP leve, com baixa utilização de memória e que se destaca pela sua escalabilidade, permitindo às equipas de desenvolvimento de *software* exponenciar as suas aplicações, sem alterar a sua arquitetura.

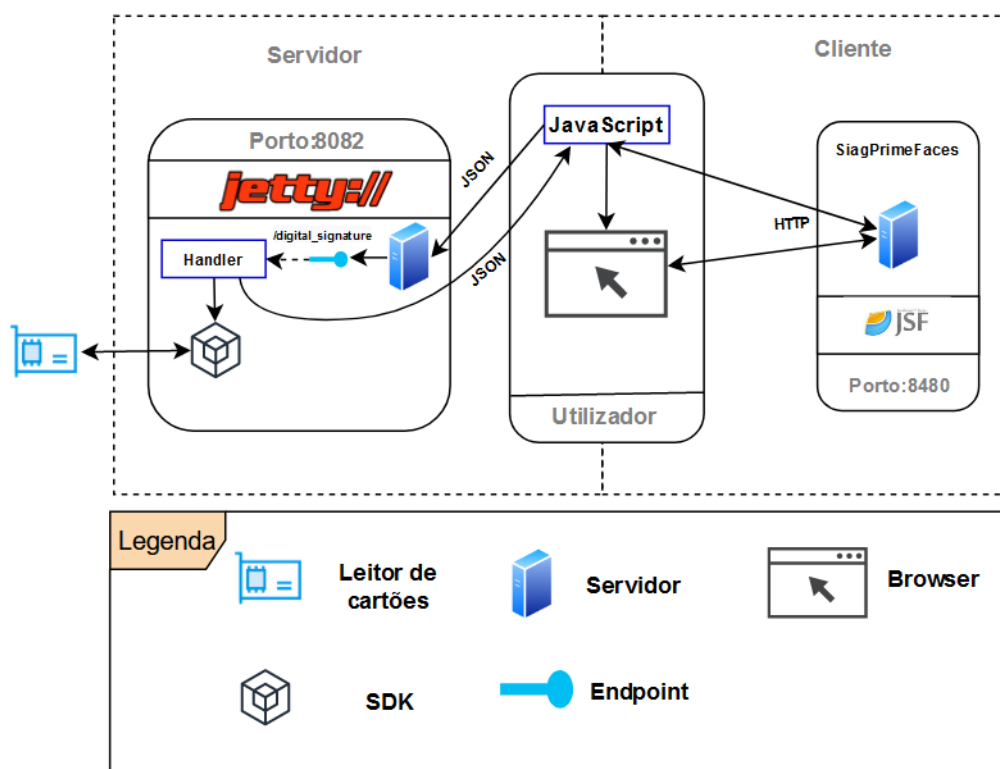


Figura 4.1: Vista geral da arquitetura proposta

A utilização de Jetty torna-se ainda mais benéfica, no contexto desta solução, por o SDK desenvolvido pela AMA, oferecer uma panóplia de funcionalidades de fácil integração, devido à escolha desta tecnologia. Será explicado mais adiante como é que estas integrações podem ser facilmente realizadas.

Cenário

O diagrama 7.3 mostra o caso de uso implementado na solução proposta para a integração do SDK do *middleware* desenvolvido pela AMA. Consiste em dois atores: a SIAG, como empresa integradora e o cliente da SIAG, utilizador dos seus serviços.

A SIAG tem de primeiramente integrar o SDK do *middleware* desenvolvido pela AMA, que inclui um conjunto de funcionalidades, como exemplificados na secção 2.2.2, onde se destaca a funcionalidade de assinatura digital com Cartão de Cidadão, a ser utilizado pelos seus clientes.

Vista lógica

O diagrama 4.2 é representativo de como está organizada a solução. Como cliente encontra-se a aplicação da SIAG, cuja *interface* é apresentada através do módulo SiagPrimeFaces. Esta aplicação efetua os pedidos por HTTP ao servidor JavaScript, a correr no lado do cliente. Para o Servidor Jetty comunicar com o JavaScript, e vice-versa, é utilizado JSON.

Este tipo de organização e comunicação assemelha-se à arquitetura cliente-servidor, em que o cliente faz os pedidos a um servidor, este processa-os e devolve o resultado dos mesmos. Neste caso, o cliente é o módulo SiagPrimeFaces, através do qual o utilizador faz o pedido de assinatura de documentos, e o servidor é o Servidor Jetty, intermediado pelo JavaScript, processa-o e devolve o ficheiro assinado.

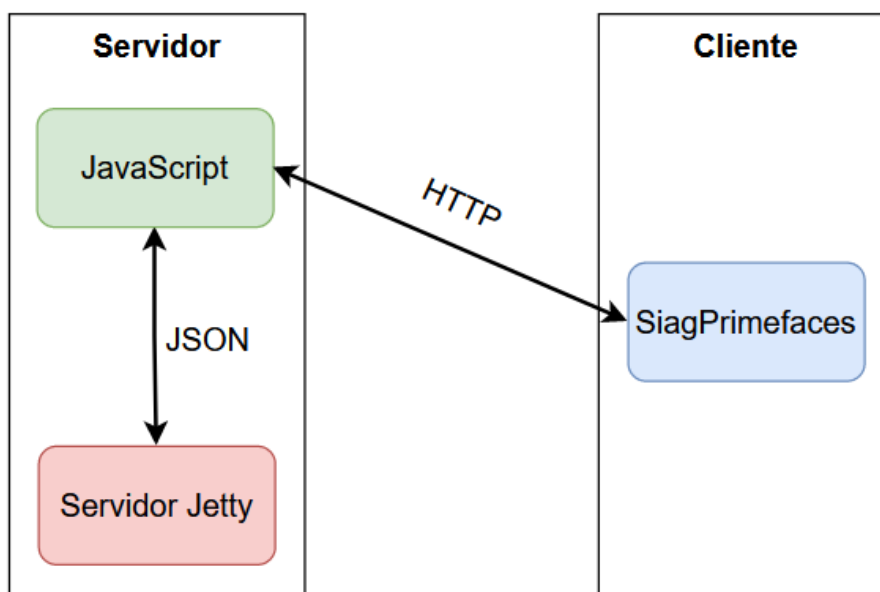


Figura 4.2: Arquitetura técnica da solução: cliente-servidor

Esta arquitetura cliente-servidor, pode ser separada em dois componentes em comunicação: cliente e servidor.

O cliente (módulo SiagPrimeFaces) utiliza as tecnologias descritas na Secção 7.1 para o desenvolvimento de *interfaces*. Para associar os componentes UI (reutilizados da biblioteca PrimeFaces) a modelo de dados, utiliza a *framework* JavaServer Faces, como é exemplificado na figura 4.2. O utilizador após inserir os parâmetros necessários para a assinatura de um documento, clica no botão "Avançar". O controlador fornecido pelo JavaServer Faces, FacesServlet, recebe o pedido, *renderiza* a *interface* seguinte e transfere os dados para o modelo responsável por essa mesma *interface*, neste caso a classe DocumentosDespesaController.java. Quaisquer alterações que o modelo queira realizar à vista, passam pelo Controlador, para depois as mostrar ao utilizador.

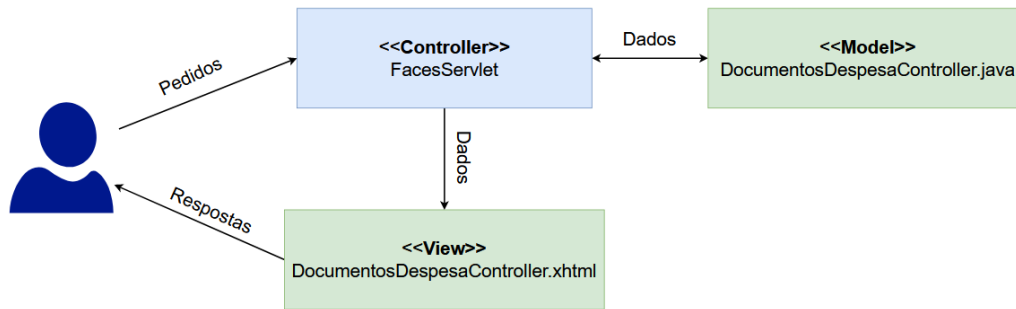


Figura 4.3: Arquitetura técnica do módulo SiagPrimeFaces:JavaServer Faces

O servidor Jetty é um servidor a correr na máquina do cliente. Quando o utilizador executa o JNLP, este inicia a aplicação a partir da classe definida no seu arranque, neste caso a classe JettyServer.java. O servidor é iniciado, assim como um *handler* associado a uma determinada API. Para a integração proposta, a classe ContextHandler, fornecida pela biblioteca do Jetty 9, associa ao *endpoint* “\digital\signature”, o *handler* DigitalSignatureServlet.java para gerir a lógica dos pedidos feitos. Esta classe acede à meta-informação recebida no pedido HTTP e utiliza a classe Auth.java para servir de intermediária com o SDK desenvolvido pela AMA.

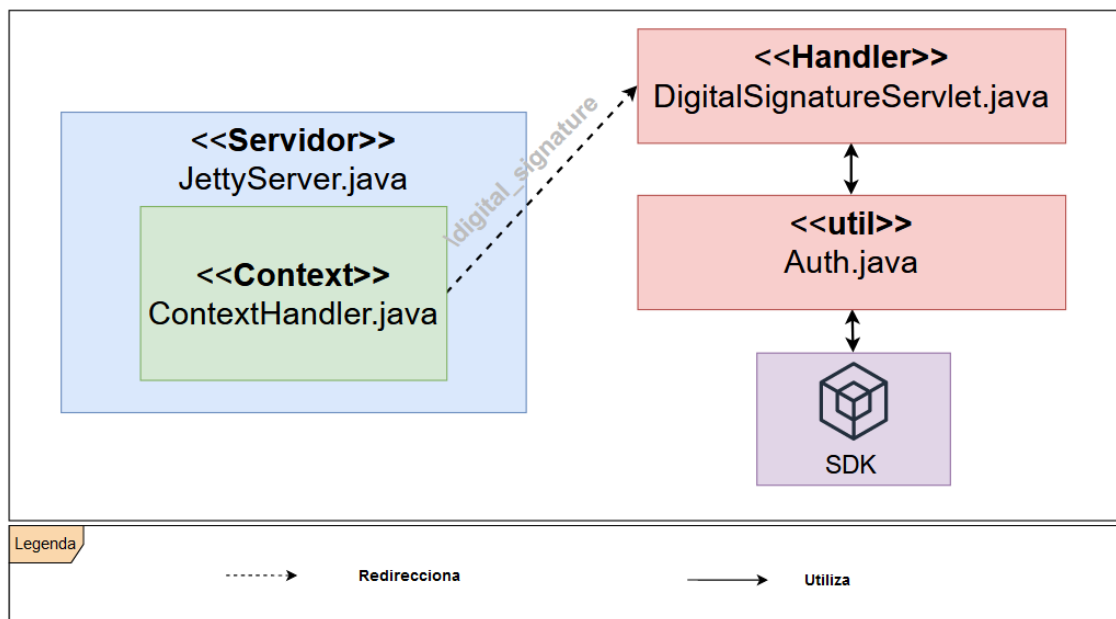


Figura 4.4: Arquitetura técnica do módulo SiagSignature: Servidor Jetty

Vista de processo

Para a representação desta vista, foi utilizado o diagrama de sequência. Este tipo de representação apresenta uma sequência de ações ou controlos de fluxo, num sistema semelhante a um diagrama de fluxo de dados.

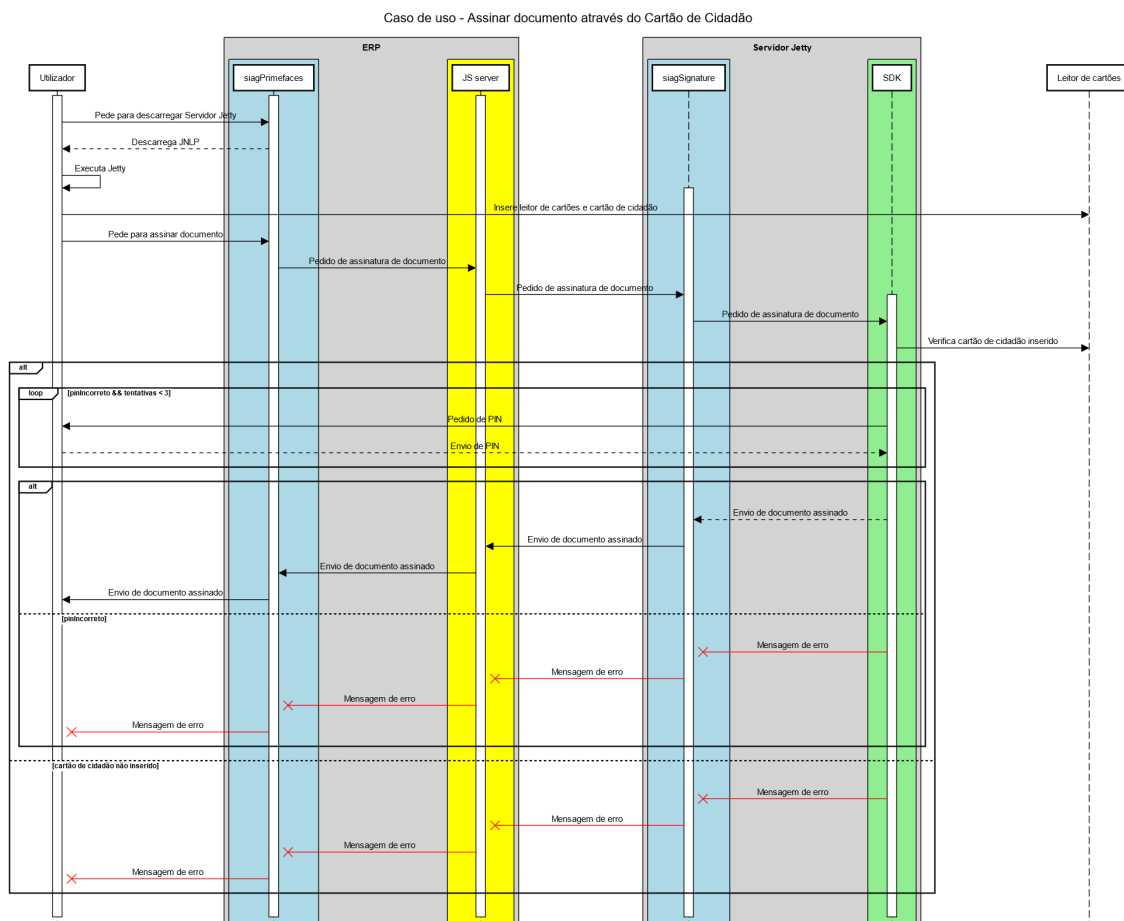


Figura 4.5: Diagrama de sequência - Assinatura digital com Cartão de Cidadão

Este diagrama representa como é que o servidor Jetty comunica com os restantes sistemas/dispositivos envolvidos na solução proposta.

- O utilizador começa por descarregar o JNLP - executável para correr Servidor Jetty;
- Executa o JNLP e o Servidor Jetty inicia no porto 8082, na própria máquina do cliente;
- Insere o leitor de cartões de cidadão na porta USB e o respetivo Cartão de Cidadão;
- O utilizador acede à interface responsável pelo pedido de assinatura do documento, insere as configurações de assinatura e realiza o pedido, clicando no botão "Avançar";
- O SDK desenvolvido pela AMA e que contém as bibliotecas para aceder e assinar documentos, através do Cartão de Cidadão, verifica se este está inserido no leitor de cartões;
- Caso o Cartão de Cidadão esteja inserido:
 - É pedido ao utilizador que insira o PIN. Este passo é repetido enquanto o PIN estiver errado e não exceder o número de tentativas máximas definidas.
 - Caso o PIN esteja correto, o documento é assinado e devolvido ao utilizador;
 - Caso o PIN esteja incorreto, é devolvida uma mensagem de erro ao utilizador;
- Caso o Cartão de Cidadão não esteja inserido:
 - É devolvida uma mensagem de erro ao utilizador.

Vista física

Para representar este tipo de vista, foi utilizado o diagrama de implementação, ou "deployment diagram". Este diagrama mostra como é que os elementos do sistema estão configurados em tempo de execução. Este tipo de representação é usado para a modelação dos aspetos físicos de um sistema orientado a objetos.

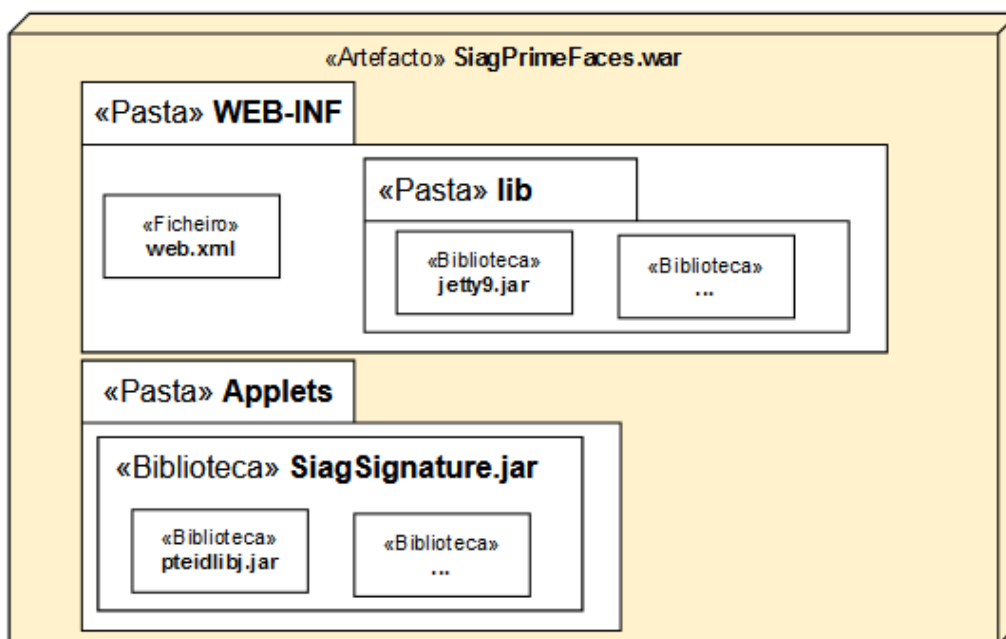


Figura 4.6: Estrutura SiagPrimeFaces.war

Um ficheiro WAR é um pacote comprimido de classes java e aplicações que são executadas num *web server*.

Para uma aplicação ser *deployed* num servidor *web*, precisa de ser comprimida num ficheiro WAR, estruturado como na figura 4.6, que contém as bibliotecas na pasta "lib", o ficheiro de configurações "web.XML" e a pasta "applets", que contém aplicações para correr no *browser*.

Neste caso, o ficheiro "deploy-jetty.jnlp", como descrito na secção 7.1.6, tem como referência a biblioteca SiagSignature.jar, localizada na pasta "applets", para poder executar o servidor Jetty remotamente.

Vista de desenvolvimento

No diagrama 4.8, estão representados os pacotes no desenvolvimento da integração do SDK do *middleware*, desenvolvido pela AMA, para a assinatura digital de documentos. Os pacotes a vermelho são os módulos maiores, projectos maven, com as suas próprias configurações de execução e *deployment*, e os seus respetivos pacotes.

Nas tabelas 7.7 e 7.8 encontra-se descrito as classes correspondente a cada pacote da Figura 4.8.

A listagem 3 de código serve de intermediário entre os *endpoints* da solução. De um lado existe o servidor onde está a correr a aplicação da SIAG onde o utilizador insere os parâmetros para assinar um documento e do outro, a correr na máquina do cliente, o Servidor Jetty. Como os servidores estão a

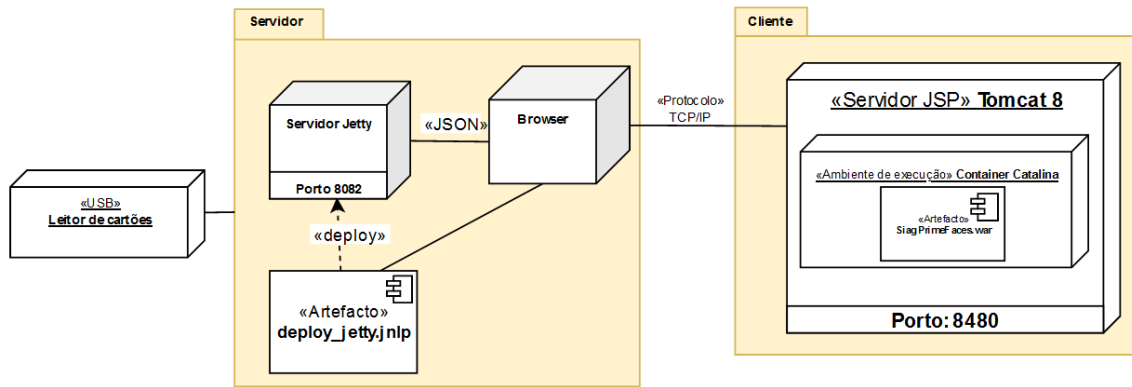


Figura 4.7: Diagrama de componentes

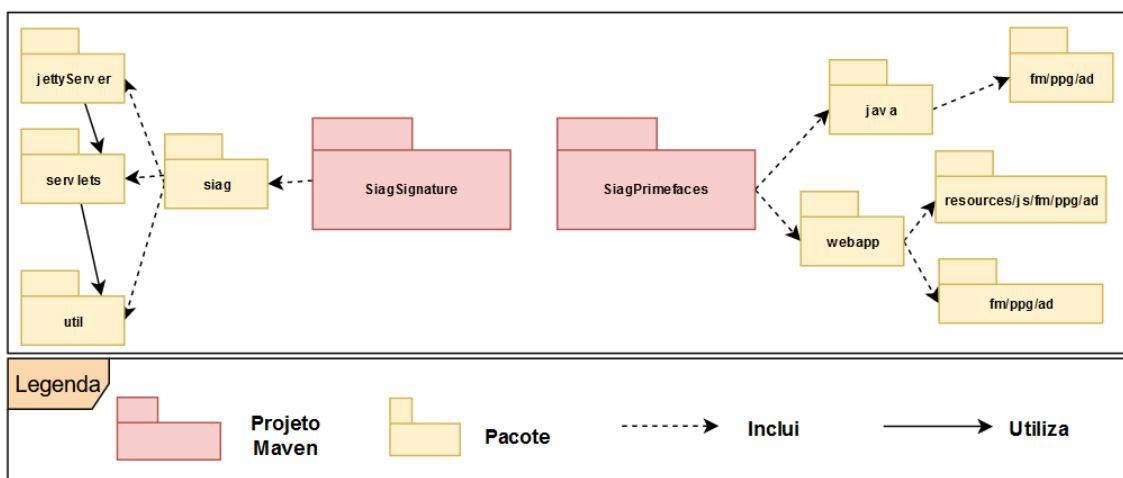


Figura 4.8: Diagrama de pacotes

correr em máquinas diferentes, não têm como comunicar: é aí que entra o JavaScript, uma linguagem de programação bastante flexível que atua do lado do cliente.

Nesta solução, a classe assinatura.js recebe o pedido da aplicação da SIAG e reencaminha para o Servidor Jetty, que está a correr no mesmo ambiente de execução que o JavaScript.

Listagem 3: Classe assinatura.js

```

1 function signDocument(base64, reducedSignature, location, reason, signaturePosX, signaturePosY, page){
2   var url = ' http: // localhost:8082 / digital_signature ';
3   $.ajax({
4     url: url,
5     dataType: "json",
6     type: "POST",
7     crossDomain: true,
8     data : {
9       'fileBase64' : base64,
10      'reducedSignature' : reducedSignature,
11      'location' : location,
12      'reason' : reason,
13      'signaturePosX' : signaturePosX,
14      'signaturePosY' : signaturePosY,
15      'page' : page,
16    },
17  });
18 }

```

O servidor local, neste caso a classe JettyServer.java, recebe um pedido em JSON, no porto definido,

para o *servlet* "digital_signature". Correspondente a este *servlet*, está mapeada a classe `DigitalSignatureServlet.java`, que recebe o pedido, retira a meta-informação do cabeçalho HTTP da mensagem recebida e efetua o pedido de assinatura à classe `Auth.java`.

A classe `Auth` no seu arranque, começa por carregar a biblioteca que contém a API dos serviços fornecidos pelo SDK, com o trecho de código abaixo.

Listagem 4: Carregar biblioteca em Java (AMA, 2022c)

```

1  static {
2      try {
3          System.loadLibrary(" pteidlibj ");
4      } catch ( UnsatisfiedLinkError e) {
5          System.err.println ("Native code library failed to load. \n" + e);
6          System.exit (1);
7      }
8  }

```

Esta classe recebe o pedido de assinatura de um documento, vindo do *servlet*. Com a informação necessária retirada anteriormente, efetua o pedido ao *middleware* da AMA.

Listagem 5: Exemplo de pedido de assinatura em C++ (AMA, 2022c)

```

1  PTEID_PDFSignature signature("~/home/user/input.pdf");
2
3  /* Adicionar uma imagem customizada assinatura visvel
4     O array de bytes image_data deve conter uma imagem em formato
5     JPEG com as dimenses recomendadas (351x77 px) */
6  PTEID_ByteArray jpeg_data(image_data, image_length);
7  signature .setCustomImage(jpeg_data);
8
9  // No caso de se querer o formato pequeno da assinatura
10 signature .enableSmallSignatureFormat();
11
12 // Configurar o perfil da assinatura :
13 signature . setSignatureLevel (PTEID_SignatureLevel::PTEID_LEVEL_TIMESTAMP);
14
15 // Especificar local da assinatura e motivo
16 const char * location = "Lisboa, Portugal";
17 const char * reason = "Concordo com o conteúdo do documento";
18
19 // Especificar o nmero da pgina e a posio nessa mesma pgina onde a indicao visual da assinatura
20 // aparece
21 int page = 1;
22 double pos_x = 0.1; // Valores de 0 a 1
23 double pos_y = 0.1; // Valores de 0 a 1
24 eidCard.SignPDF(signature, page, pos_x, pos_y, location, reason, "~/home/user/output.pdf");

```

Como se pode observar pela listagem 5, a API disponibilizada pela AMA para a assinatura digital de um documento é:

- um objeto do tipo `PTEID_PDFSignature`, que recebe a localização do ficheiro a assinar;
- o número da página a ser assinada;
- coordenada horizontal da assinatura, considerando 0 o canto superior esquerdo;
- coordenada vertical da assinatura, considerando 0 o canto superior esquerdo;
- parâmetros opcionais como:
 - localização;
 - razão.
- localização do ficheiro de saída do ficheiro assinado;

- por último, o objeto que invoca a chamada da função, que é um objeto representante do Cartão de Cidadão, inserido no leitor de cartões.

No final da assinatura, é devolvido ao servidor Javascript, também em JSON, uma mensagem de sucesso com o documento assinado, em base64, ou uma mensagem de erro explicativa.

4.1.4 Demonstração

De seguida encontra-se a *interface* com o utilizador correspondente ao caso de uso definido na secção 4.1.3.

A interface de parametrização de assinatura para o Cartão de Cidadão apresenta os seguintes elementos:

- Modo de assinatura:** Seleção entre "CMD" (radio desativado) e "CC" (radio ativado).
- Assinatura reduzida ***: Checkbox desativado.
- Localização**: Campo de texto vazio.
- Razão**: Campo de texto vazio.
- Escolha a posição da assinatura no documento ***: Gridda de 6 linhas e 4 colunas.
- Página**: Campo de texto com o valor "1", botões de incremento (+) e decremento (-).
- Avançar**: Botão de ação principal.

Figura 4.9: Interface para parametrizar assinatura - Cartão de Cidadão

Cada *input* necessário no ato de assinatura digital por Cartão de Cidadão encontra-se identificado por um número.

- **Modo de assinatura:** Componente do tipo "SelectOneRadio", em que o utilizador só pode escolher um elemento da lista apresentada. Este *input* especifica qual é o tipo de assinatura escolhida. Conforme a escolha, o formulário abaixo é atualizado;
- **Assinatura reduzida:** Componente do tipo "SelectBooleanCheckbox", em que só pode ter o valor de "Verdadeiro" ou "Falso". O utilizador decide se quer que a assinatura seja reduzida;
- **Localização:** Componente do tipo "InputText", em que permite ao utilizador inserir texto. Este *input* serve para introduzir a localização de onde está a assinar, sendo opcional no processo de assinatura;

- **Escolha a posição da assinatura no documento:** Componente do tipo "SelectOneButton", em que só um botão pode ser selecionado. Este *input* serve para indicar onde é que a assinatura vai ser introduzida no documento, usando uma tabela de 6 linhas por 4 colunas para simular o documento;
- **Página:** Componente do tipo "Spinner", que permite o acréscimo ou decréscimo de um determinado valor. No contexto deste formulário, tem o propósito de indicar qual é a página em que o utilizador quer inserir a assinatura;
- **Avançar:** Componente do tipo "Button", para começar o processo de tratamento dos dados inseridos e envio para a AMA.

Estando o Servidor Jetty a correr e o leitor de cartões inserido com o Cartão de Cidadão posto, após o utilizador clicar "Avançar", surge um *popup* para este inserir o PIN, como exemplificado na figura 4.10. Após inserir o PIN com sucesso, na Figura 7.4 o documento a assinar (situado à esquerda), é substituído pelo documento assinado, e surge um *popup* de sucesso. Tendo em conta que o Servidor Jetty é uma consola, que imprime comandos, não é apresentada uma *interface*.

A interface é um formulário de diálogo com o título "Assinar com Cartão de Cidadão" e um ícone de um cartão de cidadão. Abaixo do título, há um aviso: "ATENÇÃO: Vai realizar uma assinatura eletrónica válida com o seu Cartão de Cidadão." Segue-se um campo de texto rotulado "PIN de assinatura". Na base do formulário, há dois botões: "CANCELAR" (cinza) e "CONFIRMAR" (azul).

Figura 4.10: Interface para inserir PIN de assinatura - Cartão de Cidadão

4.1.5 Avaliação

Como definido na secção 4.1.2, há três restrições e objetivos a cumprir quanto à implementação do SDK do *middleware* desenvolvido pela AMA para a assinatura de documentos.

Quanto às restrições, o *software* a integrar tinha de ser, preferencialmente *open-source*, compatível com a linguagem de programação Java 8, e assinar documentos de acordo com a lei portuguesa. A utilização da tecnologia Jetty e da biblioteca desenvolvida pela AMA, cumpre as restrições acima referidas.

Quanto aos requisitos não-funcionais definidos, a solução implementada tinha de ser adjetivada como modificável, portátil e usável.

Relativamente ao primeiro requisito não-funcional, como dito anteriormente, o Servidor Jetty assina documentos, de pedidos que sejam endereçados à API "digital_signature". Caso, seja do futuro interesse da empresa, incluir funcionalidades como autenticação, uma possível solução será criar outro "servlet", que utilizaria a classe Auth.java, para aceder às funcionalidades eletrónicas do Cartão de Cidadão, retirar informação como o nome próprio e autenticar na plataforma, com base no nome do utilizador.

Assim, a única alteração é a criação de um novo "servlet", sem necessidade de alterar o "servlet" que assina documentos, tornando a solução facilmente modificável.

Relativamente ao requisito não-funcional "Portabilidade", a atual solução funciona em qualquer sistema operativo, visto que são dois servidores em comunicação, por não dependência de alguma característica de um determinado sistema operativo. Quanto ao ambiente de execução onde a solução está inserida, o uso de JWS e JNLP, apesar de funcional, é desencorajado, para a versão 9+ de JRE. Há duas possíveis soluções:

- Utilização de OpenWebStart, que é uma reimplementação de JWS;
- Abertura de um porto na máquina do cliente;

Quanto ao requisito da usabilidade, era expectável a solução apresentar uma *interface* que permite ao utilizador inserir os dados necessários para a assinatura, de forma simples, intuitiva e lógica.

Assim, para uma melhor eficácia e utilização de tecnologias a longo-prazo, é da opinião da empresa reiterar o passo 3 "Design e Desenvolvimento", com uma das soluções propostas acima. Qualquer uma destas opções altera a comunicação entre cliente-servidor, contudo não altera a lógica de processamento e de pedidos.

4.1.6 Comunicação

Como constatado anteriormente, a necessidade de assinatura de documentos, como orçamentos, autorizações, entre outros, é de grande premência, para clientes que requerem este tipo de serviços automatizados e facilitados. A solução proposta por mim permite a integração com uma biblioteca desenvolvida pela AMA, que possibilita o acesso às funcionalidades do Cartão de Cidadão, entre as quais, a assinatura de documentos, com a mesma validade legal que uma assinatura à mão, reconhecida por um advogado ou notário.

A utilização de tecnologias como Jetty, é uma parte fulcral da solução, por, não só permitir a comunicação cliente-servidor, mas também ser um servidor aplicacional local bastante leve, configurado com regras de segurança, através do uso de tecnologias como JNLP para a sua execução. Em suma, esta solução é eficaz para a resolução do problema, com base nas características do mesmo. Atualmente é uma boa solução, contudo, com futuras atualizações do Java Runtime Version a executar na máquina do cliente, deixará de o ser e necessitar de alterações à sua implementação, como referido na secção anterior.

4.2 Iteração 2: Integração de serviço Chave Móvel Digital

4.2.1 Identificação do Problema

Como dito anteriormente, a possibilidade de assinatura digital é crucial e urgente para os clientes da SIAG. Ainda mais importante é realizar este serviço através de vários meios, seja por Cartão de Cidadão ou um meio mais moderno e abrangente a qualquer cidadão: Chave Móvel Digital.

O Governo quer, com esta medida, facilitar o acesso aos serviços online do Cartão de Cidadão, até aqui pouco utilizados devido à dificuldade em aceder aos leitores de cartões e em utilizá-los.

Um utilizador que não tenha Cartão de Cidadão na sua posse, ou leitor de cartões, vê-se obrigado a imprimir o documento que pretende assinar. Esta é uma das razões para o desenvolvimento da CMD, permitindo ao cidadão assinar documentos em qualquer sítio e plataforma, desde que tenha o seu telemóvel na sua posse, para poder introduzir o OTP - código de seis dígitos enviado para o telemóvel.

4.2.2 Definição de objetivos

O Serviço de Assinatura da Chave Móvel Digital permite que as aplicações integrem a funcionalidade de assinatura digital de documentos com uso da Chave Móvel Digital nos seus fluxos. Para isto, a AMA disponibiliza um documento a especificar como é que deve ser feita a integração de cada uma das operações que compõem o serviço de assinatura com Chave Móvel Digital.

Para além do documento a especificar como é que deve ser feita a integração das operações que constituem o serviço, também é fornecido um documento WSDL. Um documento WSDL (Web Services Description Language) é uma linguagem baseada em XML utilizada para descrever *web services*. Este documento para além de descrever o serviço, também especifica como acedê-lo e quais as operações fornecidas pelo mesmo.

De seguida segue a descrição de cada operação implementada. A especificação de cada operação, descrita no WSDL fornecido pela AMA, encontra-se na secção 7.3.

1. **GetCertificate**: obtém certificado público do cidadão;
2. **SCMDSign**: quando se pretende assinar um único documento, deve ser utilizada esta operação que recebe o *hash* do documento a assinar;
3. **SCMDMultipleSign**: quando se pretende assinar vários documentos, deve ser utilizada esta operação que recebe a lista dos *hash* dos documentos a assinar. Esta operação é indicada para a assinatura múltipla e em simultâneo de vários documentos, não sendo, por isso, implementado nesta iteração.
4. **GetCertificateWithPin**: obtém o certificado do cidadão com o PIN de assinatura associado. Não foi necessário implementar esta operação por o certificado do Cidadão ser utilizado na operação **SCMDSign**;
5. **ForceSMS**: força o envio de um SMS com um novo código OTP, associada a uma operação em processo. No contexto deste projeto, não foi implementada esta operação por não ser operação obrigatória;

6. **ValidateOtp**: último passo do processo, que valida o código de segurança enviado e devolve o *hash* assinado, uma lista de *hash* assinados ou o certificado do cidadão, conforme ter sido invocada anteriormente a operação SCMDSign, SCMDMultipleSign ou GetCertificateWithPin.

Tipos de dados que tenham "base64Binary" na sua descrição requerem que sejam codificados em formato ASCII, para evitar a alteração dos mesmos quando transferidos entre sistemas, e que suporte sistemas que não utilizem codificação de 8 bits.

Os outros serviços não precisam de ser detalhados por não terem sido utilizados no desenvolvimento desta iteração.

O serviço de assinatura digital através da Chave Móvel Digital, desenvolvido pela AMA, é disponibilizado por intermédio da comunicação com o protocolo SOAP. Por isso, para a integração das operações que compõem o serviço, é necessário:

1. Integrar protocolo de comunicação HTTPS com autenticação;
2. Enviar mensagens SOAP para transferência de dados.
3. Implementar as operações referidas acima;
4. Desenvolver uma *interface*, que utilize a *framework* JavaServer Faces, para o utilizador poder assinar um documento, utilizando a Chave Móvel Digital.

Para que a solução seja considerada viável, a AMA disponibilizou um documento³ com orientações de como implementar o serviço. Algumas destas orientações enquadram-se nos requisitos não-funcionais expectáveis da solução. São estes:

- **Legalidade** - Por a Chave Móvel Digital conter dados privados do Cidadão, é crucial que a solução cumpra as orientações legais impostas pela AMA para a integração deste serviço. Exemplo de algumas orientações são:
 - "Não guardar nos logs os dados do PIN ou Pin temporário";
 - "Não apresentar visível para o utilizador os dígitos do PIN da CMD".
- **Interoperabilidade** - Capacidade de um sistema comunicar com outro sistema, utilizando formatos de dados e protocolos específicos;
- **Usabilidade** - Esta *interface* tem de cumprir as orientações definidas pela AMA, relativamente à experiência do utilizador. Alguns exemplos são:
 - "Apresentar sempre o documento a assinar pelo cidadão.";
 - "Identificar e informar sobre os vários passos do processo de assinatura [cf. secção 3.2.4 da POL16] à medida que os mesmo ocorrem.".

O documento fornecido pela AMA para a integração do serviço Chave Móvel Digital numa aplicação externa serve de orientador para uma correta integração deste serviço em ambiente de pré-produção.

³https://github.com/amagovpt/doc-CMD-assinatura/blob/main/guidelines/guidelines_CMD_assinatura_apps_externas.xlsx

Isto significa que a implementação será feita em ambiente de teste e, quando concluída, será avaliada, de acordo com as diretrizes definidas, *a priori*, pela AMA. Após a aprovação da implementação e de todas as diretrizes serem cumpridas, a SIAG poderá proceder para ambiente de produção, semelhantemente ao serviço Chave Móvel Digital, oferecido pela aplicação Autenticacao.gov.

4.2.3 Design e desenvolvimento

Atributos qualitativos

As tabelas 7.9, 7.10 e 7.11 descrevem cenários concretos para os três atributos de qualidade que devem estar incluídos na solução.

O diagrama 4.11 representa os principais componentes da solução proposta no sistema geral e as respectivas ligações.

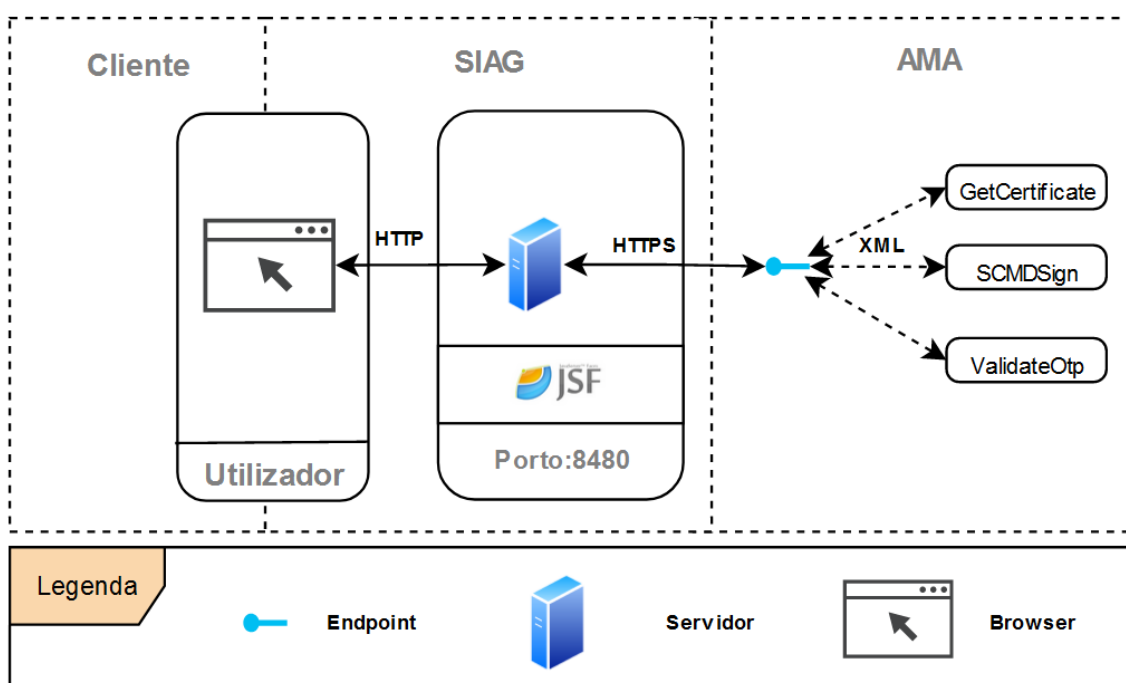


Figura 4.11: Vista geral da arquitetura da solução

Ao contrário da iteração anterior, nesta solução existem 3 entidades envolvidas: O cliente, a SIAG e a AMA. O sistema da SIAG serve de intermediário entre o pedido de assinatura do cliente e a devolução do documento assinado pelo certificado do Cidadão.

O cliente através do *browser* insere os dados necessários para a assinatura digital de um documento, através da Chave Móvel Digital, como:

- Número de telemóvel;
- PIN de assinatura;
- Dados opcionais como:
 - Localização;
 - Razão;

- Posição de assinatura;
- E, posteriormente, o OTP.

Tendo em conta que os dados introduzidos são de conteúdo sensível, estes não podem ser transmitidos em *plain-text* entre a SIAG e a AMA. Assim, os dados necessitam de ser encriptados, nomeadamente através de criptografia assimétrica RSA, em que é utilizada a chave pública fornecida pela AMA para a cifra dos mesmos. Posteriormente, os dados são transmitidos através de mensagens SOAP, em formato XML para o *endpoint* definido⁴.

Como explicado na secção 2.1.4, através do cabeçalho do pedido efetuado, é possível identificar qual o componente responsável pelo processamento dos mesmos, como o GetCertificate, SCMDSign e ValidateOtp. Após o reencaminhamento dos pedidos, a AMA descripta os dados enviados, utilizando a chave privada, processa os pedidos e envia uma mensagem SOAP em formato XML com a resposta. Esta criptografia RSA traz uma camada extra de segurança à camada de transporte HTTPS, na medida em que só a AMA é que consegue descriptar os dados sensíveis do Cidadão.

Visão de casos de uso

O diagrama 4.12 mostra o principal, e único, caso de uso implementado na solução proposta para a integração do serviço de assinatura digital com Chave Móvel Digital desenvolvido pela AMA.

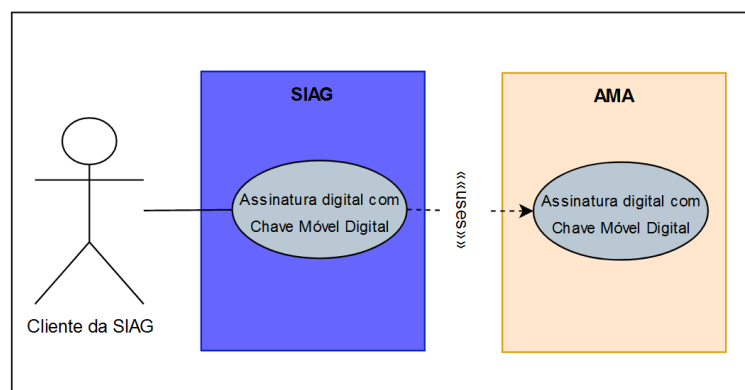


Figura 4.12: Diagrama de caso de uso - Assinatura digital com Chave Móvel Digital

Vista lógica

O diagrama 4.13 é representativo de como está organizada a solução. Como se pode observar, esta arquitetura assemelha-se à arquitetura cliente-servidor, em que há um cliente que faz pedidos, neste caso o sistema da SIAG (Que será desdobrado de seguida), e o servidor da AMA, que responde a esses mesmos pedidos. Como dito anteriormente, para efetuar uma correta integração entre a entidade externa (AMA) e a SIAG, é necessário que a comunicação seja feita com base no protocolo de transporte HTTPS (Protocolo HTTP com uma camada extra de segurança que permite utilizar protocolos de encriptação como TLS e SSL), e os dados sejam enviados numa mensagem SOAP, em formato XML. HTTPS é uma extensão segura de HTTP, em que os sites que configurarem um certificado SSL/TLS podem utilizar o protocolo HTTPS para estabelecer uma comunicação segura com o servidor. Tem

⁴<https://preprod.cmd.autenticacao.gov.pt/Ama.Authentication.Frontend/SCMDService.svc>

como objetivo tornar segura a transmissão de informações sensíveis como dados pessoais, de pagamento ou de *login*.

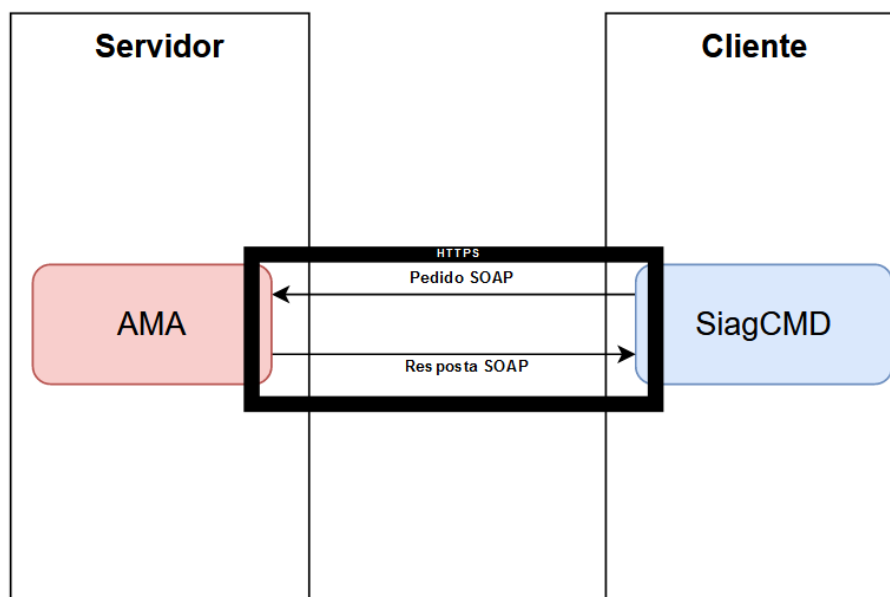


Figura 4.13: Arquitetura técnica da solução: cliente-servidor

Esta arquitetura cliente-servidor, pode ser separada em dois componentes em comunicação: cliente e servidor.

O servidor corresponde ao servidor da AMA que responde a pedidos de assinatura digital por Chave Móvel Digital, no *endpoint* definido.

O cliente SiagCMD é um conjunto de módulos em comunicação, que alberga o módulo SiagPrimefaces, responsável pela apresentação da *interface* e o módulo Siag, responsável pelo processamento de operações invocadas de outros sistemas.

Os pedidos efetuados pelo módulo SiagPrimefaces ao módulo Siag são intermediados pelo *broker* ActiveMQ, e postos numa *queue* JMS. O consumidor, relativamente a este caso de uso, é o sub-módulo "SignDocument". Este sub-módulo contém toda a lógica responsável pelo processamento de pedidos a entidades externas para a assinatura de documentos, como a AMA e a Multicert (como se poderá observar na próxima iteração).

O módulo Siag contém a lógica de processamento de pedidos a entidades externas, seja para faturação eletrónica, como ACIN⁵ e ESPAP⁶, ou o "fénix", que trata da de gestão de alunos, ou assinatura de documentos.

No contexto da Figura 4.14, o sub-módulo "SignDocument" é o componente responsável pelo envio de pedidos SOAP à AMA e pela devolução da respetiva resposta ao módulo "SiagPrimefaces", para apresentação dos dados ao utilizador.

O módulo SiagPrimefaces é o mesmo utilizado na iteração anterior. Contudo, para esta solução, este módulo comunica com o módulo siag, através do *middleware* ActiveMQ.

⁵<https://www.acin.pt/>

⁶<https://www.espap.gov.pt/?AspxAutoDetectCookieSupport=1>

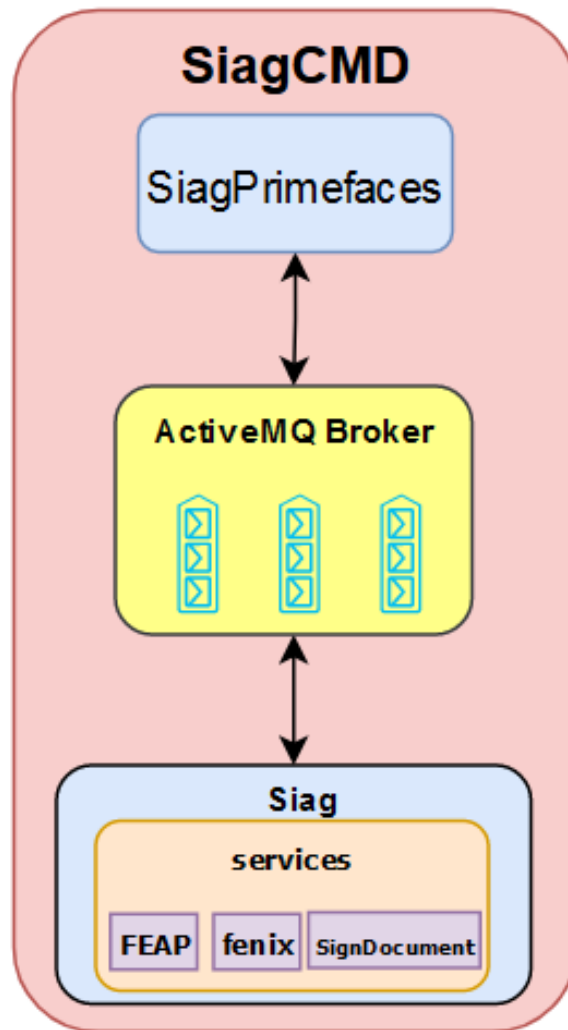


Figura 4.14: Arquitetura técnica do módulo SiagCMD

Vista de processo

A assinatura digital de um documento por Chave Móvel Digital é um processo complexo que envolve validação de dados, encriptação dos mesmos, *hash* do documento e "estilização" da assinatura. Este processo é representado através do diagrama de sequência 4.15, que será explicado em maior detalhe cada um dos passos referidos anteriormente.

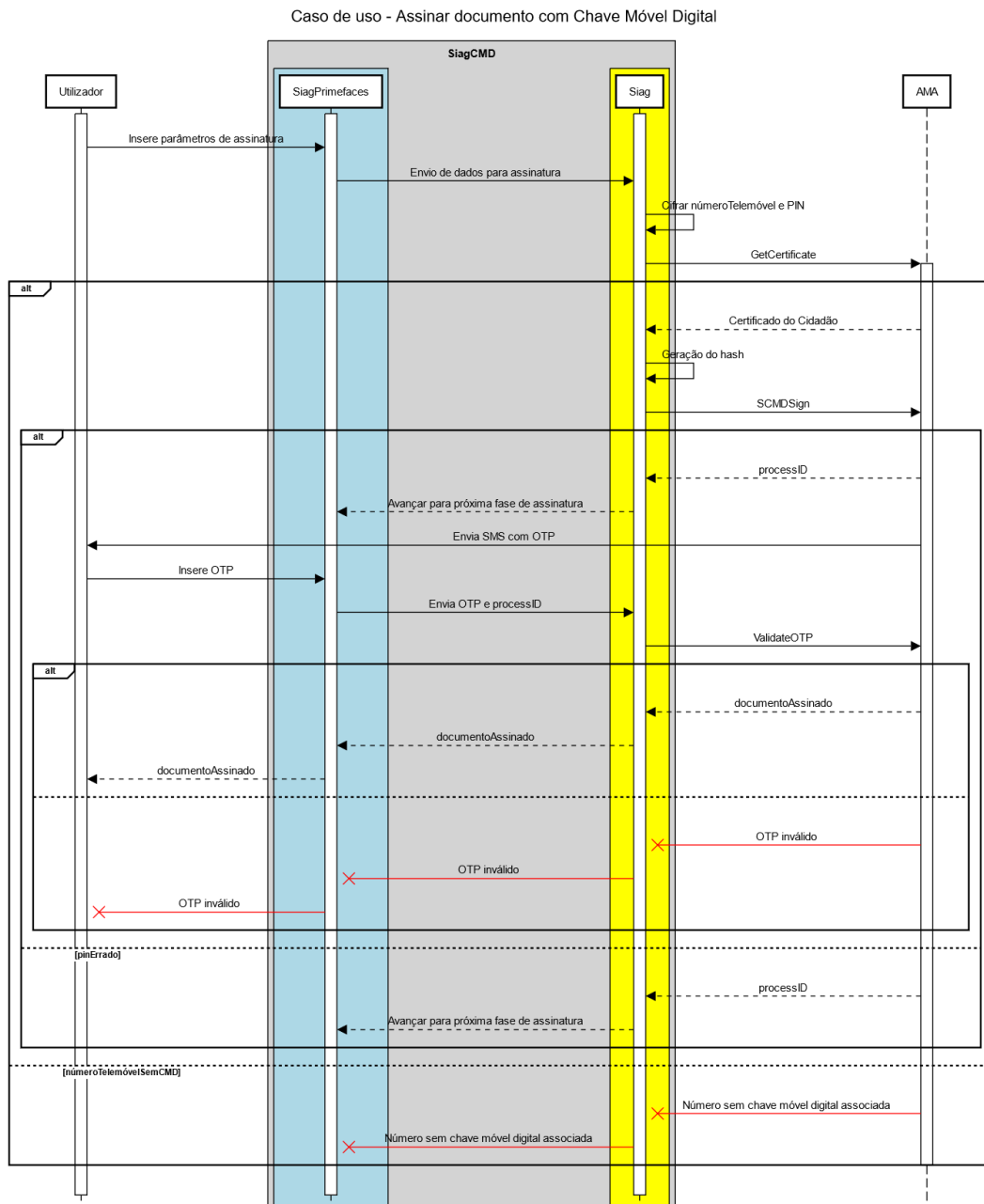


Figura 4.15: Diagrama de sequência - Assinar documento com Chave Móvel Digital

Ao utilizador é exposta uma *interface* onde pode inserir os dados necessários para assinar um documento através da Chave Móvel Digital. Para isto, ele precisa de inserir o número de telemóvel (com CMD ativa), o PIN de assinatura de documentos (enviado por carta), a posição no documento (grelha de quatro linhas por seis colunas) e o número da página. Dados como "localização" e "razão" são opcionais. Caso o utilizador opte por assinar invisivelmente um documento, só o número de telemóvel e PIN de assinatura são necessários.

De seguida, o controlador responsável pela *interface* envia os dados para o módulo Siag, para que os processe e submeta para a AMA. O módulo recebe os dados, cifra com a chave pública da AMA o número de telemóvel e PIN de assinatura e envia o primeiro para a AMA para poder obter o certificado do Cidadão, através da operação **GetCertificate**. Caso o número inserido não tenha Chave Móvel Digital ativada, é comunicado este erro ao utilizador e o processo de assinatura termina.

Após a invocação da operação **GetCertificate**, é necessário enviar para a AMA o documento com a assinatura "em branco" do Cidadão, cifrando com o certificado público do cidadão. Esta "falsa" assinatura, também chamada da *hash*, é estilizada de acordo com o que a entidade integradora pretende⁷, neste caso a SIAG. Para o documento ser enviado, deve ser gerado um *hash* do mesmo, utilizando criptografia RSA, com codificação EMSA-PKCS1-v1_5. Semelhantemente aos dados sensíveis do utilizador, o documento a assinar não pode ser transferido em *plain-text* entre entidades, daí a necessidade da codificação com a chave pública do Cidadão.

De seguida, é invocada a operação **SCMDSign**. Caso os dados enviados na invocação forem válidos, a chamada ao *web service* retorna um *processID* que é uma informação identificativa do processo de assinatura digital a decorrer. O módulo Siag comunica ao SiagPrimefaces para avançar para a próxima fase de assinatura, em que o utilizador insere o OTP. Caso o PIN não corresponda ao do número de telemóvel, a AMA retorna um *processID* e, por esse motivo, o processo de assinatura continua para a fase de inserção do OTP, apesar do SMS enviado conter uma mensagem de "PIN incorreto". O processo de assinatura termina.

Simultaneamente com o envio do *processID*, a AMA envia uma SMS ao número de telemóvel inserido na primeira operação, com o código OTP. O utilizador insere o OTP, o módulo SiagPrimefaces através de ActiveMQ envia o código OTP, juntamente com o *processID*, para a AMA associar a esse processo, o respetivo código OTP. o módulo Siag invoca a operação **ValidateOtp** com os dados referidos anteriormente. Caso os dados sejam válidos, o documento enviado em **SCMDSign** (com a assinatura "em branco") é retornado, com a assinatura inserida e validada. Este documento é depois apresentado ao utilizador. Caso o OTP inserido esteja incorreto, é enviada uma mensagem de erro ao utilizador, terminando o processo de assinatura do documento.

Vista física

Este tipo de vista tem como objetivo mostrar como é que os elementos de um sistema estão configurados em tempo de execução e como é que é feita a modelação dos aspetos físicos de um sistema orientado a objetos. Ao contrário da iteração anterior, em que é implementado um servidor na máquina do cliente e por isso é importante mostrar como é que os dois sistemas físicos (SIAG e servidor Jetty)

⁷Exemplo na Figura 7.8

estão configurados e em comunicação, nesta iteração não há integração de sistemas físicos na solução, daí este tipo de vista não ser importante no contexto da arquitetura da mesma.

Vista de desenvolvimento

No diagrama 4.16, estão representados os pacotes no desenvolvimento da integração da assinatura digital de um documento, através da Chave Móvel Digital. Os pacotes a vermelho são os módulos maiores, projectos maven, com as suas próprias configurações de execução e *deployment*, e os seus respetivos pacotes.

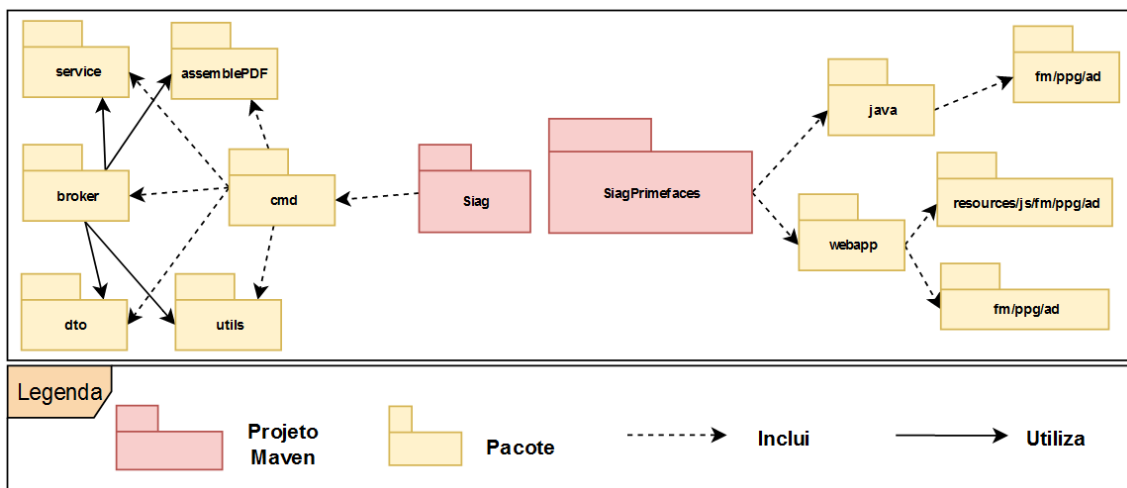


Figura 4.16: Diagrama de pacotes

Na tabela 7.13 encontra-se descrito as classes correspondente a cada pacote da Figura 4.16.

Para a geração automática das classes do pacote *dto* e *service*, foi utilizado o gerador de código Java WSDL2Java, que requer um documento WSDL. Do pacote *service*, utilizado para configurar a ligação entre a entidade integradora e a AMA, destaca-se a classe `BasicHttpBinding_SCMDServiceImpl.java`. Na listagem 6, encontra-se uma representação da configuração da ligação HTTPS ao endereço da AMA, para utilização do serviço Chave Móvel Digital, e a respetiva autenticação da SIAG, cujos dados são fornecidos pela AMA.

Listagem 6: Ligação e autenticação ao serviço Chave Móvel Digital

```

1 private final BasicHttpBinding_SCMDServiceStub b;
2 public BasicHttpBinding_SCMDServiceImpl(){
3     try {
4         b = new BasicHttpBinding_SCMDServiceStub(new URL("https://preprod.cmd.autenticacao.gov.pt/Ama.
5             Authentication.Frontend/SCMDService.svc"),new SCMDService_ServiceLocator());
6     } catch (AxisFault e) {
7         throw new ReturnObjectException().addMessage(IConstants.MENSAGEM_SIAG_ERRO_APLICACAO, new
8             String[] { "SCMDService() error." });
9     } catch (MalformedURLException e) {
10        throw new ReturnObjectException().addMessage(IConstants.MENSAGEM_SIAG_ERRO_APLICACAO, new
11            String[] { "SCMDService() error." });
12    }
13 }
14 public void authenticate (String username, String password){
15     b.setUsername(username);
16     b.setPassword(password);
17 }

```

Também do mesmo pacote destaca-se a classe `BasicHttpBinding_SCMDServiceStub.java`, que cria um pedido SOAP, especifica qual a operação, URL, parâmetros, cabeçalhos, entre outros... encapsula-os num envelope SOAP XML e invoca a operação.

Listagem 7: Pedido SOAP à operação `GetCertificate` do serviço Chave Móvel Digital

```

1  @Override
2  public java.lang.String getCertificate (byte[] applicationId , java.lang.String userId) throws java.rmi.
   RemoteException {
3      if (super.cachedEndpoint == null) {
4          throw new org.apache.axis.NoEndPointException();
5      }
6      org.apache.axis.client.Call _call = createCall ();
7      _call . setOperation ( _operations [1]);
8      _call . setUseSOAPAction(true);
9      _call . setSOAPActionURI("http://Ama.Authentication.Service/SCMDService/GetCertificate");
10     _call . setEncodingStyle( null );
11     _call . setProperty (org.apache.axis.client.Call.SEND_TYPE_ATTR, Boolean.FALSE);
12     _call . setProperty (org.apache.axis.AxisEngine.PROP_DOMULTIREFS, Boolean.FALSE);
13     _call . setSOAPVersion(org.apache.axis.soap.SOAPConstants.SOAP11_CONSTANTS);
14     _call . setOperationName(new javax.xml.namespace.QName("http://Ama.Authentication.Service/", " GetCertificate "
   ));
15     setRequestHeaders( _call );
16     setAttachments( _call );
17     try {
18         java.lang.Object _resp = _call . invoke(new java.lang.Object[] {new String( applicationId , StandardCharsets
   .UTF_8), userId});
19
20         if ( _resp instanceof java.rmi.RemoteException) {
21             throw (java.rmi.RemoteException)_resp;
22         }
23         else {
24             extractAttachments ( _call );
25             try {
26                 return (java.lang.String) _resp ;
27             } catch (java.lang.Exception _exception) {
28                 return (java.lang.String) org.apache.axis . utils . JavaUtils . convert( _resp , java.lang.String . class )
   ;
29             }
30         }
31     } catch (org.apache.axis.AxisFault axisFaultException) {
32         axisFaultException . printStackTrace ();
33         throw axisFaultException ;
34     }
35 }
36 }

```

O pacote *assemblePDF* é responsável pelo tratamento da assinatura no documento PDF final. Para isso, são necessárias as 3 classes contidas.

A primeira classe têm como objetivo preparar o documento a ser assinado, gerando um *hash*, para que a AMA, quando o receba, devolva a assinatura correspondente ao *hash*. Primeiramente, o documento, passado como parâmetro da função, é assinado com o certificado do utilizador. A geração do *hash* do documento deve ser feita conforme o “PKCS 1: RSA Cryptography Specifications Version 2.2”⁸. De seguida, na linha 10 é aplicado o algoritmo de *hash* SHA.256 ao documento a assinar “em branco”. Após, a geração do *hash* do documento é anexado o prefixo correspondente ao algoritmo de encriptação, representado pela linha 13. No final, há um documento encriptado e assinado com o certificado do utilizador, com o prefixo do algoritmo de encriptação usado, que será enviado para a AMA, para gerir a assinatura correspondente. Apesar deste documento estar assinado pelo Cidadão, a assinatura não é considerada válida por não ser autenticada pela entidade que emitiu o certificado, neste caso a AMA.

⁸ponto 9.2 até ao passo 2: <https://tools.ietf.org/html/rfc8017page-45>

Listagem 8: Geração de *hash*

```

1 BouncyCastleDigest digest = new BouncyCastleDigest();
2 PdfPKCS7 sgn = new PdfPKCS7(null, _certificates, DigestAlgorithms.SHA256, null, digest, false);
3 String hashAlgorithm = DigestAlgorithms.SHA256; // "SHA-256";
4 MessageDigest md = digest.getMessageDigest(hashAlgorithm);
5
6 byte [] hashedDocument = DigestAlgorithms.digest(data, md);
7 byte [] hashedPKCS7 = sgn.getAuthenticatedAttributeBytes(hashedDocument, PdfSigner.CryptoStandard.CMS,
8     citizenCertificate, null);
9
10 InputStream i = new ByteArrayInputStream(hashedPKCS7);
11 byte [] docBytesHash = DigestAlgorithms.digest(i, md);
12 ByteArrayOutputStream outputStream = new ByteArrayOutputStream();
13 outputStream.write(_sha256SigPrefix);
14 outputStream.write(docBytesHash);
15 byte [] temporary_document_without_signature [] = outputStream.toByteArray();

```

A classe "InjectAmaSignatureContainer" é o responsável pela descriptação do documento e a inserção da assinatura enviada pela AMA. Na linha 4 da listagem 9, é inserida a assinatura do utilizador, gerada pela AMA. Depois, é descriptado o documento *hashed*, utilizando o mesmo padrão de encriptação e os mesmos parâmetros do método `getAuthenticatedAttributeBytes` usado na listagem 8. No final, resulta o documento com a assinatura do utilizador, já validada pela AMA.

Listagem 9: Assinatura de documento

```

1 BouncyCastleDigest digest = new BouncyCastleDigest();
2 PdfPKCS7 sgn = new PdfPKCS7(null, _certificates, DigestAlgorithms.SHA256, null, digest, false);
3 // set the signature bytes
4 sgn.setExternalDigest(_signature, null, "RSA");
5 // call GetEncoded with the same parameters as the original GetAuthenticatedAtt ...
6 byte [] document_signed = sgn.getEncodedPKCS7(_documentHash, PdfSigner.CryptoStandard.CMS, null, null, null);

```

Como responsável pela estilização da assinatura, preparação do documento para ser *hashed* e, posteriormente, assinado, há a classe `PDFSigningManager.java`. Como demonstrado na Figura 7.8, a assinatura do utilizador tem determinados dados presentes na assinatura, incluindo informações pessoais do assinante, data e hora da assinatura, e logotipo da Entidade que o determinado Cidadão representa. A classe `PDFSigningManager.java` contém a lógica de obtenção de dados de um certificado (para obter o nome próprio e número de identificação, por exemplo), configuração das dimensões dos dados e da própria assinatura, assim como da sua localização no documento.

Esta classe invoca o método de assinatura da classe `BlankSignatureContainer.java`, após a chamada à operação `GetCertificate`, recebe o documento *hashed* e envia-o para o *broker*. Também invoca o método de assinatura da classe `InjectAmaSignatureContainer.java`, após a validação do OTP inserido pelo utilizador.

A classe `SCMDBrokerService` do pacote *broker* é o intermediário entre o módulo `siagPrimeFaces` e o serviço de assinatura por Chave Móvel Digital. Para isso:

- Recebe os dados submetidos pelo utilizador e encripta-os, utilizando o pacote *utils*;
- Envia-os para a classe `BasicHttpBinding_SCMDServiceImpl.java`, de forma a invocar as operações da AMA, especificadas pelas classes do pacote *service*;
- Recebe respostas da AMA, conforme a operação, em formato definido pelas classes do pacote *dto*, e envia os dados necessários, para a classe que, inicialmente, invocou o *broker* através do ActiveMQ, neste caso a classe `DocumentosDespesaController.java`.

Relativamente ao módulo `siagPrimeFaces`, usa a mesma estrutura e padrão de desenvolvimento *web* descrito na iteração anterior. O modelo `DocumentosDespesaController.java`, responsável pela *interface* apresentada ao utilizador, envia os pedidos ao módulo `siag`, através do *middleware* `ActiveMQ` e aguarda a resposta do *broker*.

4.2.4 Demonstração

De seguida encontra-se a *interface* com o utilizador correspondente ao caso de uso representado pela Figura 4.12.

Modo de assinatura: CMD CC

Número de telemóvel: *

PIN *

Assinatura visível

Localização

Razão

Escolha a posição da assinatura no documento *

Página

Figura 4.17: Interface para parametrizar assinatura - Chave Móvel Digital

- **Modo de assinatura:** Componente do tipo "SelectOneRadio", em que o utilizador só pode escolher um elemento da lista apresentada. Este *input* especifica qual é o tipo de assinatura escolhida. Conforme a escolha, o formulário abaixo é atualizado;
- **Número de telemóvel:** Componente do tipo "InputText", em que permite ao utilizador inserir texto. Este *input* serve para introduzir a número de telemóvel;
- **PIN:** Componente do tipo "Password", em que permite ao utilizador inserir texto, oculto por *bullet points*. Este *input* serve para introduzir o PIN de assinatura associado ao número de telemóvel inserido anteriormente;

- **Assinatura visível:** Componente do tipo "SelectBooleanCheckbox", em que só pode ter o valor de "Verdadeiro" ou "Falso". O utilizador decide se quer que a assinatura seja visível no documento. Caso seja visível, aparecem os seguintes *inputs*:
 - **Localização:** Componente do tipo "InputText", em que permite ao utilizador inserir texto. Este *input* serve para introduzir a localização de onde está a assinar, sendo opcional no processo de assinatura;
 - **Escolha a posição da assinatura no documento:** Componente do tipo "SelectOneButton", em que só um botão pode ser selecionado. Este *input* serve para indicar onde é que a assinatura vai ser introduzida no documento, usando uma tabela de 6 linhas por 4 colunas para simular o documento;
 - **Página:** Componente do tipo "Spinner", que permite o acréscimo ou decréscimo de um determinado valor. No contexto deste formulário, tem o propósito de indicar qual é a página em que o utilizador quer inserir a assinatura;
- **Preview:** Componente do tipo "Button". Serve para o utilizador ver como fica a assinatura no documento, antes de proceder à assinatura através da AMA;
- **Avançar:** Componente do tipo "Button", para começar o processo de tratamento dos dados inseridos e envio para a AMA.

Após a introdução dos parâmetros necessários para a assinatura através da Chave Móvel Digital, o utilizador pode clicar no botão "Avançar". Nesse momento, é feita a validação dos dados inseridos e, caso estejam corretos, o processo de assinatura avança para a próxima etapa. O utilizador recebe um SMS com o código OTP e o processo de assinatura digital avança para a próxima fase, representado pela Figura 7.7.



Figura 4.18: Interface de inserção do OTP - Chave Móvel Digital

- **One-Time-Password:** Componente do tipo "InputMask", em que o utilizador é forçado a inserir texto num determinado padrão. Escolheu-se este tipo de dados por o código OTP só ser composto por números e ter um tamanho definido de 6 dígitos, de forma a prevenir erros do utilizador na introdução do One-Time-Password.
- **Assinar:** Componente do tipo "Button", para efetuar a assinatura do documento.

4.2.5 Avaliação

Como definido na secção 4.2.2, havia três requisitos a cumprir quanto à implementação do serviço de assinatura digital com Chave Móvel Digital, desenvolvido pela AMA. Para a solução ser considerada viável, tinha de ser adjetivada como legal, interoperável e usável.

Relativamente ao primeiro requisito não-funcional, para ser integrado um serviço externo desenvolvido por outra entidade, é necessário cumprir os requisitos impostos pela mesma. Alguns desses requisitos eram a empresa integradora não guardar os dados introduzidos pelo utilizador para assinar um documento e a introdução do PIN ser feita de forma anónima.

1. **”Não guardar nos logs os dados do PIN ou Pin temporário-** Todo o processo de assinatura não armazena dados. Os dados pessoais do utilizador (como número de telemóvel e PIN) são encriptados e enviados para a AMA e os dados da assinatura (como razão e localização, posição da assinatura e número da página) são processados e também enviados para a AMA. Em qualquer momento, estes dados não são guardados no sistema da SIAG;
2. **”Não apresentar visível para o utilizador os dígitos do PIN da CMD -** O componente UI utilizado para a inserção do PIN oculta os dados inseridos nesse *input*. Desta forma, um utilizador que olhe para a *interface* enquanto o PIN esteja a ser introduzido, os dígitos serão mantidos em anonimato, preservando a privacidade do Cidadão utilizador do serviço.

Quanto ao requisito de interoperabilidade, era obrigatório a solução contemplar o protocolo de transferência HTTPS e utilizar mensagens SOAP para envio de dados, imposto pela AMA para integração do seu serviço de Chave Móvel Digital para assinatura de documentos. Para isso, e como se pode observar na listagem 6, na linha 4 é feita a ligação ao serviço através do *URL* `”https://preprod.cmd.autenticacao.gov.pt /Ama.Authentication.Frontend/SCMDService.svc”`), que utiliza como protocolo de transferência HTTPS e cuja autenticação é feita na linha 13 e 14. Também, na listagem 7, é possível observar como é que é encapsulado um pedido SOAP XML e invocar uma operação do serviço Chave Móvel Digital. Assim, por o processo de assinatura digital estar funcional, os requisitos de interoperabilidade são cumpridos.

O último requisito não-funcional definido para aprovar esta solução, semelhantemente à última iteração, é usabilidade. Estando o serviço integrado, é crucial haver uma *interface* intuitiva e fácil para os utilizadores poderem fazer uso do serviço de Chave Móvel Digital. O processo de assinatura digital tem de ser claro e cumprir as orientações definidas pela AMA. Algumas destas orientações eram:

- **”Apresentar sempre o documento a assinar pelo cidadão.-** Como se pode observar na Figura 7.6, à esquerda do formulário para introdução dos dados para a assinatura, está o documento a assinar;
- **”Identificar e informar sobre os vários passos do processo de assinatura [cf. secção 3.2.4 da POL16] à medida que os mesmo ocorrem.-** Acima do documento a assinar e do respetivo formulário de assinatura, encontra-se os vários passos do processo de assinatura.

Em suma, o processo de assinatura digital por Chave Móvel Digital, em ambiente de pré-produção, foi concluído com sucesso. Os objetivos definidos na secção 4.2.2 foram cumpridos, assim como as orientações da AMA.

No momento da escrita desta tese, esta solução está pronta para ser avaliada por um representante da AMA, para que possa proceder para ambiente de produção e utilização dos clientes da SIAG.

4.2.6 Comunicação

A assinatura digital de documentos é uma realidade cada vez mais comum na vida dos Cidadãos. Um em cada cinco portugueses utiliza a Chave Móvel Digital como meio de autenticação em plataformas e como meio de assinatura de documentos. Este serviço, ao contrário da iteração anterior, não requer um *hardware* específico para assinar documentos, como é o caso do SDK que requer um leitor de cartões. Para uso da Chave Móvel Digital, basta um telemóvel que todos os Cidadãos têm. Assim, a solução proposta nesta iteração tenciona complementar os utilizadores que não queiram assinar documentos, recorrendo ao Cartão de Cidadão, aumentando assim o número de clientes da SIAG, interessados neste serviço.

Para além do serviço de assinatura digital, a Chave Móvel Digital também permite a autenticação de Cidadãos em diversas plataformas. Tendo a SIAG já estreitado contactos com a AMA, integrando um primeiro serviço, a integração do serviço de autenticação é uma possibilidade real para a SIAG, na medida em que pode substituir a autenticação por nome de utilizador e *password*, por Chave Móvel Digital, acrescentando um fator extra de segurança ao processo de autenticação.

4.3 Iteração 3: Integração de Sign'Stash

4.3.1 Identificação do Problema

A SIAG tem implementada uma forma de assinatura digital de faturação eletrónica, utilizando o seu próprio certificado. Devido ao Decreto-Lei 28/2019, é necessário que o certificado utilizado seja um selo emitido por uma Autoridade Certificadora.

Surge assim a proposta oferecida pela Sign'Stash da Multicert, que contempla a assinatura de faturação eletrónica. Com a solução de assinatura para faturação eletrónica, a Multicert tem uma resposta rápida e segura às necessidades regulatórias atuais. Para isso inclui o serviço de assinatura Sign'Stash. Esta oferta permite uma multiplicidade de funcionalidades que vão desde uma simples assinatura digital (numa fatura ou qualquer tipo de documento eletrónico), ao envio do documento assinado até à sua preservação na área de segurança digital.

4.3.2 Definição de objetivos

O serviço Sign'Stash fornece uma API REST, em ambiente de pré-produção, para que clientes integrem as suas operações, nomeadamente a de assinatura digital. Para isto, é necessário;

- Estabelecer *handshake* SSL/TLS;
- Autenticação OAuth2 com credenciais fornecidas pela Multicert;
- Pedido HTTP POST `"/oauth/token"` para obter *token*;
- Pedido POST `"/signstash/einvoice-integration-ws/api/v0/document/sign/base64/"` de assinatura utilizando JSON;
- Integrar numa *interface*.

Assim, é desejado que a SIAG sirva de intermediária entre o cliente e a Multicert. Para isso, todo e qualquer cliente que queira assinar uma fatura eletrónica de acordo com as imposições legais, utilizará o selo qualificado da SIAG, certificado e produzido pela Multicert, para a validação da mesma.

De forma a poder avaliar a viabilidade da solução, é definido um requisito não-funcional:

- **Interoperabilidade** - Capacidade de um sistema comunicar com outro sistema, utilizando formatos de dados e protocolos específicos;

4.3.3 Design e desenvolvimento

Atributos qualitativos

A tabela 7.17 descreve o cenário concreto para o atributo de qualidade que deve estar incluído na solução.

O diagrama 4.19 representa os principais componentes da solução proposta no sistema geral e as respetivas ligações.

Para esta solução existem quatro entidades envolvidas:

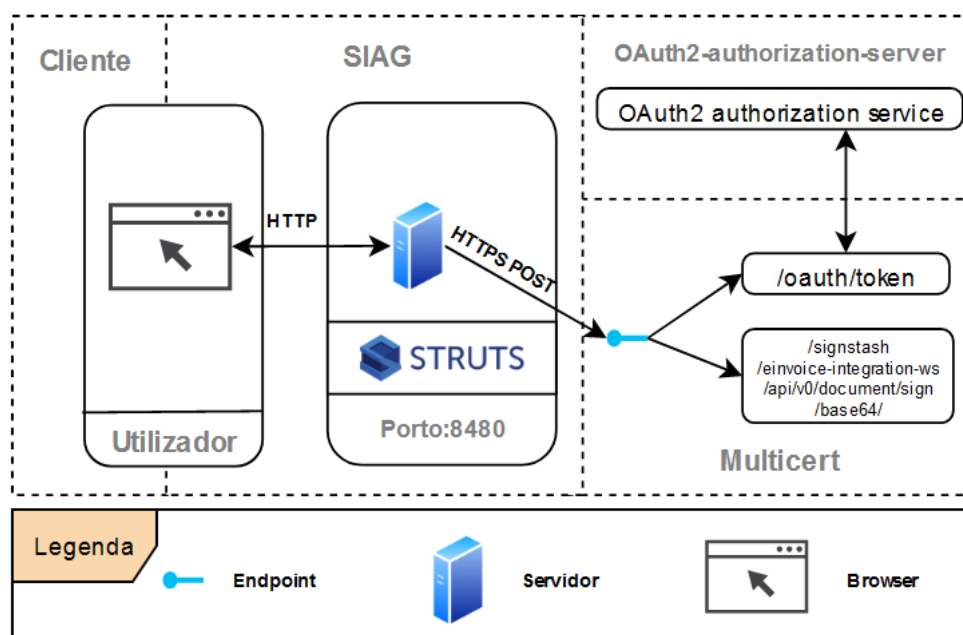


Figura 4.19: Vista geral da arquitetura da solução

- O utilizador, que comunica à SIAG que quer assinar um documento;
- A SIAG, que comunica por pedido HTTP POST com a Multicert;
- A Multicert, que comunica com a OAuth2 para a obtenção de um *token*, e que assina um documento;
- OAuth2, que contém um servidor para autenticação e geração de *tokens* de acesso;

A proposta de integração com o serviço Sign'Stash da Multicert é feita através do protocolo de transporte HTTPS. Para isso, são efetuados pedidos POST às operações fornecidas pelo Sign'Stash. Para obtenção do *token* que permite realizar o pedido de assinatura digital de uma fatura, é necessário autorizar o utilizador, recorrendo à *framework* do OAuth2, com as credenciais fornecidas pela Multicert, a ter acesso a recursos protegidos.

Visão de casos de uso

O diagrama 4.20 mostra o caso de uso implementado na solução proposta para a integração do serviço Sign’Stash da Multicert.

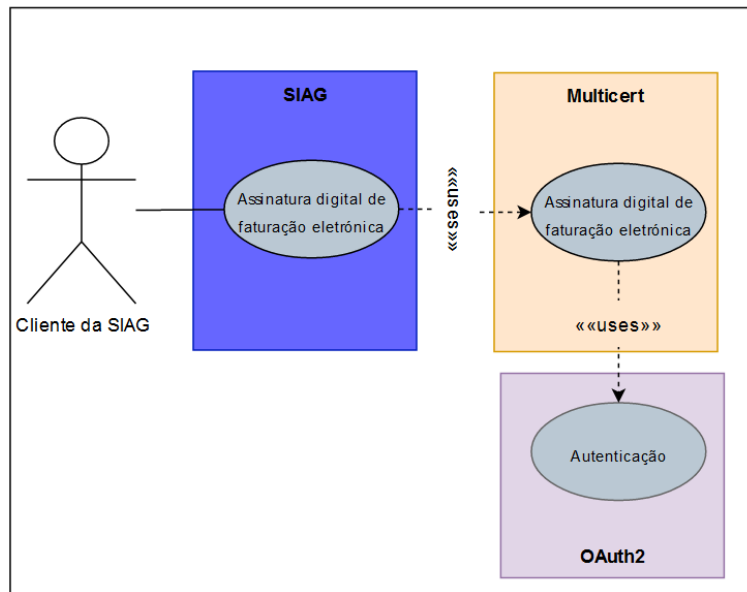


Figura 4.20: Diagrama de caso de uso - Assinatura digital de faturação eletrónica

Vista lógica

O diagrama 4.21 é representativo de como está organizada a solução. Como se pode observar, esta arquitetura assemelha-se à arquitetura cliente-servidor, em que há um cliente que faz pedidos, neste caso o sistema da SIAG (Que será desdobrado de seguida), e o servidor da Multicert, que responde a esses mesmos pedidos, e utiliza OAuth2 para autorizar utilizadores a acederem aos seus recursos protegidos. As operações fornecidas pela Multicert são disponibilizadas através de uma arquitetura REST. Como descrito na secção 2.1.4, utiliza como protocolo de transporte HTTP (Por cima utiliza o protocolo de segurança SSL) e usa como formato de representação de dados JSON.

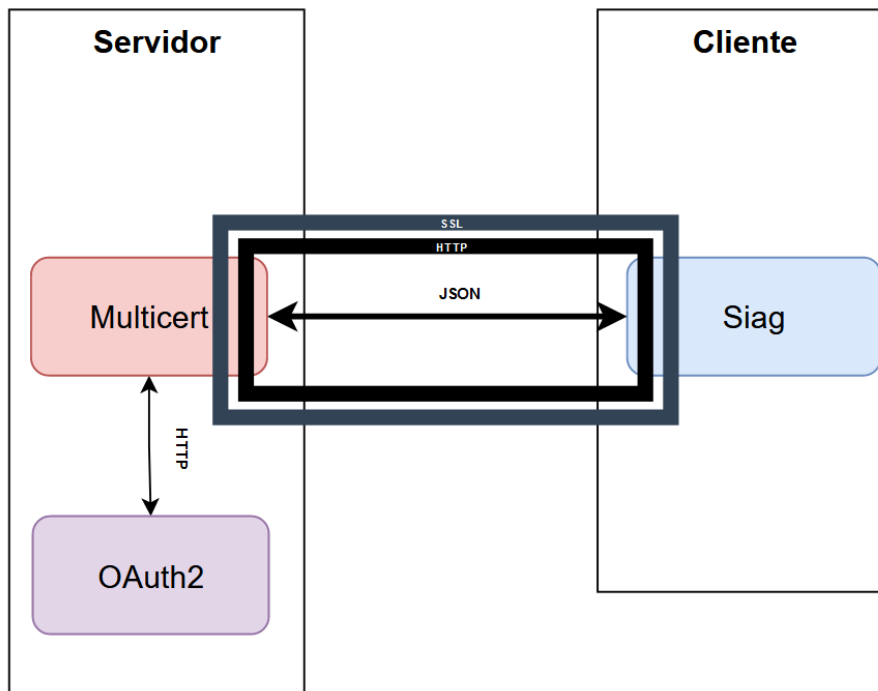


Figura 4.21: Arquitetura técnica da solução: cliente-servidor

Como outra solução de integração de serviços externos, a arquitetura da mesma assemelha-se a cliente-servidor, por haver um cliente que efetua pedidos a uma entidade externa, como um servidor. O servidor corresponde ao servidor da Multicert e a *framework* de autorização OAuth2, que a partir de credenciais, gera um *token* para que o utilizador tenha acesso a recursos protegidos como o *endpoint* de assinatura da Multicert.

O cliente Siag é o módulo que contém, como visto anteriormente, a lógica de integração com outros serviços, assim como o seu próprio *frontend*. Para isso, utiliza a *framework* Struts, baseada no padrão Model-View-Controller e páginas *web* JSP.

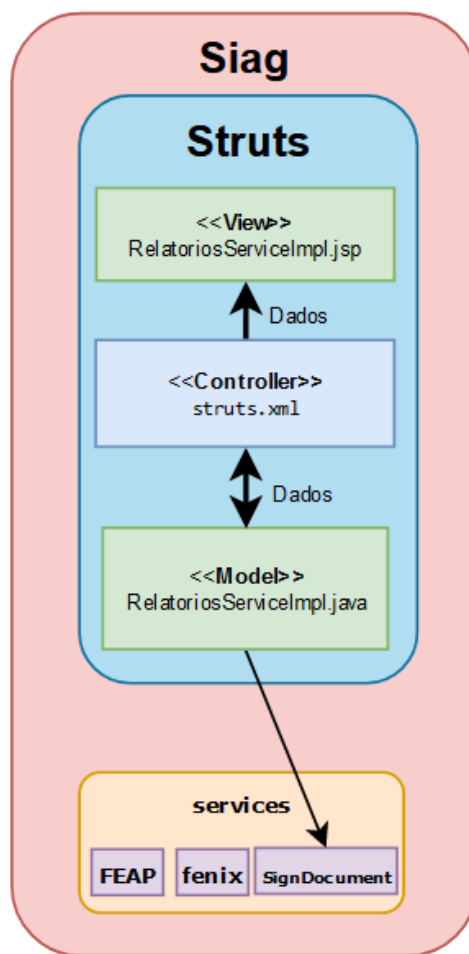


Figura 4.22: Arquitetura técnica do módulo Siag

O pedido de assinatura de faturação electrónica pelo utilizador na vista `RelatoriosServiceImpl.jsp`, é lido pelo respetivo controlador e direccionado para o modelo de dados respetivo, neste caso a classe `RelatoriosServiceImpl.java`. Esta classe invoca os serviços de assinatura integrados em "SignDocument", onde, para além de estar o serviço da Chave Móvel Digital, também está o serviço Sign'Stash da Multicert.

Vista de processo

Como é possível observar na Figura 4.3.4, o utilizador pode assinar uma fatura, mas não terá a opção de parametrizar a assinatura, por esta ser invisível, por decisão da empresa. Então, este processo de assinatura de uma fatura será o mais simples das três iterações desenvolvidas, devido à menor possibilidade de erros no processo de assinatura.

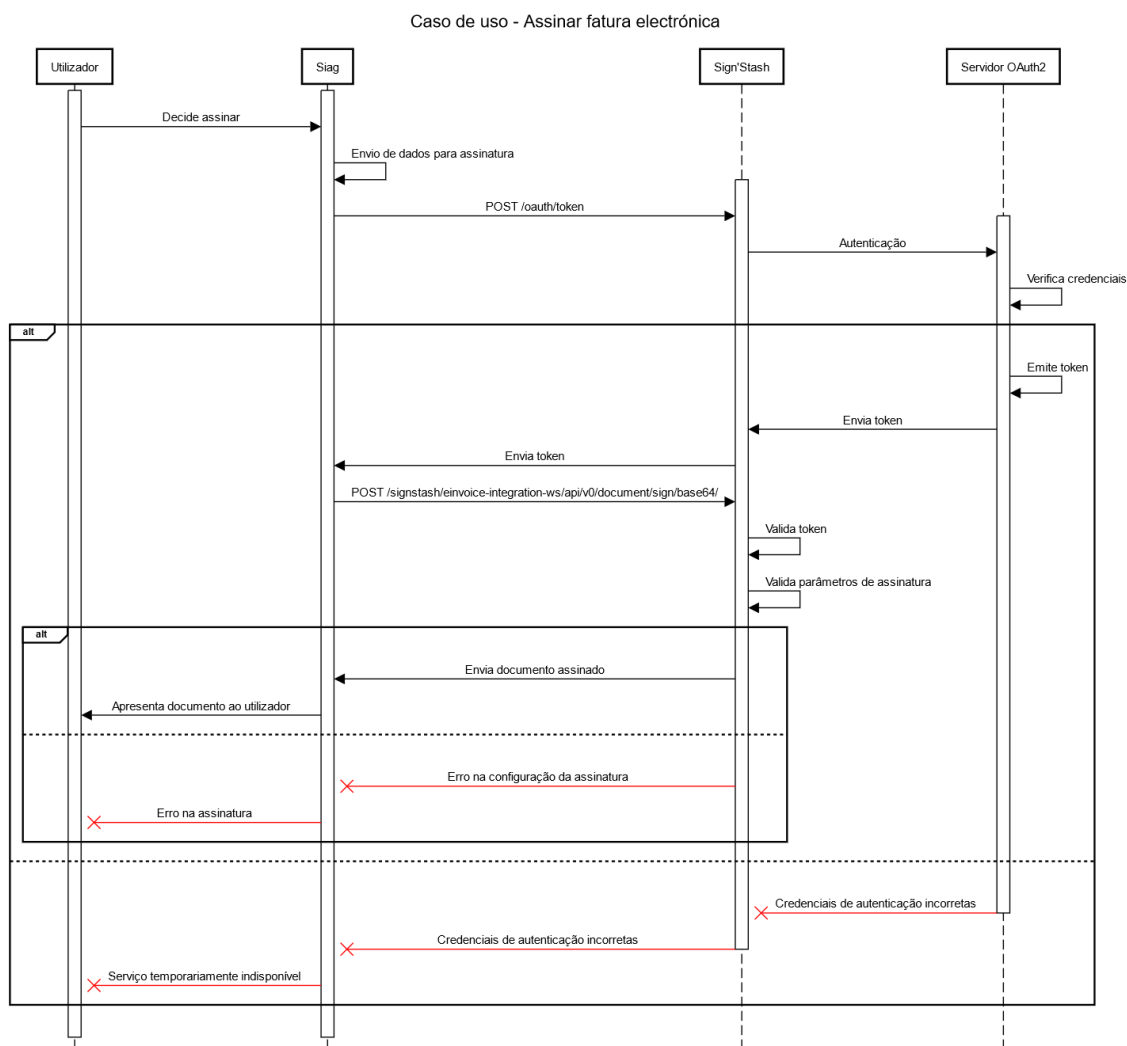


Figura 4.23: Diagrama de sequência - Assinar fatura electrónica

O utilizador ao imprimir uma fatura, terá a opção de a assinar. Para isso, clicará no botão respetivo, definido pela coluna "ASD". Aquando da impressão, a fatura já virá assinada, em que a fatura é invisível.

No momento em que o utilizador imprime a fatura, a Vista associada acede aos reencaminhamentos definidos no Controlador e invoca as funções no Modelo RelatoriosServiceImpl.java, seguindo o padrão MVC. Esta classe invoca o serviço de assinatura da Sign'Stash. O serviço de assinatura desenvolvido pela Multicert requer as credenciais do utilizador, para poder utilizar o seu certificado na assinatura. Cada cliente tem as suas configurações definidas e guardadas no repositório da SIAG, por isso, estes dados são parametrizados de forma automática.

De seguida, o módulo Siag, que contém o *broker* responsável pela invocação dos serviços da Multicert,

faz um pedido HTTP POST `"/oauth/token"` com as credenciais do utilizador. Sign'Stash utiliza a *framework* OAuth2, que contém um servidor para verificação das credenciais e envio do respetivo *token* que dá acesso ao serviço de assinatura do Sign'Stash. Caso as credenciais do utilizador não sejam válidas, o mesmo é notificado e o processo termina.

Após a invocação da operação `"/oauth/token"`, o *token* de acesso ao utilizador é enviado para o *broker* e utilizado na operação seguinte: `"/signstash/einvoice-integration-ws/api/v0/document/sign/base64/"`. Esta operação é feita através de um pedido HTTP POST, em que as configurações⁹ da assinatura são enviadas. Por a assinatura ser invisível, os parâmetros enviados são sempre os mesmos, exceptuando o documento a assinar e o *token* do utilizador. O pedido é submetido pelo *broker* e enviado para o Sign'Stash. Este verifica se o *token* é válido, ou seja, se foi emitido pelo OAuth2 e não está expirado (um *token* tem um período máximo de 1 hora para ser utilizado). Após esta verificação, também valida os parâmetros de configuração da assinatura. Caso estas verificações sejam válidas, o documento é assinado e retornado para o *broker*, que notifica o Modelo que efetuou o pedido e apresenta na Vista o mesmo. Caso o *token* tenha expirado ou não seja válido, assim como a configuração da assinatura não seja aplicável a este tipo de documento, o utilizador é notificado e o processo de assinatura termina.

Vista física

Este tipo de vista tem como objetivo mostrar como é que os elementos de um sistema estão configurados em tempo de execução e como é que é feita a modelação dos aspetos físicos de um sistema orientado a objetos. Semelhantemente à iteração anterior, esta solução resulta da integração de um serviço externo no sistema ERP da SIAG. Assim, este tipo de vista não é aplicável a esta solução, por não ser necessária a integração de sistemas físicos para a invocação de serviços como o do Sign'Stash da Multicert.

Vista de desenvolvimento

O primeiro objetivo definido era o estabelecimento de uma conexão SSL/TLS entre as duas entidades. Como referido na secção 7.1.8, o *handshake* SSL/TLS consiste em seis passos para que a troca de informação seja feita de forma segura e autêntica.

Este algoritmo de estabelecimento de conexão é feito de forma automatizada pelas bibliotecas do Java, contudo, é necessário que no momento do pedido HTTP, o cliente já tenha o certificado SSL/TLS do servidor na sua *truststore*¹⁰. Este certificado pode ser obtido através do "cadeado" presente na "address bar" do *website* do servidor.

⁹Parametrizações feitas pelo cliente no produto da SIAG para automatizar processos

¹⁰Repositório que contém certificados de outras entidades, que podem ser confiadas para troca de dados

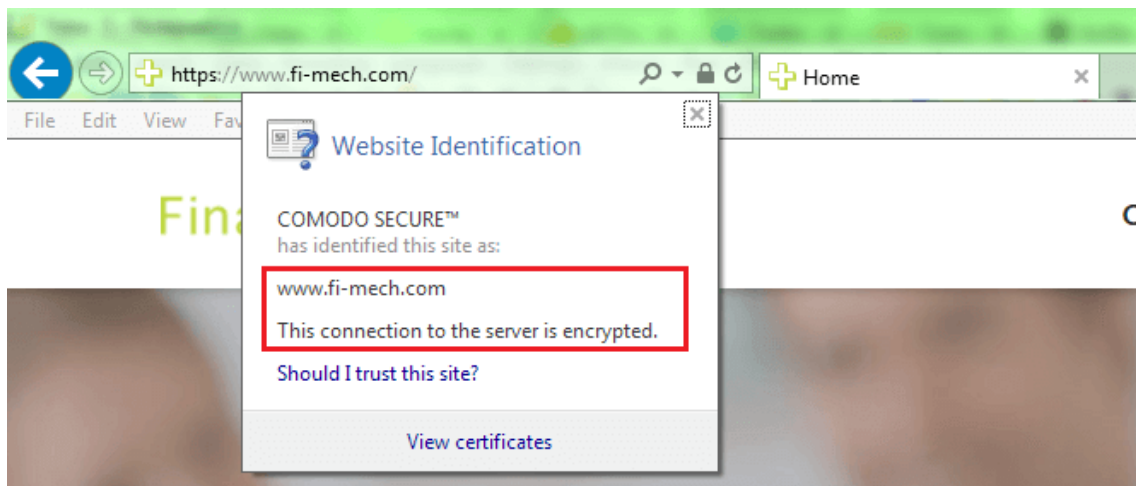


Figura 4.24: Obtenção de certificado SSL/TLS de servidor (Security, 2017)

Para importar o certificado para a *truststore*, é necessário o seguinte comando:

Listagem 10: Importar certificado SSL/TLS para *truststore*

```
keytool -importcert -file "localizacao_certificado" -keystore "localizacao_truststore" -alias "alias"
```

Quanto à integração da assinatura de uma fatura, através do serviço Sign’Stash da Multicert, o diagrama 4.25 representa os pacotes de desenvolvimento. Os pacotes a vermelho são os módulos maiores, projectos maven, com as suas próprias configurações de execução e *deployment*, e os seus respetivos pacotes.

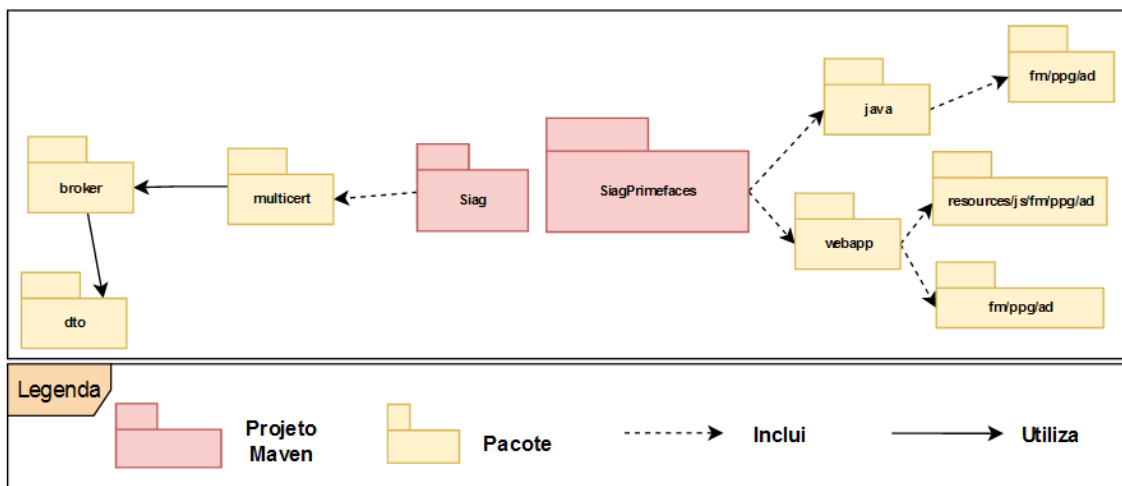


Figura 4.25: Diagrama de pacotes

Na tabela 7.19 encontra-se descrito o *broker* MULTICERTBrokerService.java correspondente ao pacote *broker* na Figura 4.25.

O pacote *dto* contém um conjunto de classes necessárias para a construção do pedido HTTP POST para a assinatura de uma fatura. Para o envio de dados, utilizou-se o formato de representação de dados JSON. Serve o diagrama 4.26 para estruturar os elementos que compõem o corpo da mensagem.

Na tabela 7.20 encontra-se descrita a estrutura de classes que compõem o corpo da mensagem JSON enviado no pedido de assinatura de uma fatura.

Como explicado na Secção 2.1.4, para a comunicação com a arquitectura REST, utiliza-se o protocolo

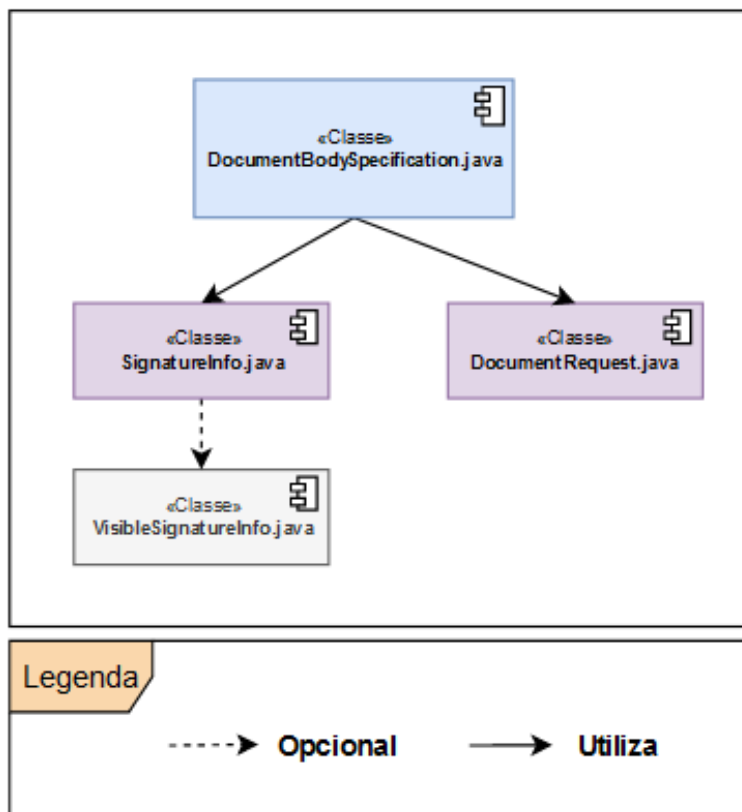


Figura 4.26: Diagrama de pacote dto

de transporte HTTP e um formato para a representação de dados, contido num pedido HTTP. Para o desenvolvimento desta solução, escolheu-se o formato de dados escolhido JSON, por a quantidade de dados enviados ser baixa e ser uma estrutura de dados simples.

A figura 4.26 representa como é que os dados são estruturados e convertidos para JSON, no corpo de uma mensagem HTTP para o *endpoint*

"/signstash/einvoice-integration-ws/api/v0/document/sign/base64/".

O elemento `DocumentBodySpecification` contém os sub-elementos `SignatureInfo` e `DocumentRequest`: o primeiro sub-elemento especifica qual é o tipo de assinatura (PAdES ou XAdES) e a localização e razão para a assinatura da fatura; o segundo sub-elemento contém o documento em Base64, assim como o seu tipo e nome. Por definição, a assinatura é invisível, mas, caso no futuro, seja do interesse da SIAG, tornar a assinatura visível, é necessário incluir o elemento `VisibleSignatureInfo`, com configurações do estilo da assinatura, como a área, posição da assinatura, entre outros...

De seguida encontra-se um exemplo do corpo de uma mensagem HTTP para a assinatura de uma fatura, conforme a especificação acima descrita.

Listagem 11: Especificação JSON para assinatura de documento (Sign'Stash, 2022)

```

1 {
2   "documentRequest": {
3     "base64Content": "documento_base64",
4     "contentType": "application /pdf",
5     "externalReference": "id",
6     "name": "nome_documento.pdf"
7   },
8   "signatureInfo": {
9     "applyTimestamp": true,
10    "location": "Test API location",
11    "reason": "Test API reason",
12    "signatureType": "PAdES",
13  }
14 }

```

O *broker* MULTICERTBrokerService.java sendo o intermediário entre os pedidos feitos e o Sign'Stash, é responsável pela construção do corpo da mensagem enviada para o *endpoint* e pelo envio da mesma. Segue a listagem ?? para exemplificar a construção do pedido HTTP e envio do mesmo.

Listagem 12: Pedido POST HTTP genérico

```

1 CloseableHttpClient httpClient = null;
2 try {
3   // Request
4   httpClient = createHttpClient ();
5   HttpPost request = new HttpPost( url);
6
7   // Headers
8   for (Entry<String, String> entry : headers.entrySet ()) {
9     request.addHeader(entry.getKey(), entry.getValue ());
10  }
11  // Body
12  if (body != null) {
13    ByteArrayEntity params = new ByteArrayEntity(body.getBytes("UTF-8"));
14    request.setEntity (params);
15  }
16  HttpResponse response = httpClient.execute (request);
17  ReturnObject ret = new ReturnObject();
18  ret.addRetorno(response.getStatusLine ().getStatusCode ());
19  ret.addRetorno(new org.json.JSONObject(EntityUtils.toString (response.getEntity ( ))));
20  return ret;
21 } catch (Exception e) {
22   throw new ReturnObjectException().addMessage(Iconstants.MENSAGEM_SIAG_ERRO_APLICACAO, new String[] {
23     "error response" + e.getMessage() });
24 } finally {
25   try {
26     if (httpClient != null) {
27       httpClient.close ();
28     }
29   } catch (Exception e) {
30     LoggerGEDI.error(LoggerGEDI.exceptionToString(e));
31   }

```

No listagem 12, é construído um pedido HTTP. Semelhantemente a um pedido SOAP, o pedido HTTP tem um cabeçalho e corpo da mensagem.

Na linha 5 é definido o método do pedido, neste caso é um pedido POST. Também é definido o URL para aceder a um determinado recurso. No contexto deste serviço é uma operação protegida por um *token*.

Na linha 9, é passado o cabeçalho do pedido: resultado da concatenação de "Bearer "mais *token* recebido da operação "/oauth/token".

Na linha 14, é adicionado ao pedido o elemento `DocumentBodySpecification` (passado como `String`, resultado da conversão para JSON), com codificação "UTF-8".

Na linha 18 e 19, é adicionado a um objeto genérico, que armazena dados de qualquer tipo, o código do resultado do pedido HTTP e a conversão do conteúdo da resposta de JSON para String.

Posteriormente, o objeto retornado é filtrado. Conforme o "status code", há duas opções: caso seja um código 200 significa que o pedido foi efetuado com sucesso e o documento foi assinado; caso não seja 200, significa que ocorreu um erro e uma exceção é lançada.

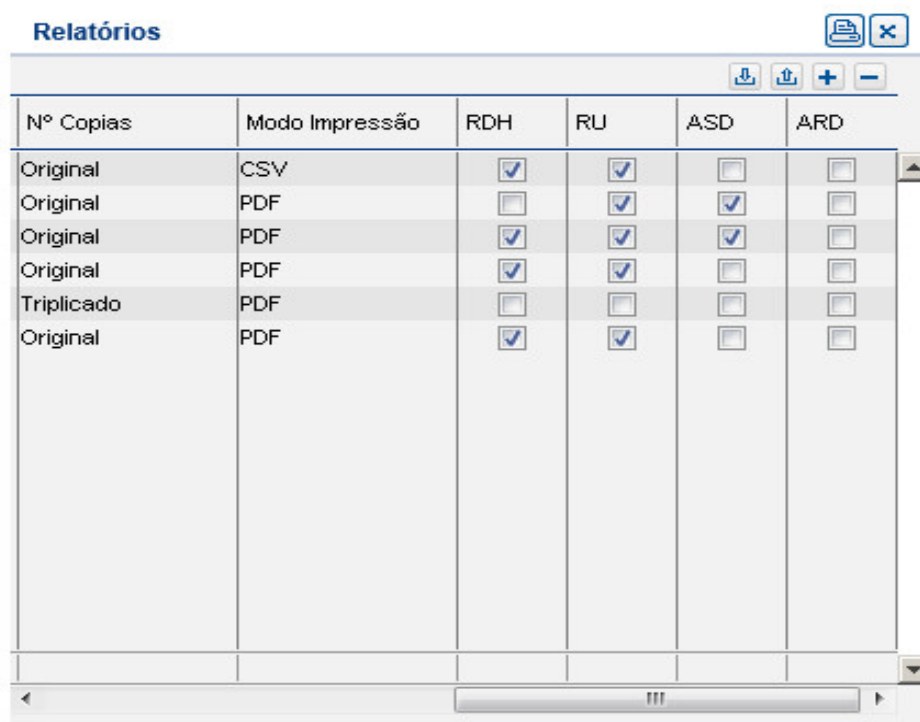
Segue o exemplo de uma mensagem de resposta de sucesso a um pedido de assinatura.

Listagem 13: Resposta HTTP a assinatura digital por Sign`Stash (Sign`Stash, 2022)

```
1 {  
2   "documentList":  
3     {  
4       "additionalContentLength": 11,  
5       "cipherStatus": "CIPHERED",  
6       "contentLength": 10,  
7       "contentType": "application /pdf",  
8       "externalReference": "id",  
9       "internalId": 123,  
10      "name": "nome_documento.pdf",  
11      "receptionDate": "string",  
12      "signError": "INVALID_CERT_SERVICE_CREDS",  
13      "signStatus": "SIGNED",  
14      "signedContent": "documento_assinado_base64",  
15      "storageEndDate": "string",  
16      "storagePeriod": 120,  
17      "storageStatus": "CONTENT_PRESERVED"  
18    },  
19    "status": "EXECUTED",  
20    "type": "SIGN"  
21 }
```

4.3.4 Demonstração

Na figura 4.27 encontra-se a *interface* desenvolvida para a impressão de faturas, assim como assinar digitalmente através do serviço Sign'Stash da Multicert.



Nº Copias	Modo Impressão	RDH	RU	ASD	ARD
Original	CSV	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Original	PDF	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Original	PDF	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Original	PDF	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Triplicado	PDF	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Original	PDF	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Figura 4.27: Interface UI para imprimir fatura - Sign'Stash

Cada linha da tabela: Relatório a assinar;

ASD: Opção para que o relatório seja assinado através do serviço Sign'Stash da Multicert;

Botão impressora: Botão para imprimir o relatório, após parametrizações.

4.3.5 Avaliação

Como definido na secção 4.3.2, era desejado que a SIAG servisse de intermediária entre um cliente seu que pretende assinar uma fatura, de acordo com as imposições legais atuais, e a Multicert que é uma entidade credenciada pelo GNS. Para isso, foi definido a integração do serviço externo Sign'Stash desenvolvido pela Multicert, para a assinatura de documentos, como faturas, em formato PDF ou XML. A viabilidade desta solução é avaliada com base no requisito não-funcional interoperabilidade. Para a integração do serviço Sign'Stash é necessário cumprir os requisitos de integração que o serviço impõe. Estes eram a comunicação ser feita sob o protocolo de transporte HTTP, com camada de segurança SSL/TLS, obtenção de *token* que autoriza o utilizador a realizar a assinatura de um documento e enviar o documento a assinar e as configurações de assinatura em JSON através de um pedido HTTP POST. Assim, a integração deste serviço foi realizada com sucesso, na medida em que o processo de assinatura encontra-se implementado e pronto para transitar para ambiente de produção. Para isto, é necessário mudar o *endpoint* de pré-produção para produção e comprar um selo qualificado remoto para assinar documentos, de forma legal.

4.3.6 Comunicação

A assinatura digital de documentos pode ser realizada de várias formas, como se pode observar pelas soluções desenvolvidas nas iterações anteriores. Contudo, a assinatura de documentos de relevância fiscal, como faturas eletrônicas, só pode ser legalmente validada caso seja credenciada por uma entidade da *Trusted List* do GNS.

A solução desenvolvida pela Multicert, denominada de Sign'Stash, contempla a assinatura digital de documentos, sejam de relevância fiscal ou não, para além do seu formato, seja em PDF ou XML. Esta diversidade de soluções que oferecem às empresas integradoras e, respetivamente, aos seus clientes, permite o desenvolvimento de um produto mais consolidado e com mais funcionalidades, ao nível da certificação de documentos.

Capítulo 5

Resultados

A assinatura digital é um processo complexo, que requer a análise de diversos critérios para uma correta e apropriada integração, de acordo com os requisitos do cliente. Critérios como o tipo de assinatura, o uso de diversas tecnologias e protocolos de segurança de forma a manter o anonimato e privacidade dos utilizadores e dos seus dados a serem transferidos entre dois sistemas independentes, a dificuldade no desenvolvimento da integração e requisitos do cliente, como a utilização de *hardware* adicional, o número de *inputs* necessários e a estilização da assinatura, definem se uma determinada solução é apropriada para o problema proposto a uma empresa integradora como a SIAG.

É o papel de um engenheiro informático analisar soluções e avaliar os seus aspetos positivos e negativos, de forma a enquadrar da melhor forma possível em sistemas complexos e variados, como o produto da SIAG.

Segue um conjunto de tabelas que resumem as iterações desenvolvidas, com base em critérios importantes para a escolha de um determinado serviço de assinatura digital.

Critério : Assinatura digital

Critérios/iterações		Integração de SDK do middleware versão 3 do Cartão de Cidadão	Integração de serviço Chave Móvel Digital	Integração de Sign'Stash
Assinatura Digital	Assina PDF	✓	✓	✓
	Assina XML	✓	✗	✓
	Assina faturas eletrónicas	✗	✗	✓

Figura 5.1: Comparação critério Assinatura digital

Todos os serviços de assinatura implementados assinam documentos PDF, sendo que é o formato mais comum de documentos. Quanto a assinar documentos de formato XML, só o *middleware* desenvolvido pela AMA e o serviço Sign'Stash permite assinar documentos deste tipo. Apesar de documentos em formato XML ser menos comum, é uma realidade em alguns sistemas que criam documentos, com a sua própria estilização, adicionando assim uma maior diversidade no tipo de documentos a assinar, comparativamente ao serviço da Chave Móvel Digital. Quanto a assinar faturas eletrónicas, é uma

funcionalidade bastante exigida no mercado, devido a uma recente alteração da regulamentação fiscal. Esta necessidade premente das entidades que necessitam de assinar faturas eletrónicas, tornam o serviço Sign'Stash uma solução bastante capaz e desejada, na medida em que é a única solução credenciada para tal.

Apesar do serviço Sign'Stash assinar documentos em PDF ou XML, assim como faturas eletrónicas, tem algumas desvantagens, que tornam a escolha das outras soluções possível, tendo em conta que este tipo de assinatura é invisível, pelo produto da SIAG. Para clientes que queiram a assinatura genérica estampada num documento, devem optar pelo SDK do *middleware* da AMA, caso queiram o seu logótipo na assinatura devem optar pelo serviço da Chave Móvel Digital, em que a assinatura é personalizada.

Critério : Nível de Segurança

Critérios/Iterações		Integração de SDK do middleware versão 3 do Cartão de Cidadão	Integração de serviço Chave Móvel Digital	Integração de Sign'Stash
Nível de Segurança	Protocolos	Baixo	Alto	Alto

Figura 5.2: Comparação critério Nível de Segurança

Para a avaliação deste critério, são analisados os protocolos de segurança utilizados para tornar a comunicação mais segura. Os dados de um utilizador necessários para a realização de uma assinatura, devem ser tratados de forma bastante segura, na medida em que a comunicação entre as duas entidades envolve a transmissão do ficheiro a assinar, assim como credenciais de autenticação do utilizador.

A primeira solução desenvolvida utiliza o protocolo de transporte HTTP para a transmissão de dados entre o servidor remoto da SIAG e o servidor local, com a biblioteca da AMA. Esta solução pode ser melhorada, cifrando a comunicação entre os dois servidores, como por exemplo a utilização de criptografia assimétrica, cifrando os dados com a chave pública da SIAG e decifrando no servidor remoto com a respetiva chave privada.

O serviço da Chave Móvel Digital utiliza o protocolo SOAP, que consiste na utilização do protocolo de transporte HTTPS para a comunicação entre a empresa integradora e a AMA. Como explicado anteriormente, esta comunicação é assegurada através da partilha de ambos os certificados. Desta forma, SOAP sob HTTPS, torna a transmissão de dados bastante segura.

O serviço Sign'Stash utiliza uma API REST para uso das suas funcionalidades e o mesmo protocolo de transporte que o serviço da CMD. Assim, também é uma solução bastante segura para a transmissão de dados sensíveis, como credenciais e documentos.

Critério : Desenvolvimento

Critérios/Iterações		Integração de SDK do middleware versão 3 do Cartão de Cidadão	Integração de serviço Chave Móvel Digital	Integração de Sign'Stash
		Desenvolvimento	Dificuldade no desenvolvimento	Média
Atualidade das tecnologias	Baixa		Alta	Média

Figura 5.3: Comparação critério Desenvolvimento

Do ponto de vista da equipa de desenvolvimento, a comparação das soluções, seja no âmbito da dificuldade na integração de serviços, seja nas tecnologias usadas e as suas restrições, traz especialmente importância para as implementações referenciadas neste projeto.

Na primeira iteração, foi necessário o desenvolvimento de um servidor local, para comunicar diretamente com o leitor de cartões. A especificação do JNLP que executa o servidor, a configuração do mesmo, tratamento de pedidos do servidor remoto e respostas do servidor local e a integração da biblioteca da AMA, tornou o desenvolvimento desta solução de dificuldade média, na medida em que foi necessário o uso de novas tecnologias como o Jetty e novos protocolos como o Java Network Launch Protocol, que configura todos os recursos essenciais para a execução de um servidor local. Como referido anteriormente, JNLP é uma tecnologia *deprecated*, na medida em que a SIAG, quando atualizar o Java de 8 para 9, necessitará de usar uma nova tecnologia como OpenWebStart, que é uma reimplementação da tecnologia JWS e JNLP.

Quanto ao serviço da Chave Móvel Digital, a integração do protocolo SOAP revelou-se de grande dificuldade, por necessitar da especificação correta do WSDL para envio de mensagens, e de uma correta interpretação da documentação deste serviço. Para além disto, a criação do *hash* (ficheiro temporário pré-assinado) necessita da personalização de um documento, na medida em que é necessário criar uma assinatura, semelhantemente à Figura 7.8. Esta personalização¹ requereu a inserção de imagens, como o logótipo da AMA e da entidade ao qual o utilizador está associado, utilização de dados do certificado público do Cidadão e a geração do hash do documento ser feita conforme o “PKCS 1: RSA Cryptography Specifications Version 2.2”. As tecnologias utilizadas para o desenvolvimento desta solução são atuais, tendo em conta que as dependências utilizadas para o envio de mensagens SOAP datam de 2017. Esta solução encontra-se em ambiente de pré-produção, pronta para avaliação de um responsável da AMA. Apesar disso, a estilização da assinatura deve ser melhorada, na medida em que as imagens encontram-se com pouca resolução e o texto não é esteticamente apelativo. Também, falta a implementação das outras operações fornecidas pelo serviço da Chave Móvel Digital, como o SCMDMultipleSign (para assinar vários documentos em simultâneo) ou o ForceSMS (para reenviar o código OTP).

Por último, a solução Sign'Stash foi mais simples de integrar, por a arquitectura REST ser (genericamente) mais simples de desenvolver. Para o envio de informação, foi necessário a configuração e conversão de tipos de dados Java, como DocumentRequest que tem o conteúdo do

¹Esta funcionalidade não é detalhada nesta dissertação por não ser do âmbito da integração de serviços

ficheiro, tipo, nome e ID, em JSON. Quanto à atualidade das tecnologias, o *software* utilizado para a realização de pedidos HTTP data de 2015. Apesar de não ser uma tecnologia antiga, é possível de ser atualizada para uma versão mais recente.

Critério : Cliente

Critérios/Iterações		Integração de SDK do middleware versão 3 do Cartão de Cidadão	Integração de serviço Chave Móvel Digital	Integração de Sign'Stash
Cliente	Hardware adicional	✓	✓	✗
	Processo ininterrupto	✗	✗	✓
	Permite estilizar assinatura	✓	✓	✗

Figura 5.4: Comparação critério Cliente

Do ponto de vista do cliente, o *stakeholder* mais importante no desenvolvimento de das soluções analisadas, há alguns aspetos importantes a considerar.

A utilização de *hardware* adicional pode ser um impedimento para a utilização de uma solução, na medida em que força o mesmo a usar equipamento extra, para além do computador para aceder ao produto da SIAG. Das três soluções desenvolvidas, só o serviço Sign'Stash não requer *hardware* adicional: o *middleware* da AMA requer o Cartão de Cidadão para identificação do Cidadão; o serviço da Chave Móvel Digital requer o telemóvel para a validação através da inserção do código OTP enviado para o mesmo. Também é desejado que a experiência do cliente seja a melhor possível, na medida em que o processo de assinatura seja fácil e rápido de realizar. Assim, o único serviço de assinatura que cumpre estes aspetos é o serviço Sign'Stash, enquanto que, após se fazer o pedido de assinatura, os outros serviços requerem a inserção de um PIN (para o serviço do *middleware* da AMA) ou o código OTP (para o serviço da Chave Móvel Digital).

Quanto à personalização da assinatura, dados como *timestamp*, localização ou razão podem ser importantes para o leitor do documento assinado. Esta funcionalidade está disponível para o serviço de assinatura por Cartão de Cidadão e Chave Móvel Digital, em que o primeiro serviço só permite inserir "razão" e "localização" da assinatura, e o segundo serviço permite inserir qualquer tipo de parâmetros que seja importante para o utilizador e futuro leitor, tendo em conta que a estampa da assinatura é desenhada e implementada pela SIAG. O serviço Sign'Stash, por a implementação da assinatura ser invisível, não permite estilizar a assinatura².

Critério : Servidor

A disponibilidade de serviços é um fator importante a ter em consideração, tendo em conta que os servidores que fornecem estas funcionalidades, nem sempre estão acessíveis³. Dos três serviços desenvolvidos, a integração do SDK do *middleware* da AMA é o único que está sempre disponível,

²Numa futura implementação em que a assinatura seja visível, vai ser possível estilizar a mesma

³Servidores podem estar em manutenção

<i>Critérios/Iterações</i>		Integração de SDK do middleware versão 3 do Cartão de Cidadão	Integração de serviço Chave Móvel Digital	Integração de Sign'Stash
Servidor	Disponibilidade	100%	Dependente da AMA	Dependente da Multicert

Figura 5.5: Comparação critério Servidor

visto que é um servidor local, descarregado da SIAG e executado na própria máquina. Os outros serviços estão dependentes da entidade que os desenvolveu, como por exemplo a AMA que disponibiliza o serviço da Chave Móvel Digital e a Multicert, que disponibiliza o serviço Sign'Stash.

Capítulo 6

Considerações finais

De um modo geral, o presente projeto atingiu o seu objetivo principal, que consiste na integração de serviços de assinatura digital, seguindo princípios e regras de interoperabilidade próprios, assim como o cumprimento das restrições tecnológicas e requisitos. Para isto, a solução foi desdobrada em três integrações de serviços de assinatura digital.

O desenvolvimento de *software* é um processo complexo que resulta do levantamento de requisitos, análise dos mesmos e, posteriormente, o seu desenvolvimento. No contexto deste projeto, o levantamento de requisitos teve especial importância, por, as soluções a desenvolver, serem baseadas nas necessidades dos clientes. A assinatura de documentos, seja de relevância fiscal ou não, é um requisito cada vez mais desejado em produtos, devido à constante modernização e desburocratização dos processos dos Cidadãos. Desta forma, foram desenvolvidas três soluções que vão de encontro à necessidade de assinatura digital, com diferentes requisitos e propósitos, com a finalidade de contemplar todo o tipo de assinaturas digitais e restrições dos seus utilizadores.

O primeiro serviço resulta da integração de uma biblioteca desenvolvida pela AMA, que fornece uma panóplia de funcionalidades associadas ao Cartão de Cidadão. Este serviço tem alguns constrangimentos para o utilizador, na medida em que necessita de *hardware* adicional e da inserção do PIN de assinatura digital associado ao Cartão de Cidadão, assim como a instalação de um servidor local na sua máquina. Esta assinatura só tem validade legal em documentos sem relevância fiscal, de formato PDF e XML.

O segundo serviço integrado é a utilização da Chave Móvel Digital para a assinatura de documentos PDF. Apesar da CMD não assinar documentos em formato XML, também requer o uso de *hardware*, como o telemóvel, e do respetivo código de assinatura. No desenvolvimento deste serviço, integrou-se protocolos como SOAP que utiliza HTTP para a comunicação e TLS para a encriptação da comunicação, de forma a proteger a privacidade dos dados transferidos entre a entidade integradora e a AMA.

O serviço Sign'Stash pode ser considerado o mais completo das três soluções desenvolvidas, na medida em que assina qualquer tipo de documento, usa protocolos que estabelecem uma comunicação segura e não requer *hardware* adicional. Contudo, este serviço é direcionado para representantes de entidades, em que o certificado utilizado na assinatura é um selo qualificado gerado em nome da empresa a que o utilizador está associado. Também, neste momento, não apresenta uma assinatura visível.

Os três serviços de assinatura digital integrados, são exemplos de como um problema genérico como a

integração de assinatura digital num sistema ERP, pode ser desdobrado e resolvido, recorrendo a diferentes tecnologias e arquiteturas. A viabilidade de cada um dos serviços integrados foi avaliada com base em requisitos não-funcionais específicos de cada serviço.

Dos requisitos esperadas das soluções desenvolvidas, ser interoperável é o mais importante na integração de *webservices* como a integração da Chave Móvel Digital e o serviço Sign'Stash, que têm protocolos e regras muito específicos para uma correta integração com entidades externas. Sem uma correta definição de transmissão de dados, como a utilização do protocolo de transporte HTTP, e formatação de dados, através de JSON ou XML, é impossível as duas entidades comunicarem. Por esse motivo, na integração de serviços externos, cumprir as regras de interoperabilidade definidas pela entidade fornecedora, permite uma correta troca de informação de maneira uniforme e eficiente entre sistemas independentes.

Com a atualização dos serviços de assinatura digital do produto da SIAG, os clientes poderão desfrutar de um serviço de assinatura mais "rico" em funcionalidades, na medida em que tem uma maior panóplia de possibilidades quanto ao tipo de documento a assinar, estilização da assinatura e certificação oficial da assinatura, através do Governo Português, ou de uma entidade credenciada como a Multicert.

Assim, serve este projeto como impulsor da desburocratização e modernização de processos de assinatura, assim como, do uso e melhoria de funcionalidades que os serviços integrados prestam, como autenticação por Cartão de Cidadão ou Chave Móvel Digital, ou apresentação de assinatura em faturas eletrónicas.

Numa perspetiva mais pessoal, este projeto enriqueceu-me enquanto futuro profissional da área de desenvolvimento de *software*. Forneceu-me as *hard skills* para poder trabalhar num sistema tão complexo e capacitado como o da SIAG, através da consolidação do uso da linguagem Java e de ferramentas de apoio a projetos, e as *soft skills* para uma correta análise da documentação de serviços, capacidade de resolver problemas e comunicação eficaz com outros colaboradores.

Bibliografia

Ana P. M. V. Almeida. O papel da interoperabilidade na administração pública: Contributos para melhorar a gestão de informação e a satisfação dos cidadãos. Master's thesis, Instituto Superior de Ciências Sociais e Políticas da Universidade de Lisboa, June 2020.

AltexSoft. What is soap: Formats, protocols, message structure, and how soap is different from rest. <https://www.altexsoft.com/blog/engineering/what-is-soap-formats-protocols-message-structure-and-how-soap-is-different-from-rest/>, August 2019.

AMA. Manual técnico do middleware cartão de cidadão. http://www.pofc.qren.pt/ResourcesUser/2013/Concursos_Avisos/AAC01_SAMA_4_Middleware_CartaoCidadao_Manual May 2007. Last accessed 19 August 2022.

AMA. Assinatura digital autenticação.gov. <https://www.autenticacao.gov.pt/web/guest/assinatura-digital/assinatura-digital-qualificada>, 2020. Last accessed 19 July 2022.

AMA. O que é a chave móvel digital? <https://www.autenticacao.gov.pt/a-chave-movel-digital>, 2021. Last accessed 19 July 2022.

AMA. Scmdservice.wsdl. <https://github.com/amagovpt/doc-CMD-assinatura/blob/main/wsdl/SCMDService.wsdl>, May 2022a. Last accessed 26 August 2022.

AMA. Declaração de práticas de operação do scmd. march 2022b.

AMA. Manual do sdk – middleware do cartão de cidadão. https://amagovpt.github.io/docs.autenticacao.gov/manual_sdk.html, 2022c. Last accessed 08 August 2022.

Pedro Antunes, Nguyen Hoang Thuan, and David Johnstone. Nature and purpose of visual artifacts in design science research. *Information Systems and e-Business Management*, 20, 06 2022. doi: 10.1007/s10257-022-00559-2.

Len Bass, Paul Clements, and Rick Kazman. *Software Architecture In Practice*. January 2003. ISBN 978-0321154958.

Matti Bragge. Model-view-controller architectural pattern and its evolution in graphical user interface frameworks. Master's thesis, Lappeenranta University of Technology School of Technology Management, August 2013.

- Inês F. Carvalho. A modernização na administração pública: O caso do programa simplex. Master's thesis, Faculdade de Direito da Universidade de Coimbra, 2020.
- Imaginary Cloud. Json vs. xml: Which one is better? <https://www.imaginarycloud.com/blog/json-vs-xml/>, January 2021.
- devmedia. Rest tutorial. <https://www.devmedia.com.br/rest-tutorial/28912>, 2013. Last accessed 04 August 2022.
- DigitalOcean. An introduction to oauth 2. <https://www.digitalocean.com/community/tutorials/an-introduction-to-oauth-2>, July 2021.
- DigitalOcean. Struts tutorial for beginners. <https://www.digitalocean.com/community/tutorials/struts-tutorial-for-beginners>, August 2022. Last accessed 10 September 2022.
- Diário de Notícias. Chave móvel digital: "É quase como mostrar o cartão de cidadão à distância". <https://www.dn.pt/sociedade/chave-movel-digital-e-quase-como-mostrar-o-cartao-de-cidadao-a-distancia-15145765.html>, September 2022. Last accessed 10 September 2022.
- ePortugal. Ama. <https://eportugal.gov.pt/entidades/agencia-para-a-modernizacao-administrativa>, 2022. Last accessed 08 August 2022.
- Governo da República Portuguesa. Chave móvel digital tem mais de 2 milhões de utilizadores ativos. <https://www.portugal.gov.pt/pt/gc22/comunicacao/noticia?i=chave-movel-digital-tem-mais-de-2-milhoes-de-utilizadores-ativos>, July 2021. Last accessed 26 August 2022.
- Guru99. Soap web services tutorial: What is soap protocol? example. <https://www.guru99.com/soap-simple-object-access-protocol.html>, 2022. Last accessed 03 August 2022.
- Red Hat. What is middleware? <https://www.redhat.com/en/topics/middleware/what-is-middleware>, 2022. Last accessed 08 August 2022.
- Alan Hevner, Alan R, Salvatore March, Salvatore T, Park, Jinsoo Park, Ram, and Sudha. Design science in information systems research. *Management Information Systems Quarterly*, 28, March 2004.
- IBM. Model-view-controller architecture. <https://www.ibm.com/docs/en/radfw/9.6.1?topic=cycle-model-view-controller-architecture>, March 2016.
- JavaWorld. Base64 encoding and decoding in java 8. <https://www.infoworld.com/article/3240006/base64-encoding-and-decoding-in-java-8.html>, December 2017. Last accessed 26 August 2022.
- Andreas Jedlitschka and Dietmar Pfahl. Reporting guidelines for controlled experiments in software engineering. December 2005. ISBN 0-7803-9507-7. doi: 10.1109/ISESE.2005.1541818.
- json.org. Introducing json. <https://www.json.org/json-en.html>. Last accessed 04 August 2022.
- Philippe Kruchten. Architectural blueprints—the “4+1” view model of software architecture. November 1995.

- MDN. Http. <https://developer.mozilla.org/en-US/docs/Web/HTTP>. Last accessed 28 July 2022.
- Ministério da Ciência e da Tecnologia. Decreto-lei n.º 290-d/99, de 2 de agosto, artigo 7º. August 1999.
- Multicert. Quem somos. <https://www.multicert.com/pt/quem-somos/>, 2022. Last accessed 19 July 2022.
- Observador. Já não precisa de leitor de cartões para usar serviços online do cartão de cidadão. <https://observador.pt/2017/12/07/ja-nao-precisa-de-leitor-de-cartoes-para-usar-servicos-online-do-cartao-de-cidadao/>, December 2017. Last accessed 10 September 2022.
- OpenLogic. What is apache activemq? <https://www.openlogic.com/blog/what-apache-activemq>, June 2020. Last accessed 14 September 2022.
- Opensoft. Web service: o que é, como funciona, para que serve? <https://www.opensoft.pt/web-service/>, 2016. Last accessed 03 August 2022.
- Oracle. Deploying software with jnlp and java web start. <https://www.oracle.com/technical-resources/articles/javase/ds-jnlp-javawebstart.html>, August 2002. Last accessed 26 August 2022.
- Cláudia A. Pena. Estudo comparativo entre as aplicações de assinatura digital com o cartão de cidadão e a chave móvel digital. Master's thesis, Escola Superior de Tecnologia e Gestão do Instituto Politécnico de Leiria, 2020.
- Mariano Pimentel, Denise Filippo, and Thiago M. dos Santos. Design science research: pesquisa científica atrelada ao design de artefatos. *RE@D – Revista de Educação a Distância e eLearning*, 3, March/April 2020. ISSN 2182-4967.
- Red Hat. Rest. <https://www.redhat.com/en/topics/api/what-is-a-rest-api>. Last accessed 28 July 2022.
- Red Hat. What's an api? <https://www.redhat.com/en/topics/api/what-is-a-rest-api>, 2022. Last accessed 28 July 2022.
- RockContent. Entenda o que é xml e quais as utilidades dessa linguagem de marcação. <https://rockcontent.com/br/blog/o-que-e-xml/>, 2018. Last accessed 04 August 2022.
- Savvy Security. Especificação de requisitos. <https://cheapsslsecurity.com/blog/how-to-check-ssl-certificate-information-in-ie/>, July 2017.
- SIAG. Especificação de requisitos, July 2020.
- Sign'Stash. Connecting your service. <https://must.atlassian.net/wiki/spaces/SIGNSTASH/pages/786720>, November 2022.
- SIMPLEX. <https://www.simplex.gov.pt/>, 2022. Last accessed 15 July 2022.
- S.R. Subramanya and B.K. Yi. Digital signatures. *Potentials, IEEE*, 25, April 2006. doi: 10.1109/MP.2006.1649003.
- António Tavares. *Administração pública portuguesa*. Fundação Francisco Manuel dos Santos, 2019. ISBN 9789898943286.

Tecmundo. O que é ssl? <https://www.tecmundo.com.br/seguranca/1896-o-que-e-ssl-.html>, April 2009. Last accessed 4 October 2022.

Nguyen Hoang Thuan, Andreas Drechsler, and Pedro Antunes. Construction of design science research questions. *Communications of the Association for Information Systems*, January 2019. ISSN 1529-3181.

John R. Venable, Jan Pries-Heje, and Richard L. Baskerville. Choosing a design science research methodology. *ACIS 2017 Proceedings*, 2017.

W3Schools. Json vs xml. https://www.w3schools.com/js/js_json_xml.asp. Last accessed 4 October 2022.

José D. Zunino. Certificação digital: assinatura digital, certificados digitais e sua utilização no mercado nacional. Master's thesis, August 2017.

Anexos

7.1 Tecnologias

7.1.1 Model-View-Controller

A variante MVC da IBM Bragge (2013) é um padrão de desenho de *web development*. Consiste em três componentes: *Model*, a *View* e *Controller*.

Model - é diretamente responsável pela gestão de dados. Trata do controlo e transferência de dados entre o *backend*, sistema/serviço de persistência de dados, e o *frontend*, responsável pela apresentação de dados. Logo, este componente é independente dos outros, na medida em que não lhe interessa como é que os dados são apresentados ou quando é que são alterados. Para ocorrerem alterações nos dados, têm de ser associado a uma "View" um modelo de dados, para quando ocorrerem alterações na vista, o modelo de dados ser notificado;

View - tem como responsabilidade a apresentação de dados ao utilizador. Para isso, necessita de um modelo de dados a associar à vista;

Controller - é o componente que recebe *input* do utilizador e redirecciona o utilizador para a vista correta. É costume utilizar o controlador fornecido pela *framework*, chamado FacesServlet.

7.1.2 JavaServer Faces

JavaServer Faces (JSF) é uma *framework web MVC* para aplicações Java que simplifica a construção de *interfaces* para aplicações *web*, através da reutilização de componentes numa página. JSF por utilizar MVC, associa componentes UI a modelos de dados, geridos por controladores do lado do servidor.

Esta *framework* permite a programadores Java focarem-se no desenvolvimento de serviços e dados, para posteriormente os integrem com a *interface*. Enquanto isso, também permite aos programadores de páginas *web* criarem *interfaces*, reutilizando componentes, disponibilizados pela API do JSF, com a lógica necessária já implementada.

7.1.3 Struts

Struts é uma *framework* para desenvolvimento *web* que utiliza o padrão MVC. Permite criar aplicações *web* extensíveis e flexíveis baseadas em tecnologias padrão como páginas JSP (JavaServer Pages) e XML(Extensible Markup Language).

Enquanto que JavaServer Faces é orientado a eventos e componentes, Struts é uma *framework* que mapeia URLs a atividades e código no *backend*, orientando o desenvolvimento *web* a páginas.

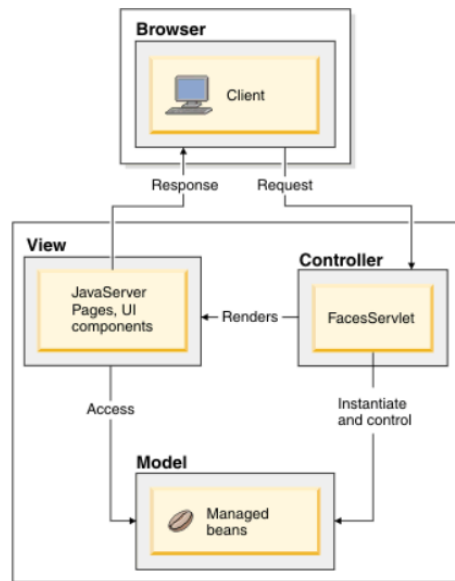


Figura 7.1: Arquitectura JavaServer Faces (IBM, 2016)

Listagem 14: Reencaminhamento de ação a modelo em Struts (DigitalOcean, 2022)

```

1 <?xml version="1.0" encoding="UTF-8"?>
2
3 <struts>
4 <package name="user" namespace="/User" extends="struts-default">
5   <action name="login" class="com.journaldev.struts2.action.LoginAction">
6     <result name="SUCCESS">/welcome.jsp</result>
7     <result name="ERROR">/error.jsp</result>
8   </action>
9 </package>
10 </struts>

```

A listagem 14 serve para ilustrar como é feito o reencaminhamento de uma ação a uma página JSP. Para a ação "login", ao qual está associada a classe "LoginAction". Todo o modelo de dados tem um método "execute()" ao qual está associada lógica de processamento da respetiva ação. Para o caso acima, caso esse método resulte em "SUCCESS", é reencaminhado para a página "welcome.jsp", se der "ERROR", é reencamnhado para a página "error.jsp". Como se pode observar, é possível ver como é que o padrão MVC está embebido nesta *framework*, na medida em que o controlador (Listagem 14) define quais as ações (modelo) estão associadas a páginas (vistas).

7.1.4 PrimeFaces

PrimeFaces é uma *framework* para aplicações que utilizem JavaServer Faces. É uma biblioteca de componentes gráficos para aplicações *web* baseadas em JSF. A utilização desta *framework* facilita o trabalho do programador e melhora a experiência do utilizador, na medida em que é mais fácil o desenvolvimento de uma aplicação, por ter uma grande opção de componentes gráficos já desenvolvidos.

7.1.5 JWS

Instalar aplicações Java no cliente era uma tarefa complicada devido às diferenças na utilização de *browsers*, versão do Java Runtime Environment e políticas de segurança. Esta dificuldade levava muitos programadores a abandonar soluções "rich client-side" (aplicações ricas em funcionalidades, independentes

de um servidor central) e a passar a utilizar tecnologia Java direcionada para servidor. Contudo, com o desenvolvimento da tecnologia Java Web Start (JWS) e o protocolo subjacente Java Network Launch Protocol (JNLP), a instalação de aplicações "cross-platform" no cliente.

Java Web Start é um mecanismo para a instalação de aplicações Java através de um *web server*. São programas, tipicamente iniciados pelo *browser*, instalados no cliente e executados fora do ambiente do *browser*. Quando distribuídos no sistema do cliente, já não precisam de ser descarregados de novo, tendo em conta que podem atualizar de forma automática. Estes programas apesar de correrem fora do *browser*, não tem liberdade total para realizarem as operações que quiserem dentro do sistema do cliente. Continuam limitadas a um *container*, com políticas de segurança definidas (Oracle, 2002).

O Java Web Start está incluído no JRE 5.0. Com o lançamento do Java Platform Standard Edition 9 (Plataforma que inclui JDK, JRE, entre outros), JWS foi descontinuado, por isso, utilizadores que corram versões mais recentes do Java Runtime Environment, não poderão correr aplicações deste tipo.

OpenWebStart é uma implementação *open-source* de Java Web Start, atualizada para as versões mais recentes de JRE.

7.1.6 JNLP

Java Network Launch Protocol permite uma aplicação ser lançada num cliente, através do uso de recursos localizados no *web server*. Consiste num conjunto de regras que descreve como é que o mecanismo de lançamento de uma aplicação deve ser implementado.

Uma aplicação, lançada por Java Web Start, é definida como uma coleção de bibliotecas que contém classes Java e suporta recursos como imagens, ficheiros XML, entre outros... Num cenário, em que uma aplicação necessite de várias bibliotecas externas, serve um ficheiro que as encapsule como uma só aplicação, como um ficheiro JNLP, que especifique como é que os recursos são associados.

Listagem 15: Exemplo JNLP

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <jnlp spec="1.0+" codebase="http://myserver.com/sampleapp"
3   href="webstart.jnlp">
4   <resources>
5     <j2se version="1.7+"
6     href="http://java.sun.com/products/autodl/j2se"/>
7     <jar href="startup.jar" main="true" />
8     <jar href="sampleapp.jar" />
9   </resources>
10  <application-desc
11    name="Sample Application"
12    main-class="mypkg.sampleapp.startup.Main" >
13  </application-desc>
14  <update check="background"/>
15 </jnlp>

```

Do exemplo acima, pode-se retirar algumas informações como:

- Este ficheiro foi executado a partir da localização "http://myserver.com/sampleapp";
- A aplicação chama-se "Sample Application" e é composta por 2 bibliotecas: a "startup.jar" e "sampleapp.jar";
- A classe executável chama-se "mypkg.sampleapp.startup.Main" e encontra-se na biblioteca "startup.jar";

- A aplicação requer uma versão mínima JRE de 1.7 para executar. Caso esta versão não esteja disponível na máquina do cliente, será descarregada a partir da localização "http://java.sun.com/products/autodl/j2se".

A utilização desta tecnologia é considerada *deprecated* para o Java Runtime Version 9, paralelamente ao JWS.

7.1.7 ActiveMQ e Java Message Service

Java Message Service (JMS) foi idealizado com o propósito de desenvolver aplicações empresariais, em Java, com maior facilidade, que assincronamente enviam e recebem dados. Define uma API comum de envio de mensagens e suporta dois modelos de mensagens: *point-to-point* ou *publish-subscribe*.

No primeiro modelo, os pedidos são enviados para uma fila FIFO (First-in, First-out) de mensagens. Os produtores enviam as mensagens enquanto que os consumidores as recolhem.

No segundo modelo, as mensagens são enviadas para um "tópico" ao qual só os consumidores desse tópico, tem acesso.

ActiveMQ é um intermediário de mensagens que implementa JMS. De acordo com a OpenLogic (OpenLogic, 2020), Apache ActiveMQ é a melhor solução para troca de mensagens a nível empresarial, por ser gratuito, estar em constante desenvolvimento e suportar vários tipos de protocolos, fornecendo canais de comunicação entre aplicações desenvolvidas em diferentes linguagens de programação.

7.1.8 Secure Socket Layer e Transport Secure Layer

Secure Socket Layer ou SSL é um tipo de segurança digital que encripta a comunicação entre duas entidades. Esta tecnologia permite que cliente e servidor possam trocar informações em total segurança, protegendo a integridade e a veracidade do conteúdo que circula na Internet. Tal segurança só é possível através da autenticação das partes envolvidas na troca de informações (Tecmundo, 2009).

Transport Secure Layer ou TLS é uma reimplementação da tecnologia SSL, feita por outra empresa. Por esse motivo, os termos SSL e TLS são usados de forma indistinta.

Para haver troca de informação, é necessário estabelecer *handshake* SSL/TLS: processo em que um cliente e um servidor estabelecem algoritmos de encriptação, assim como a partilha de chaves públicas para comunicar de forma segura e trocar e validar os certificados da outra entidade.

Este processo pode ser resumido da seguinte forma:

1. O **cliente** envia um *acknowledgement*, juntamente com informação criptográfica, versão do SSL e algoritmos de encriptação que o cliente suporta;
2. O **servidor** responde com o algoritmo de encriptação escolhido. Pode pedir o certificado do cliente;
3. O **cliente** verifica o certificado do servidor e um conjunto de dados que permite computar uma chave privada para a encriptação de mensagens subsequentes. Caso tenha sido pedido o certificado do cliente, este envia um conjunto de dados encriptados com a própria chave pública e o seu certificado;
4. O **servidor** verifica o certificado. Este passo só acontece se for necessário autenticação do cliente;

5. O **cliente** envia uma mensagem a informar que a conexão foi feita com sucesso;
6. O **servidor** envia uma mensagem a informar que a conexão foi feita com sucesso;

Durante o resto da sessão estabelecida, o cliente e servidor trocam mensagens encriptadas com a mesma chave privada (Computada a partir do passo 3) - denominado de **encriptação simétrica**.

7.1.9 OAuth2

OAuth2 é uma *framework* de autenticação que permite a aplicações externas ter acesso a contas de utilizadores num serviço HTTP. Para isto, fornece uma API de autenticação às aplicações que integrem esta *framework*.

Abstract Protocol Flow

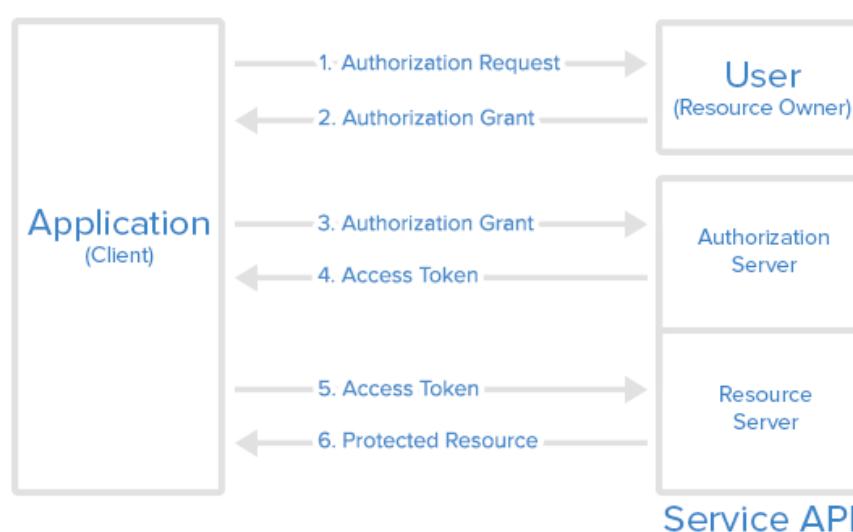


Figura 7.2: Fluxo de autenticação OAuth2 (DigitalOcean, 2021)

Acima encontra-se o fluxo de funcionamento de autenticação através da API da OAuth2. O primeiro e segundo passo são para a obtenção das credenciais de autenticação (como *username* e *password*). Os seguintes são o pedido do *token* à API para obter acesso a recursos privados, como *endpoints*, por exemplo.

7.1.10 Middleware e API

Middleware é um sistema de software que fornece funcionalidades para serem integradas noutros sistemas ou bases de dados. Contém protocolos de comunicação definidos, mapeamento e transformação de dados, autenticação e gestão de API. Ajuda os programadores a construir aplicações de forma mais eficaz, agindo como camada de ligação entre aplicações, dados e utilizadores.

Há vários casos de uso para utilização de um *middleware*, contudo, no contexto deste projeto, destaca-se o que utiliza APIs.

Muitas funcionalidades de um *middleware* podem ser acedidas através do uso de APIs, que são um conjunto de ferramentas, definições e protocolos que permitem a aplicações comunicarem uma com a outra. APIs tornam possível ligar produtos e serviços completamente diferentes, através de um meio de comunicação comum (Hat, 2022).

Muitos programadores usam *middleware* para integrar diferentes componentes de software noutras aplicações. O *middleware* oferece uma API para gerir a entrada e a saída de dados necessárias do sistema. Assim, as funcionalidades são abstraídas do utilizador.

No contexto desta tese, entende-se *middleware* como a “camada” de *software* entre o computador e o leitor de cartões e é através deste que são disponibilizadas ao sistema operativo e outras aplicações funcionalidades de autenticação e assinatura eletrónica, através da comunicação com o Cartão de Cidadão.

7.1.11 Software Development Kit

SDK é a sigla para *Software Development Kit*. É uma ferramenta fornecida por um fabricante de uma plataforma de hardware, sistema operativo, linguagem de programação ou de um *middleware*.

O SDK facilita a vida dos programadores, permitindo adicionar funcionalidades e construir componentes de *software* de uma aplicação com maior rapidez. SDKs podem variar desde uma simples API até *hardware* sofisticado que comunique com um determinado sistema.

Exemplo de um Software Development Kit é o *middleware* versão 3 do Cartão de Cidadão, desenvolvido pela AMA, que consiste num conjunto de bibliotecas utilizadas no acesso e suporte ao cartão de cidadão. Este SDK é comumente utilizado pelo programa Autenticacao.Gov, a que muitas Entidades Públicas e Privadas recorrem para a utilização das funcionalidades eletrónicas do Cartão de Cidadão, seja para autenticação, seja para a assinatura digital de documentos.

7.2 Legislação aplicável da Chave Móvel Digital

A legislação apresentada e referente à atividade da Chave Móvel Digital é a seguinte (AMA, 2022b):

- Regulamento (UE) n.º 910/2014 do Parlamento Europeu e do Conselho, de 23 de julho de 2014 relativo à identificação eletrónica e aos serviços de confiança para as transações eletrónicas no mercado interno;
- Despacho 155/2017 (Criação de assinaturas eletrónicas à distância, com gestão por um prestador qualificado de serviços de confiança em nome do signatário), de 5 de Dezembro de 2017, do Gabinete Nacional de Segurança;
- Decreto-Lei n.º 290-D/99, de 2 de Agosto, em todos os pontos que não forem contrariados pelo Regulamento (UE) n.º 910/2014;
- Lei n.º 37/2014, de 26 de Junho com as alterações introduzidas pela Lei n.º 32/2017, de 1 de Junho, e respetiva regulamentação;
- Lei n.º 67/98, de 26 de Outubro (Lei da proteção de dados pessoais);•Decreto-Lei n.º 36/2003 (Código da propriedade industrial);
- Lei n.º 41/2004 (Lei da proteção de dados pessoais no sector das comunicações eletrónicas);
- Regulamento(UE) n.º 2016/679 do Parlamento Europeu e do Conselho, de 27 de abril de 2016, relativo à proteção das pessoas singulares no que diz respeito ao tratamento de dados pessoais e à livre circulação desses dados (Regulamento Geral sobre a Proteção de Dados);

- Regulamento (UE) n° 611/2013 do Parlamento Europeu e do Conselho, de 24 de Junho de 2013, relativo às medidas aplicáveis à notificação e violação de dados pessoais;
- Lei das Comunicações Eletrónicas, aprovada pela Lei n° 5/2004 de 10 de Fevereiro;
- Decisão da Autoridade Nacional de Comunicações (ANACOM), aprovada por deliberação do respetivo Conselho de Administração, de 12 de Dezembro de 2013, relativa às exigências de comunicação e divulgação ao público de violações de segurança ou perdas de integridade ocorridas em redes e serviços de comunicações;
- Lei n° 109/2009, de 15 de Setembro (Lei do Cibercrime);
- Regulamento (CE) n° 593/2008 do Parlamento Europeu e do Conselho, de 17 de Junho de 2008, sobre a lei aplicável às obrigações contratuais (Roma I);
- Regulamento (CE) n° 864/2007 do Parlamento Europeu e do Conselho, de 11 de Julho de 2007, relativo à lei aplicável às obrigações extracontratuais (Roma II).

7.3 Figuras, Listagens e Tabelas

Iteração 1: Integração de SDK do *middleware* versão 3 do Cartão de Cidadão

Tabela 7.1: Cenário de Modificabilidade

Atributo qualitativo	Modificabilidade
Cenário	Um membro da equipa de desenvolvimento pretende adicionar outra funcionalidade do <i>middleware</i> desenvolvido pela AMA
Fonte	Equipa de desenvolvimento
Estímulo	Membro da equipa de desenvolvimento pretende adicionar outra funcionalidade
Artefacto	Módulo SiagSignature
Ambiente de execução	Desenvolvimento
Resposta	O programador consegue adicionar outra funcionalidade ao módulo SiagSignature, sem ter de fazer alterações ao software já desenvolvido
Medida de resposta	O custo pode ser medido num, ou mais, dos seguintes factores:
	Tempo
	introdução de defeitos no sistema
	Esforço
	Número de artefactos alterados

Tabela 7.2: Cenário de Portabilidade

Atributo qualitativo	Portabilidade
Cenário	Um utilizador pretende assinar um documento com o Cartão de Cidadão, num computador com sistema operativo MacOS
Fonte	Utilizador
Estímulo	Um utilizador pretende assinar um documento com o Cartão de Cidadão
Artefacto	Módulo SiagSignature
Ambiente de execução	A utilizar o produto da SIAG
Resposta	O utilizador consegue assinar um documento
Medida de resposta	Satisfação do utilizador; Documento assinado

Tabela 7.3: Cenário de Usabilidade

Atributo qualitativo	Usabilidade
Cenário	O utilizador pretende assinar um documento de forma eficiente e simples
Fonte	Utilizador
Estímulo	Utilizador pretende assinar documento
Artefacto	Módulo SiagPrimeFaces
Ambiente de execução	Aplicacional
Resposta	O utilizador consegue assinar um documento de forma intuitiva, simples e sem cometer erros
Medida de resposta	O custo pode ser medido num, ou mais, dos seguintes factores:
	Número de erros cometidos
	Satisfação do utilizador
	Aprendizagem do utilizador na realização do cenário

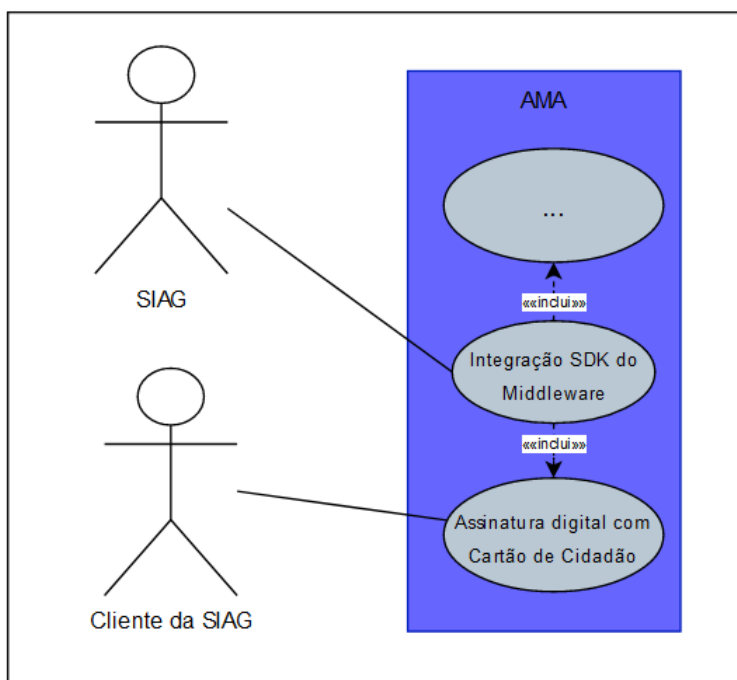


Figura 7.3: Diagrama de caso de uso - Assinar documento usando Cartão de Cidadão

Tabela 7.4: Cenário - Integrar SDK do *middleware*

Nome do caso de uso	Integrar SDK do <i>middleware</i>
Cenário	Integrador quer aceder às funcionalidades do SDK do <i>middleware</i> desenvolvido pela AMA
Evento	Após uma correta integração, é possível utilizar as funcionalidades da biblioteca
Descrição breve	A biblioteca tem um conjunto de funcionalidades disponíveis correspondente ao Cartão de Cidadão electrónico
Actores	SIAG
Casos de uso relacionados	Nenhum
Pré-Condições	Nenhuma
Pós-Condições	Funcionalidades disponíveis para serem usadas
Fluxo de eventos	Inserir biblioteca no servidor aplicacional; Carregar biblioteca estaticamente;
Condições excepcionais	Nenhuma

Tabela 7.5: Cenário - Assinar documento com Cartão de Cidadão

Nome do caso de uso	Assinar documento com Cartão de Cidadão
Cenário	Utilizador quer assinar um documento, utilizando o seu Cartão de Cidadão
Evento	Após configurar as especificações da assinatura, clica no botão "Avançar"
Descrição breve	Um utilizador tem um conjunto de documentos a serem autorizados. Caso o documento ainda não tenha sido assinado, ser-lhe á fornecida essa opção.
Actores	Utilizador
Casos de uso relacionados	Integrar SDK do <i>middleware</i>
Pré-Condições	O utilizador tem de saber o PIN de assinatura digital do seu Cartão de Cidadão, enviado por carta pela AMA
Pós-Condições	Documento assinado
Fluxo de eventos	Utilizador selecciona documento a assinar; Utilizador inicia Servidor Jetty; Especifica configurações de assinatura; Submete o pedido de assinatura; É pedido ao utilizador que insira o PIN; É devolvido o documento assinado.
Condições excepcionais	Caso o Servidor Jetty não tenha sido iniciado ou o utilizador tenha errado no PIN 3 vezes, é enviada uma mensagem de erro e o processo de assinatura termina

Tabela 7.6: Descrição de componentes

Nome	Tipo	Descrição
Cliente	Aparelho	Laptop/PC com JRE 1.7+
Browser	Internet Explorer, Google Chrome, Firefox	Qualquer versão
Servidor Jetty	Servidor Jetty	Versão 9
deploy_jetty	Ficheiro	Versão JNLP 1.0+
Servidor	Servidor Tomcat	Versão 8

Tabela 7.7: Descrição de pacotes contidos no SiagSignature

siag	
JettyServer	
JettyServer.java	Inicia o Servidor Jetty
servlets	
Digital Signature Servlet.java	Responde a pedidos feitos ao servidor
util	
Auth.java	Classe de utilidade responsável pela lógica de assinatura de documentos e que comunica diretamente com o SDK desenvolvido pela AMA

Tabela 7.8: Descrição de pacotes contidos no SiagPrimefaces

java	
fm/ppg/ad	
DocumentosDespesaController.java	Controlador da interface e gestor da lógica do processamento de pedidos de assinatura
webapp	
resources/js/fm/ppg/ad	
assinatura.js	Reencaminha pedidos entre <i>endpoints</i>
fm/ppg/ad	
documentosDespesa.xhtml	Realiza pedido de assinatura de um documento
documentosDespesa.json	Ficheiro de configurações de formulário de input para assinatura de documento
deploy_jetty.jnlp	Recurso para lançar servidor Jetty



Figura 7.4: Interface para parametrizar assinatura - Cartão de Cidadão

Iteração 2: Integração de serviço Chave Móvel Digital

Tabela 7.9: Cenário de Legalidade

Atributo qualitativo	Legalidade
Cenário	Um representante da AMA tenciona avaliar a solução desenvolvida pela SIAG para garantir que os requisitos legais impostos são cumpridos
Fonte	Representante da AMA
Estímulo	SIAG quer avançar para ambiente de produção
Artefacto	Módulo SiagPrimeFaces e Siag
Ambiente de execução	Desenvolvimento
Resposta	Por a solução cumprir os requisitos legais impostos pela AMA, esta é aprovada e procede para ambiente de produção
Medida de resposta	Nenhuma

Tabela 7.10: Cenário de Interoperabilidade

Atributo qualitativo	Interoperabilidade
Cenário	O sistema quer enviar/receber informação para/de uma entidade externa
Fonte	Utilizador
Estímulo	Utilizador pretende assinar documento
Artefacto	Módulo Siag
Ambiente de execução	Aplicacional
Resposta	O envio de informação entre as duas entidades é feita de forma facilitada e eficiente
Medida de resposta	Pode ser medida num, ou mais, dos seguintes factores:
	Demora na resposta;
	Facilidade na integração do serviço externo;

Tabela 7.11: Cenário de Usabilidade

Atributo qualitativo	Usabilidade
Cenário	O utilizador pretende assinar um documento de forma eficiente e simples
Fonte	Utilizador
Estímulo	Utilizador pretende assinar documento
Artefacto	Módulo SiagPrimeFaces
Ambiente de execução	Aplicacional
Resposta	O utilizador consegue assinar um documento de forma intuitiva, simples e sem cometer erros
Medida de resposta	O custo pode ser medido num, ou mais, dos seguintes factores:
	Número de erros cometidos
	Satisfação do utilizador
	Aprendizagem do utilizador na realização do cenário

Tabela 7.12: Realização caso de uso - Assinar documento com Chave Móvel Digital

Nome do caso de uso	Assinar documento com Chave Móvel Digital
Cenário	Utilizador quer assinar um documento, utilizando a Chave Móvel Digital
Evento	Após configurar as especificações da assinatura, clica no botão "Avançar"
Descrição breve	Um utilizador tem um conjunto de documentos a serem autorizados. Caso o documento ainda não tenha sido assinado, ser-lhe é fornecida essa opção.
Actores	Utilizador
Casos de uso relacionados	Nenhum
Pré-Condições	O utilizador tem de saber o PIN de assinatura digital da sua Chave Móvel Digital; Ter o telemóvel, para receber o OTP;
Pós-Condições	Documento assinado
Fluxo de eventos	Utilizador selecciona documento a assinar; Especifica configurações de assinatura; Submete o pedido de assinatura; Recebe o OTP no seu telemóvel; Insere o OTP na interface; É devolvido o documento assinado.
Condições excepcionais	Caso o PIN esteja incorreto não recebe o OTP no telemóvel e o processo de assinatura termina; Caso o OTP inserido na interface esteja incorreto, termina o processo de assinatura,

Tabela 7.13: Descrição de pacotes contidos no cmd

cmd	
assemblePDF	
BlankSignatureContainer.java	Classe responsável pela inserção de assinatura em branco
InjectAmaSignatureContainer.java	Classe responsável pela inserção da assinatura do Cidadão
PDFSigningManager.java	Classe responsável pela estilização da assinatura
broker	
SCMDBrokerService.java	Broker intermediário entre siagPrimeFaces e AMA
dto	
HashStructure.java	Estrutura que contém a assinatura, nome e identificador do documento
SignRequest.java	Estrutura que contém os parâmetros para realizar um pedido de assinatura
SignResponse.java	Estrutura de resposta a um pedido, que contém o certificado do Cidadão, SignStatus e HashStructure
SignStatus.java	Estrutura de resposta que contém informações sobre o pedido efetuado
service	
BasicHttpBinding_SCMDserviceImpl.java	Classe gerada automaticamente responsável pela configuração da ligação HTTPS
BasicHttpBinding_SCMDserviceSkeleton.java	Classe gerada automaticamente responsável pela descrição de cada operação
BasicHttpBinding_SCMDserviceStub.java	Classe gerada automaticamente responsável por configurar o pedido SOAP a cada operação
SCMDService_PortType.java	Interface das operações, gerada automaticamente
SCMDService_Service.java	Interface das configurações da ligação HTTP, gerada automaticamente
deploy.wsdd	Ficheiro para dar "deploy" do serviço SCMD
undeploy.wsdd	Ficheiro para dar "undeploy" do serviço SCMD
utils	
SCMDUtilService.java	Classe genérica para encriptação de informação e conversão de certificados

Listagem 16: Especificação Operação 1 - GetCertificate (AMA, 2022a)

```

1 <wsdl:operation name="GetCertificate">
2   <wsdl:input wsaw:Action="http://Ama.Authentication.Service/SCMDService/GetCertificate" message="
   tns:SCMDService_GetCertificate_InputMessage"/>
3   <wsdl:output wsaw:Action="http://Ama.Authentication.Service/SCMDService/GetCertificateResponse" message="
   tns:SCMDService_GetCertificate_OutputMessage"/>
4 </wsdl:operation>
5
6 xs:element name="GetCertificate">
7   <xs:complexType>
8     <xs:sequence>
9       <xs:element minOccurs="0" name="applicationId" nillable="true" type="xs:base64Binary"/>
10      <xs:element minOccurs="0" name="userId" nillable="true" type="xs:string"/>
11    </xs:sequence>
12  </xs:complexType>
13 </xs:element>
14
15 <xs:element name="GetCertificateResponse">
16   <xs:complexType>
17     <xs:sequence>
18       <xs:element minOccurs="0" name="GetCertificateResult" nillable="true" type="xs:string"/>
19     </xs:sequence>
20   </xs:complexType>
21 </xs:element>

```

Operação	GetCertificate	
	Parâmetros	Tipo
Parâmetro de Entrada	applicationId	byte[]
	userId (cifrado)	base64String
Parâmetro de Saída	certificate	string

Tabela 7.14: Operação 1 - GetCertificate

Listagem 17: Especificação Operação 1 - SCMDSign (AMA, 2022a)

```

1 <wsdl:operation name="SCMDSign">
2   <wsdl:input wsaw:Action="http://Ama.Authentication.Service/SCMDSign" message="
   tns:SCMDSign_InputMessage"/>
3   <wsdl:output wsaw:Action="http://Ama.Authentication.Service/SCMDSignResponse" message="
   tns:SCMDSign_OutputMessage"/>
4 </wsdl:operation>
5
6 <xs:element name="SCMDSign">
7   <xs:complexType>
8     <xs:sequence>
9       <xs:element xmlns:q1="http://schemas.datacontract.org/2004/07/Ama.Structures.CCMovelSignature"
10        minOccurs="0" name="request" nillable="true" type="q1:SignRequest"/>
11     </xs:sequence>
12   </xs:complexType>
13 </xs:element>
14 <xs:complexType name="SignRequest">
15   <xs:sequence>
16     <xs:element name="ApplicationId" nillable="true" type="xs:base64Binary"/>
17     <xs:element minOccurs="0" name="DocName" nillable="true" type="xs:string"/>
18     <xs:element name="Hash" nillable="true" type="xs:base64Binary"/>
19     <xs:element name="Pin" nillable="true" type="xs:string"/>
20     <xs:element name="UserId" nillable="true" type="xs:string"/>
21   </xs:sequence>
22 </xs:complexType>
23
24 <xs:element name="SCMDSignResponse">
25   <xs:complexType>
26     <xs:sequence>
27       <xs:element xmlns:q2="http://schemas.datacontract.org/2004/07/Ama.Structures.CCMovelSignature"
28        minOccurs="0" name="SCMDSignResult" nillable="true" type="q2:SignStatus"/>
29     </xs:sequence>
30   </xs:complexType>
31 </xs:element>
32 <xs:complexType name="SignStatus">
33   <xs:sequence>
34     <xs:element name="Code" nillable="true" type="xs:string"/>
35     <xs:element name="Message" nillable="true" type="xs:string"/>
36     <xs:element name="ProcessId" nillable="true" type="xs:string"/>
37   </xs:sequence>
38 </xs:complexType>

```

Operação	SCMDSign	Parâmetros	Tipo	Obrigatório?	Descrição
Parâmetro de Entrada	SignRequest	ApplicationId	byte[]	Sim	Identificador da aplicação que efetua o request
		DocName	string	Não	Nome do Documento ou identificador para permitir ao Cidadão identificar o ato que vai originar a assinatura
		Hash	byte[]	Sim	Hash da informação sobre a qual vai ser gerada a assinatura
		Pin	base64String(cifrado)	Sim	Código PIN do utilizador
		UserId	base64String(cifrado)	Sim	Identificador da conta do utilizador (ex.: Nº de Telemóvel: "+351 966666666")
Parâmetro de Saída	SignStatus	ProcessId	string		
		Code	string		
		Message	string		
		...			

Tabela 7.15: Operação 2 - SCMDSign

Listagem 18: Especificação Operação 1 - ValidateOtp (AMA, 2022a)

```

1 <wsdl:operation name="ValidateOtp">
2   <wsdl:input wsaw:Action="http://Ama.Authentication.Service/SCMDSservice/ValidateOtp" message="
   tns:SCMDSservice_ValidateOtp_InputMessage"/>
3   <wsdl:output wsaw:Action="http://Ama.Authentication.Service/SCMDSservice/ValidateOtpResponse" message="
   tns:SCMDSservice_ValidateOtp_OutputMessage"/>
4 </wsdl:operation>
5
6 <xs:element name="ValidateOtp">
7   <xs:complexType>
8     <xs:sequence>
9       <xs:element minOccurs="0" name="code" nillable="true" type="xs:base64Binary"/>
10      <xs:element minOccurs="0" name="processId" nillable="true" type="xs:string"/>
11      <xs:element minOccurs="0" name="applicationId" nillable="true" type="xs:string"/>
12    </xs:sequence>
13  </xs:complexType>
14 </xs:element>
15
16 <xs:element name="ValidateOtpResponse">
17   <xs:complexType>
18     <xs:sequence>
19       <xs:element xmlns:q4="http://schemas.datacontract.org/2004/07/Ama.Structures.CCMovelSignature"
20         minOccurs="0" name="ValidateOtpResult" nillable="true" type="q4:SignResponse"/>
21       <xs:element name="applicationId" nillable="true" type="xs:base64Binary"/>
22     </xs:sequence>
23   </xs:complexType>
24 </xs:element>
25
26 <xs:element name="SignResponse">
27   <xs:complexType>
28     <xs:sequence>
29       <xs:element xmlns:q2="http://schemas.datacontract.org/2004/07/Ama.Structures.CCMovelSignature"
30         minOccurs="0" name="SCMDSignResult" nillable="true" type="q2:SignStatus"/>
31       <xs:element name="certificate" nillable="true" type="xs:string"/>
32       <xs:element name="Signature" nillable="true" type="xs:base64Binary"/>
33     </xs:sequence>
34   </xs:complexType>
35 </xs:element>
36
37 <xs:complexType name="SignStatus">
38   <xs:sequence>
39     <xs:element name="Code" nillable="true" type="xs:string"/>
40     <xs:element name="Message" nillable="true" type="xs:string"/>
41     <xs:element name="ProcessId" nillable="true" type="xs:string"/>
42   </xs:sequence>
43 </xs:complexType>

```

Operação	ValidateOtp		
	Parâmetros	Tipo	
Parâmetro de Entrada	code	base64String	
	processId	string	
	applicationId	string	
Parâmetro de Saída	SignResponse	processId	byte[]
		Signature	Byte[]
		certificate	String
	SignStatus	ProcessId	string
		Code	string
		Message	string

Tabela 7.16: Operação 3 - ValidateOtp

Assinar com Chave Móvel Digital 

Clique para ativar ou gerir a assinatura da sua Chave Móvel Digital

Inserir número de telemóvel associado

+351 - Portugal

Inserir PIN da assinatura da Chave Móvel Digital

CANCELAR **CONFIRMAR**

Figura 7.5: Introdução de PIN para assinatura com Chave Móvel Digital

Assinatura

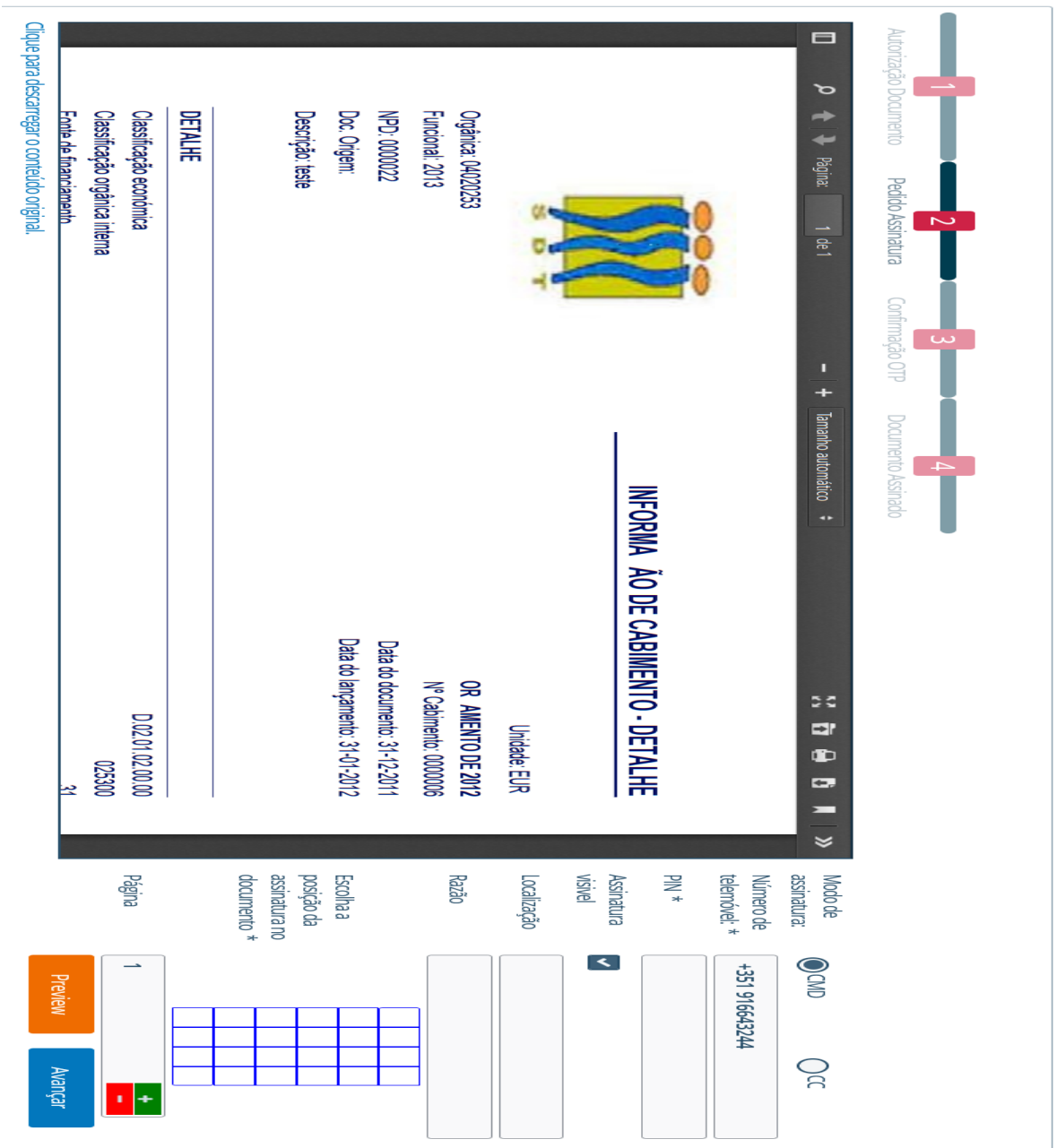


Figura 7.6: Interface para parametrizar assinatura - Chave Móvel Digital

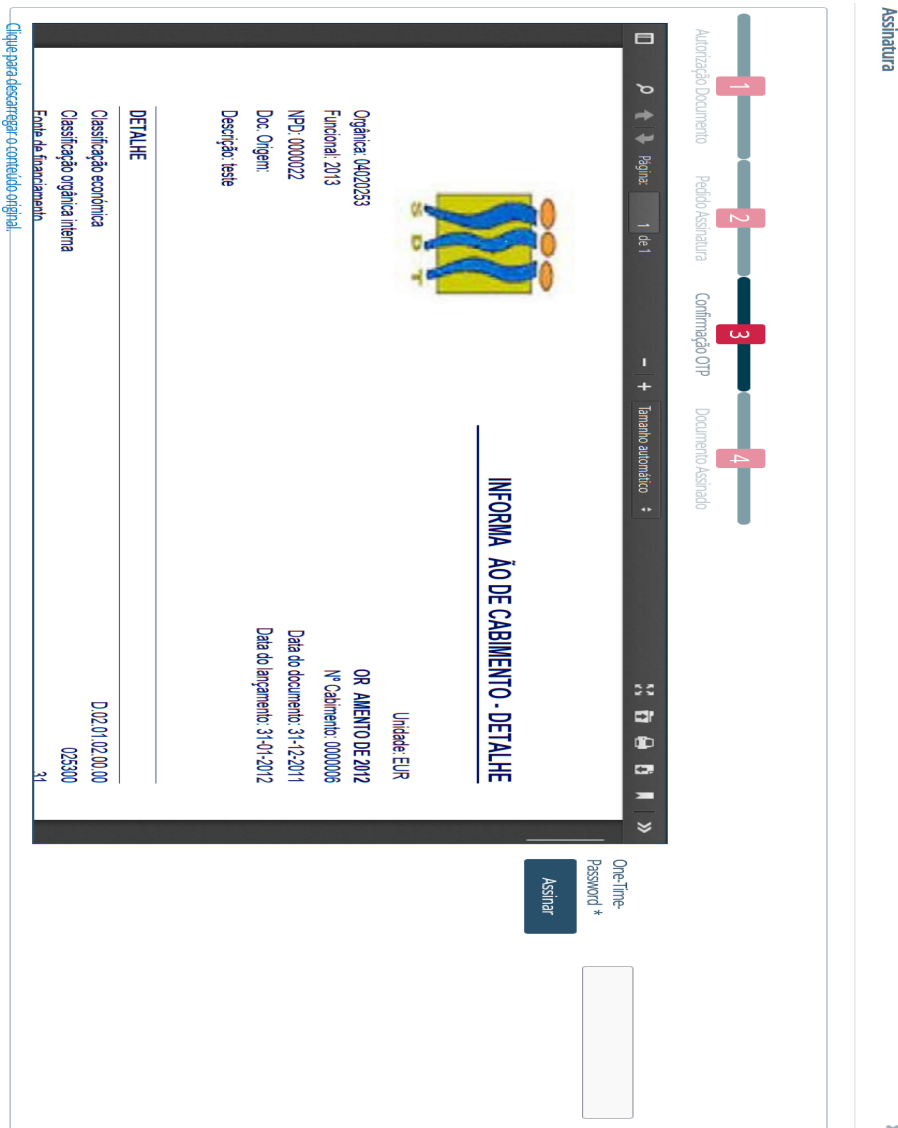


Figura 7.7: Interface de inserção do OTP - Chave Móvel Digital

Digitally signed by
Ricardo Filipe Nunes
Madeira
Date: 2022.09.21
00:04:59 BST
Reason: razao
Location: localizacao

Figura 7.8: Assinatura protótipo por Chave Móvel Digital

Iteração 3: Integração de Sign'Stash

Tabela 7.17: Cenário de Interoperabilidade

Atributo qualitativo	Interoperabilidade
Cenário	O sistema quer enviar/receber informação para/de uma entidade externa
Fonte	Utilizador
Estímulo	Utilizador pretende assinar documento
Artefacto	Módulo Siag
Ambiente de execução	Aplicacional
Resposta	O envio de informação entre as duas entidades é feita de forma facilitada e eficiente
Medida de resposta	Pode ser medida num, ou mais, dos seguintes factores:
	Demora na resposta;
	Facilidade na integração do serviço externo;

Tabela 7.18: Realização caso de uso - Assinar fatura eletrónica

Nome do caso de uso	Assinar fatura eletrónica
Cenário	Utilizador quer assinar uma fatura, de acordo com a regulação fiscal
Evento	O utilizador selecciona que quer assinar uma fatura
Descrição breve	O utilizador pretende assinar uma fatura utilizando o selo qualificado da SIAG
Actores	Utilizador
Casos de uso relacionados	Nenhum
Pré-Condições	
Pós-Condições	Fatura assinada
Fluxo de eventos	Utilizador selecciona uma fatura para assinar; Indica que quer assinar, clicando na respetiva checkbox; Surge um popup com o documento impresso e disponível para ser descarregado
Condições excepcionais	

Tabela 7.19: Descrição de pacote broker contido em multisert

multisert	
broker	
MULTISERTBrokerService.java	Broker intermediário entre Siag e Sign'Stash

Tabela 7.20: Descrição de pacote dto contido em multisert

multisert	
dto	
DocumentBodySpecification.java	Estrutura que contém as classes abaixo
DocumentRequest.java	Informação base sobre o documento a assinar, como tipo de documento e nome
Visible SignatureInfo.java	Configurações da assinatura visível
SignatureInfo.java	Informação sobre a assinatura, como tipo de assinatura, localização e razão

Listagem 19: Exemplo pedido token (Sign'Stash, 2022)

```
1 curl -v -X POST https://staging.must.digital/oauth2/authorization -server/oauth/token -d "grant_type =  
client_credentials" -H "Authorization: Basic <base64 of client_id:client_secret >"
```

Listagem 20: Exemplo pedido assinatura (Sign'Stash, 2022)

```
1 {  
2   "description": "Some context.",  
3   "documents": [  
4     {  
5       "documentRequest": {  
6         "base64Content": "document_in_base64",  
7         "contentType": "application/pdf",  
8         "externalReference": "Ext_Ref",  
9         "name": "document_example.pdf"  
10      },  
11      "signatureInfo": {  
12        "advancedSignatureInfo": {  
13          "canonicalTransform": "EXCLUSIVE_WITH_COMMENTS",  
14          "signatureLocation": "SignatureInformation",  
15          "signatureTransform": "ENVELOPED"  
16        },  
17        "applyTimestamp": true,  
18        "location": "Test API location",  
19        "reason": "Test API reason",  
20        "signatureType": "PAdES",  
21        "visibleSignatureInfo": {  
22          "area": "width=200,height=400",  
23          "font": "color=#000000,size=11",  
24          "locationType": "COORDINATE",  
25          "locationValue": "page=1,x=100,y=200",  
26          "oneOffTemplate": "Digitally Signed by $ certificate .getCommonName().",  
27          "templateType": "ONE_OFF"  
28        }  
29      }  
30    },  
31  ],  
32  "externalReference": "Ext_Ref",  
33  "returnSignedContent": true  
34 }
```