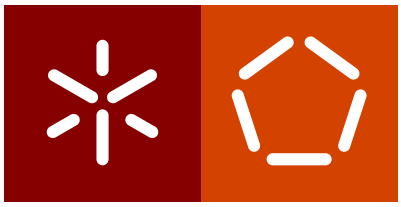
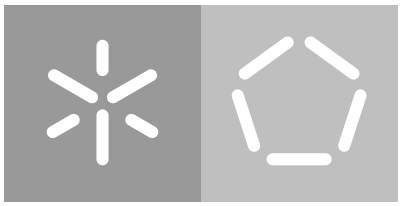


Universidade do Minho
Escola de Engenharia
Departamento de Informática

Octávio José Azevedo Maia

CLAV
Autenticação e integração na plataforma iAP

Dezembro 2019



Universidade do Minho
Escola de Engenharia
Departamento de Informática

Octávio José Azevedo Maia

CLAV
Autenticação e integração na plataforma iAP

Dissertação de Mestrado
Mestrado Integrado em Engenharia Informática

Trabalho realizado sob a orientação do Professor
José Carlos Ramalho

Dezembro 2019

DIREITOS DE AUTOR E CONDIÇÕES DE UTILIZAÇÃO DO TRABALHO POR TERCEIROS

Este é um trabalho académico que pode ser utilizado por terceiros desde que respeitadas as regras e boas práticas internacionalmente aceites, no que concerne aos direitos de autor e direitos conexos.

Assim, o presente trabalho pode ser utilizado nos termos previstos na licença abaixo indicada.

Caso o utilizador necessite de permissão para poder fazer um uso do trabalho em condições não previstas no licenciamento indicado, deverá contactar o autor, através do RepositóriUM da Universidade do Minho.

Atribuição
CC BY

https://creativecommons.org/licenses/by/4.0/

DECLARAÇÃO DE INTEGRIDADE

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

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

RESUMO

CLAV: Autenticação e integração na plataforma iAP

Existe uma preocupação cada vez maior em relação à quantidade de informação gerada e recebida por diversas instituições. Não só no que toca ao consumo excessivo de papel, como também a gestão de grandes quantidades de informação.

Com o intuito de simplificar a gestão documental, o governo tem desenvolvido diversas estratégias. Nomeadamente, na *Administração Pública (AP)*, com base em normas e orientações provenientes da Comissão Europeia.

O *Projeto M51-CLAV-Arquivo digital: Plataforma modular de classificação e avaliação da informação pública (CLAV)*, surge como uma dessas estratégias. Este visa a classificação e a avaliação de toda a documentação que circula na Administração Pública portuguesa, utilizando um referencial comum que permite o desenvolvimento de instrumentos de natureza transversal a aplicar em contexto organizacional.

Esta dissertação tem como objetivo primário, a integração dos serviços da plataforma *Autenticação.Gov* no CLAV, bem como a criação de estratégias apropriadas de gestão de utilizadores, autenticação de pedidos referentes à *API* pública disponibilizada, autenticação no backend e segurança da aplicação.

Keywords— Autenticação.Gov, Autenticação, CLAV, Segurança

ABSTRACT

CLAV: Authentication and integration on the iAP platform

There's a ever growing worry about the quantity of information that's generated and received by multiple institutions. Not only related to the excessive ammount of paper consumed, but as well as the management of such big quantities of information.

With the purpose of simplifying the process of managing such documents, the portuguese government has been developing various strategies. Namely, in the Public Administration, with directives provided by the European Comission.

The CLAV project comes up as one of those strategies. This allows to classify and evaluate all the documentation that circulates in the Portuguese Public Administration, using a common referential, that allows for the development of instruments of transversal nature that are applied in an organizational context.

This dissertation has as its primary objective, the integration of the services provided by the *Autenticação.Gov* platform in the CLAV project, as well as the creation of appropriate strategies of user management, authentication of requests refering to the public API, as well as backend authentication and overall security.

Keywords— Autenticação.Gov, Authentication, CLAV, Security

CONTEÚDO

1 INTRODUÇÃO 1
1.1 Enquadramento 1
1.2 Objetivos 2
1.3 Metodologia 3
1.4 Síntese 3
2 ESTADO DA ARTE 4
2.1 Encriptação e criptografia 5
2.1.1 Vulnerabilidades 5
2.1.2 Melhorias no processo de hashing 7
2.2 Autenticação.Gov 8
2.2.1 Principais funcionalidades 10
2.2.2 Workflow do Autenticação.Gov 12
2.2.3 Funcionamento do Autenticação.Gov 15
2.2.4 Especificação do Autenticação.Gov 25
2.2.5 Incongruências no Autenticação.Gov 36
2.3 Cartão de Cidadão 38
2.3.1 Especificação do Cartão de Cidadão 39
2.4 API de dados 41
2.4.1 Funcionamento do JSON Web Token 41
2.5 Síntese 43
3 SOLUÇÃO 44
3.1 Autenticação local 44
3.1.1 Especificação 44
3.1.2 Workflow da autenticação 45
3.2 Autenticação através de Cartão de Cidadão 46
3.2.1 Especificação 46
3.2.2 Workflow da autenticação 47
3.3 Proteção da API de dados 48
3.3.1 Especificação 48
3.3.2 Workflow da autenticação 49
3.4 Gestão de utilizadores 50
3.4.1 Especificação 50

3.4.2 Workflow da edição de um utilizador 51
3.5 Métrica da plataforma 52
3.5.1 Especificação 52
3.5.2 Workflow da incrementação da métrica na plataforma 52
3.6 Síntese 53
4 IMPLEMENTAÇÃO 54
4.1 Encriptação de informação sensível 56
4.1.1 Funcionamento do bcrypt 57
4.1.2 bcrypt no mundo real 58
4.2 Autenticação de utilizadores 61
4.2.1 Autenticação através de username e password 62
4.2.2 Autenticação através de Cartão de Cidadão 72
4.3 Gestão de utilizadores 76
4.3.1 Armazenamento e validação de sessões 78
4.4 Autenticação de pedidos à API de dados 82
4.4.1 Registo 82
4.4.2 Renovação 83
4.5 Gestão de chaves API 88
4.6 Autenticação backend 90
4.7 Métrica da plataforma CLAV 91
4.8 Síntese 92
5 CONCLUSÕES E TRABALHO FUTURO 93

LISTA DE FIGURAS

Figura 1 Representação de uma função de *hash*.(13) 5
Figura 2 Evolução do número de autenticações realizadas e utilizadores únicos baseados na plataforma *Autenticação.Gov*.(15) 9
Figura 3 Representação do número total de autenticações realizados com a plataforma *Autenticação.Gov*.(15) 9
Figura 4 Evolução do número de Autenticações realizadas através do Cartão de Cidadão.(15) 9
Figura 5 Evolução do número de Autenticações realizadas através da Chave Móvel Digital.(15) 10
Figura 6 Workflow de autenticação via Cartão de Cidadão(6). 12
Figura 7 Workflow de re-autenticação via Cartão de Cidadão(6). 14
Figura 8 Funcionamento do Autenticação.Gov na plataforma CLAV. 15
Figura 9 Comunicação entre o Autenticação e o Fornecedor de Serviços (plataforma CLAV) 25
Figura 10 Interação entre o utilizador e os respetivos *Providers*. (4) 27
Figura 11 Figura representativa da diferença entre comunicação via *Hyper Text Transfer Protocol (HTTP)* e *Hyper Text Transfer Protocol Secure (HTTPS)*. 29
Figura 12 Resumo do processo de *handshake* entre cliente e servidor. 30
Figura 13 Exemplo da assinatura e leitura através de par de chaves. 32
Figura 14 Formato de um certificado X.509 v3. 34
Figura 15 Exemplo de um Cartão de Cidadão(1). 38
Figura 16 Workflow da autenticação local. 45
Figura 17 Workflow da autenticação através de Cartão de Cidadão. 47
Figura 18 Workflow da autenticação de pedidos à API de dados. 49
Figura 19 Workflow da edição de um utilizador na plataforma CLAV. 51
Figura 20 Workflow da incrementação de métrica na plataforma CLAV. 52
Figura 21 Diagrama de sequência ilustrativo da comunicação entre frontend e backend na plataforma CLAV. 55
Figura 22 Página principal de autenticação na plataforma CLAV. 61
Figura 23 Página de registo na plataforma CLAV. 62

Figura 24 Página quando ocorre um registo com sucesso na plataforma CLAV. 63
Figura 25 Página quando ocorre um erro de registo na plataforma CLAV. 64
Figura 26 Página quando ocorre um erro de registo na plataforma CLAV. 64
Figura 27 Diagrama de sequência correspondente ao processo de registo na plataforma CLAV. 65
Figura 28 Página de login através de username e password na plataforma CLAV. 65
Figura 29 Página quando ocorre um login com sucesso na plataforma CLAV. 66
Figura 30 Página quando ocorre um erro de login na plataforma CLAV. 66
Figura 31 Página quando ocorre um erro de login na plataforma CLAV. 67
Figura 32 Diagrama de sequência correspondente ao processo de login local na plataforma CLAV. 67
Figura 33 Página de recuperação de credenciais na plataforma CLAV. 68
Figura 34 Mensagem de aviso do envio de email de recuperação. 68
Figura 35 Email de recuperação enviado pela plataforma CLAV. 69
Figura 36 Mensagem de sucesso após alteração de password na plataforma CLAV. 69
Figura 37 Mensagem de erro após inserção de email não existente na plataforma CLAV. 70
Figura 38 Mensagem de erro após tentativa de recuperação de uma conta registada com Cartão de Cidadão. 70
Figura 39 Diagrama de sequência relativo ao processo de recuperação na plataforma CLAV. 71
Figura 40 Página principal do Autenticação.Gov. 72
Figura 41 Pedido de inserção do PIN de autenticação por parte do Autenticação.Gov. 72
Figura 42 Listagem dos dados requisitados pela plataforma CLAV. 73
Figura 43 Página de registo de um novo utilizador através do Cartão de Cidadão. 74
Figura 44 Página de erro resultante de um PIN incorreto ou negação da leitura dos dados pretendidos. 74
Figura 45 Diagrama de sequência relativo ao processo de registo e login através de Cartão de Cidadão na plataforma CLAV. 75
Figura 46 Página de listagem de todos os utilizadores da plataforma CLAV. 76
Figura 47 Página de edição de um utilizador da plataforma CLAV. 76

Figura 48 Diagrama de sequência relativo à gestão e edição de utilizadores na plataforma CLAV. 77
Figura 49 Página de registo de uma chave API na plataforma CLAV. 82
Figura 50 Email de registo contendo uma chave API da plataforma CLAV. 83
Figura 51 Página de renovação de uma chave API na plataforma CLAV. 84
Figura 52 Mensagem de sucesso após fornecer um email correto para renovação de uma chave API na plataforma CLAV. 84
Figura 53 Email de renovação contendo a nova chave API da plataforma CLAV. 85
Figura 54 Mensagem de sucesso após clicar no link de renovação de uma chave API na plataforma CLAV. 85
Figura 55 Email de renovação contendo a nova chave API da plataforma CLAV. 86
Figura 56 Mensagem de erro após fornecer um email incorreto para renovação de uma chave API na plataforma CLAV. 86
Figura 57 Diagrama de sequência relativo ao processo de registo de uma chave API na plataforma CLAV. 87
Figura 58 Diagrama de sequência relativo ao processo de renovação de uma chave API na plataforma CLAV. 87
Figura 59 Página de listagem de todas as chaves API emitidas pela plataforma CLAV. 88
Figura 60 Página de edição de uma chave API emitida pela plataforma CLAV. 89
Figura 61 Diagrama de sequência relativo à gestão e edição de chaves API na plataforma CLAV. 89
Figura 62 Página responsável pela visualização da métrica sobre a plataforma CLAV. 91
Figura 63 Diagrama de sequência relativo ao processo de recolha de métricas na plataforma CLAV. 92

LISTA DE TABELAS

Tabela 1 Aplicação do algoritmo *SHA-1* a um conjunto de passwords. 6

Tabela 2 Aplicação do algoritmo *SHA-1* com auxílio de *salting* a um conjunto de passwords. 7

Tabela 3 Tempo necessário para testar combinações de passwords para 1000 utilizadores. 59

Tabela 4 Comparação entre hardware no cálculo de H/s, eficiência e custo.(20) 60

SIGLAS

AMA Agência para a Modernização Administrativa.

AP Administração Pública.

API Application programming interface.

ASIC Application-specific integrated circuit.

BI Bilhete de Identidade.

CAP Chip Authentication Program.

CC Cartão de Cidadão.

CLAV Projeto M51-CLAV-Arquivo digital: Plataforma modular de classificação e avaliação da informação pública.

CMD Chave Móvel Digital.

CORS Cross-Origin Resource Sharing.

CPU Central Processing Unit.

DES Data Encryption Standard.

DGLAB Direção-Geral do Livro, dos Arquivos e das Bibliotecas.

EEPROM Electrically-Erasable Programmable Read-Only Memory.

FA Fornecedor de Autenticação.

FPGA Field-programmable gate array.

GPU Graphics Processing Unit.

HMAC Hash-based Message Authentication Code.

HTTP Hyper Text Transfer Protocol.

HTTPS Hyper Text Transfer Protocol Secure.

IAP Interoperabilidade na Administração Pública.

ITU International Telecommunication Union.

JVM Java Virtual Machine.

JWT JSON Web Token.

LC Lista Consolidada.

MAC Message Authentication Code.

MD5 Message-Digest algorithm 5.

MIEI Mestrado Integrado em Engenharia Informática.

NIC Número Identificação Civil.

OASIS Organization for the Advancement of Structured Information Standards.

OCSP Online Certificate Status Protocol.

OWASP Open Web Application Security Project.

PKCS Public-Key Cryptography Standards.

PKI Public key infrastucture.

POSIX Portable Operating System Interface.

PUK Pin Unlock Key.

RSA Rivest-Shamir-Adleman.

SAML Security Assertion Markup Language.

SHA Secure Hash Algorithm.

SSL Secure Sockets Layer.

SSO Single-Sign On.

STORK Secure identity across borders linked.

TDES Triple Data Encryption Standard.

TLS Transport Layer Security.

UM Universidade do Minho.

USENIX The Advanced Computing Systems Association.

XML Extensible Markup Language.

1

INTRODUÇÃO

1.1 ENQUADRAMENTO

Os processos burocráticos, administrativos e logísticos a que hoje as instituições estão sujeitas levam à produção de grandes quantidades de informação que, na sua maioria devem estar disponíveis para consulta a longo-termo, não só por motivos legais, mas também por questões associadas à preservação de dados arquivísticos.

A grande quantidade de informação produzida desencadeia longos e complexos processos que arrecadam para as instituições despesas elevadas e impacto ambiental significativo. A incomportabilidade deste contexto, torna a desmaterialização dos processos fundamental, que se identifica como uma medida enquadrada no eixo estratégico do desenvolvimento sustentável.(8)

O Governo tem neste sentido, desenvolvido estratégias para a transformação digital, nomeadamente, na Administração Pública, tendo por base normas e orientações provenientes da Comissão Europeia. Entre outros, pretende-se a redução do consumo de papel na mesma (3), através da desmaterialização de processos, da promoção da adoção de sistemas de gestão documental eletrónica ou outros, bem como a digitalização de documentos destinados a ser arquivados.

Uma das medidas previstas foi a adoção de processos de «classificação, avaliação e seleção de informação, tendo em consideração, sempre que possível, os princípios de uma Macroestrutura Funcional (MEF) e a Avaliação Supra-Institucional da Informação Arquivística (ASIA)».(12)(11)

Assim sendo, surgiu o "Projeto M51-CLAV-Arquivo digital: Plataforma modular de classificação e avaliação da informação pública", projeto nacional financiado pela *Direção-Geral do Livro, dos Arquivos e das Bibliotecas (DGLAB)* em colaboração com a *Universidade do Minho (UM)*, no âmbito do Aviso N.º 02/SAMA2020/2016(2), para dar cumprimento à Medida 51 do Simplex + "Arquivo digital".

O Simplex é um programa do governo que visa a simplificação legislativa e administrativa, e a modernização dos serviços públicos. Em particular, com o objetivo

de tornar a Administração Pública mais eficiente surgiu o "Arquivo Digital" do Ministério da Cultura. Este pretende utilizar instrumentos transversais de gestão da informação, com o fim de classificar e avaliar todos os documentos produzidos e recebidos nos organismos públicos.

Estes instrumentos serão disponibilizados por uma plataforma modular de serviços partilhados, com a possibilidade de integração com os sistemas de informação existentes em qualquer organismo, sendo que esta plataforma também permite a desmaterialização dos procedimentos, atualmente obrigatórios, para se poder eliminar documentação em papel no Estado.

Através de um projeto colaborativo que envolveu grande parte dos organismos da Admi-nistração Pública, Central e Local, foi originada a *Lista Consolidada (LC)* para a classificação e avaliação da informação pública. Esta LC foi desenvolvida usando uma abordagem supra-institucional e funcional, cujo valor maior é a interoperabilidade semântica, viabilizada pela criação de uma linguagem comum e transversal à Administração Pública, que resultou, entre outros aspetos, na criação de códigos de classificação comuns.

Este trabalho enquadra-se no Projeto CLAV, nas suas terceiras e quarta fase de desenvolvimento, na implementação de uma aplicação Web para a gestão e manutenção da plataforma CLAV.

1.2 OBJETIVOS

A realização desta dissertação tem como objetivo os seguintes pontos:

- Estudo e interpretação dos serviços da plataforma Autenticação.Gov e respetivos requisitos.
- Integração e desenvolvimento de métodos apropriados de registo e login de utilizadores, bem como a gestão dos mesmos, utilizando a ferramenta Autenticação.Gov.
- Criação de métodos apropriados para autenticação de pedidos via a API pública disponibilizada.
- Avaliação dos métodos gerados, garantindo assim uma correta implementação dos mesmos na plataforma CLAV.

1.3 METODOLOGIA

A metodologia para a realização desta dissertação segue os seguintes passos:

- Leitura e análise inicial do projeto.
- Estudo das ferramentas já implementadas.
- Estudo da ferramenta Autenticação.Gov e requisitos por ela impostos.
- Desenvolvimento de um protótipo com funcionalidades semelhantes ou idênticas à esperada.
- Implementação na plataforma CLAV.

1.4 SÍNTESE

Neste capítulo foi realizada uma breve introdução e enquadramento desta dissertação no projeto CLAV, listando os objetivos propostos durante o desenvolvimento da mesma, bem como a metodologia adoptada.

No capítulo seguinte irá ser documentado o estado da arte atual, sendo focado em especial atenção o Autenticação.Gov, sendo descrito o seu workflow, funcionamento e especificação técnica. Além deste, será também mencionada a especificação do Cartão de Cidadão e da proteção da API de dados disponibilizada ao público.

2

ESTADO DA ARTE

A plataforma CLAV pode ser divididas em 3 eixos principais:

1. **Frontend** Responsável pela interação dos utilizadores com plataforma, bem como chamadas à *Application programming interface (API)*, podendo esta ser de cariz público ou privado, através do acesso a certas funcionalidades da plataforma (por exemplo: listagens de utilizadores, entidades, legislações, etc).
2. **Backend** Desenvolvido em *NodeJS*, este é responsável por satisfazer os serviços requisitados pelo *Frontend*. Inclui toda a lógica da aplicação, como a camada de acesso a dados, leitura e armazenamento de informação.
3. **API de dados** Responsável pela comunicação e gestão de pedidos dos utilizadores da plataforma, sendo que a sua principal função é servir de elo de ligação entre os resultados guardados em base de dados e o *Frontend* disponibilizado aos utilizadores.

Devido à natureza dos dados presentes na plataforma CLAV, foi necessário proceder a uma correta e estruturada implementação de diversos mecanismos de segurança e proteção contra uso indevido de dados.

2.1 ENCRIPTAÇÃO E CRIPTOGRAFIA

Em qualquer aplicação no mundo real, é impensável guardar dados sensíveis, como passwords, em formato *plain-text*. Daí surgiu a necessidade de proteger informação sensível através de métodos designados por *hashing*[1].

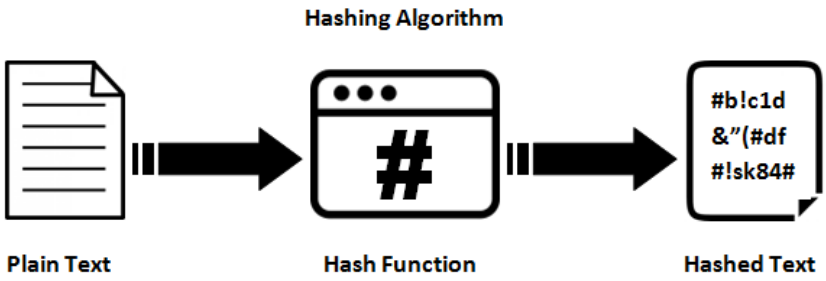


Figura 1: Representação de uma função de *hash*.(13)

Em geral, o *hashing* é caraterizado como uma operação de sentido único, através da qual se gera um resultado único, impossível de reverter para o texto original. Embora esta noção esteja correta e seja de facto impossível, através de um texto já processado por uma função de *hash*, reconstruir o texto original em questão, este não é infalível.

2.1.1 *Vulnerabilidades*

Para exemplificar vulnerabilidades existentes nas funções criptográficas que recorrem apenas ao *hashing* de passwords, vamos recorrer ao algoritmo **SHA-1**, pertencente à familia de algoritmos *Secure Hash Algorithm (SHA)*.

O principal problema das funções da família SHA assenta no facto de que foram desenvolvidas para serem computacionalmente rápidas. A rapidez com que uma função pode calcular *hashes*, tem um impacto imediato e significativo na quão seguro uma password é.

Embora cálculos mais rápidos levem ao desenvolvimento de algoritmos mais eficientes a nível de computação, também abrem portas a ataques *bruteforce*. Atualmente, com o auxilio de *Central Processing Unit (CPU)* e *Graphics Processing Unit (GPU)* modernas, é possível calcular milhões, ou até bilhões de *hashes* por segundo.

1 Uma função hash é um algoritmo que mapeia dados de comprimento variável para dados de comprimento fixo.

Outro problema assenta no facto que qualquer função responsável por *hashing*, para o mesmo texto, retorna sempre o mesmo resultado de *hash*, como podemos verificar na tabela 1.

Resultados do algoritmo SHA-1		
Utilizador	Password	*Hashed* password
António	12345	**8cb2237d0679ca88db6464eac60da96345513964**
Alice	sup3rs3gur4	0ce594a80be23686fa95527e219ff162291f80f0
Manuel	uncrackabl3	c4f9036ecefde84b5a8dc4296abbab1a6c53be60
Gustavo	12345	**8cb2237d0679ca88db6464eac60da96345513964**
Joana	1a2b3c4d	b01afc2b077956acc69f99e0b7df1cb70cb01331

Tabela 1: Aplicação do algoritmo *SHA-1* a um conjunto de passwords.

Como exemplificado na Tabela 1, o utilizador **António** e **Gustavo** partilham a mesma password, logo o resultado após o *hashing* da mesma é idêntico.

Embora passwords idênticas sejam extremamente comuns, o facto do *hash* resultante ser idêntico apresenta um factor de risco muito mais elevado do que as passwords serem idênticas, pois estaria a expor um número exorbitante de utilizadores a uma grave falha de segurança.

Porém, o simples acto de *hashing* de passwords não é uma solução segura. Este rapidamente foi descartado devido a ser extremamente susceptível a ataques baseado em *rainbow tables*, também conhecidos por *ataques de dicionário*.

Este ataque tem como base o facto de algoritmos como o *SHA-1* ser extremamente rápido e eficiente, sendo que em vez de calcular em tempo real *hashes* aleatórios, utilizam valores de hash pré-calculados para toda e qualquer possível combinação de caracteres, utilizando para validação o método *bruteforce*.

Esta vulnerabilidade foi de tão larga escala, que poderia quebrar *99.9%* de todas as combinações possíveis de 14 caracteres alfanuméricos em 11 minutos (utilizando a *rainbow table* menos extensa, sendo que com tabelas mais extensas esta figura descia consideravelmente).

2.1.2 *Melhorias no processo de hashing*

De modo a solucionar o problema anterior, é necessário implementar uma função de *hashing* mais lenta, que seja eficaz na proteção de informação e capaz de abrandar, ou evitar possíveis ataques. Também é necessário que esta função seja adaptativa, ou seja, que devido a avanços tecnológicos em hardware esta se possa adaptar para ter um desempenho similar aos níveis atuais.

De modo a solucionar o problema proveniente de ataques *bruteforce* baseados no uso de *rainbow tables*, foi necessário diversificar ainda mais os *hashes* gerados. Para tal foi adicionado um *salt*(18), de modo a fazer qualquer password verdadeiramente única.

De acordo com a *Open Web Application Security Project (OWASP)*, o *salt* é um valor aleatório de tamanho fixo, considerado criptograficamente forte, que é adicionado ao input de uma dada função de *hash*, independentemente deste do input ser ou não único.

Na tabela 2 podemos verificar a importância de utilizar a técnica de *salting* juntamente com o *hashing* de passwords.

Resultados do algoritmo SHA-1 c/ salting			
Utilizador	Password	Salt	*Hashed* password + **salt**
António	12345	r8ZGQH	**c783b62c876b77310818b3b4cc6863ca008f7d10**
Alice	sup3rs3gur4	yvL9H8	d68c7166f916b91b66dda685e8eee1af70528933
Manuel	uncrackabl3	C4uHRv	87d8e4e7ae79ab2487133a6513e35cb511687d5a
Gustavo	12345	jKM2Lh	**f69e3288fc17c383cab12aa78a725d675610e81a**
Joana	1a2b3c4d	WNj7Vt	866038b3d43de9a9b0b49a6ee1ffc6cbf64d3c6d

Tabela 2: Aplicação do algoritmo *SHA-1* com auxílio de *salting* a um conjunto de passwords.

Como exemplificado na Tabela 2, embora o utilizador **António** e **Gustavo** partilhem a mesma password, o resultado do *hashing* é diferente.

Facilmente chegamos à conclusão que o *salting* é essencial para manter a segurança de informação importante como passwords, pois um conjunto infinito de passwords idênticas nunca terão o mesmo *hash*.

Embora seja teoricamente possível quebrar este tipo de combinação, esta requer poder computacional exponencialmente maior que um simples ataque *rainbow table*, visto que o *salt* é totalmente aleatório, tornando qualquer ataque inviável.

Em suma, o método de autenticação ideal deve implementar ambos estes métodos(7), *hashing* e *salting*, sendo que um dos métodos mais comuns e seguros da atualidade, é o *bcrypt*, tornando-a no perfeito candidato para utilização neste projeto.

2.2 AUTENTICAÇÃO.GOV

O Autenticação.Gov surge da necessidade de identificação unívoca de um utilizador perante sítios na Web. Cabe a esta solução o processo de autenticação e o fornecimento dos atributos do utilizador necessários a que cada entidade possa efetuar a identificação do utilizador.

O Autenticação.Gov, em conjunto com o *Cartão de Cidadão (CC)*, também permite fazer uso da funcionalidade de Federação de Identidades da *Interoperabilidade na Administração Pública (iAP)* para a identificação sectorial de um Cidadão, isto é, a obtenção dos seus identificadores junto das entidades participantes da iniciativa do Cartão de Cidadão.

É também responsável pela gestão dos vários fornecedores de atributos disponíveis e possui uma estreita ligação com a infraestrutura de chave pública do Cartão de Cidadão (*Public key infrastucture (PKI)*), com o intuito de manter elevados níveis de segurança e privacidade no processo de autenticação e identificação, utilizando para tal a linguagem *Security Assertion Markup Language (SAML)*, sendo esta explicada em maior detalhe na secção 2.2.4.1.

Através do Autenticação.Gov é possível a criação de credenciais comuns a todos os sites da Administração Pública, assegurando que o utilizador se necessita de autenticar apenas uma única vez para executar um ou vários serviços que podem ser iniciados em portais transversais, sendo este o fundamento do *Single-Sign On (SSO)*.

Permite também proceder à autenticação de um utilizador com recursos a outros certificados digitais que não o do Cartão de Cidadão, possibilitando e alargando o leque de autenticação disponível para as Entidades que pretendam delegar a autenticação nesta componente. De seguida podemos verificar a quantidade de utilizadores únicos e autenticações realizadas com o auxílio da plataforma *Autenticação.Gov*.

No âmbito do projeto CLAV irá ser implementada a autenticação através dos seguintes meios de autenticação:

- **Obrigatório:** Cartão de Cidadão (CC)

  A utilização do Cartão de Cidadão como meio de autenticação representa mais de 50% de todos as autenticações realizadas na plataforma *Autenticação.Gov*.

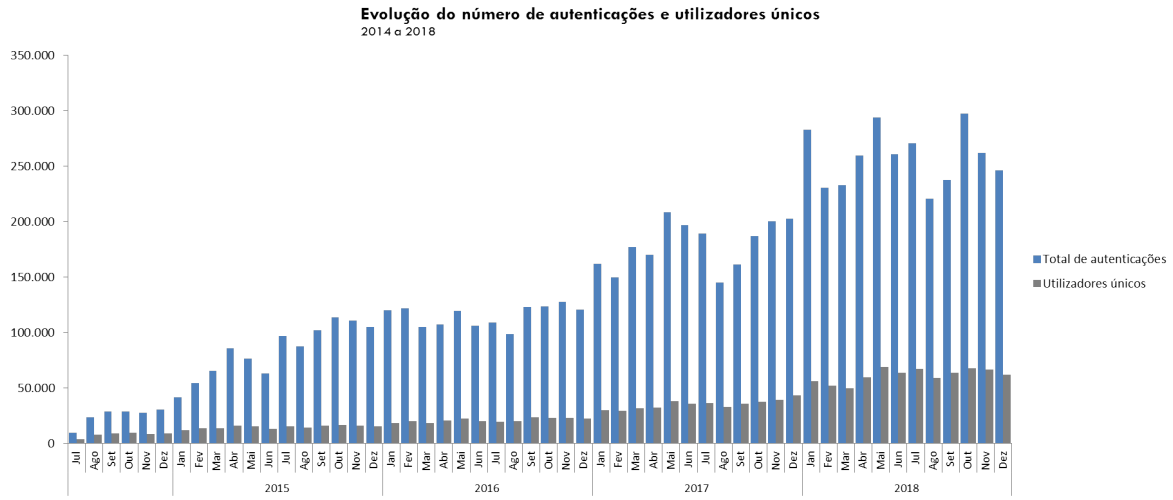


Figura 2: Evolução do número de autenticações realizadas e utilizadores únicos baseados na plataforma *Autenticação.Gov*.(15)

Ano	Autenticações
2014	147.796
2015	1.001.277
2016	1.381.474
2017	2.148.304
2018	3.093.017
Total	**7.771.868**

Figura 3: Representação do número total de autenticações realizados com a plataforma *Autenticação.Gov*.(15)

Ano	Autenticações
2014	145.061
2015	626.926
2016	747.851
2017	1.253.308
2018	1.590.881
Total	**4.364.027**

Figura 4: Evolução do número de Autenticações realizadas através do Cartão de Cidadão.(15)

- **Opcional:** *Chave Móvel Digital (CMD)*

  Com a crescente adesão a serviços de autenticação baseadas em *smartphones*, a utilização da Chave Móvel Digital tem vindo a ganhar popularidade nos últimos anos, provando ser uma opção complementar, bem como viável, à utilização do Cartão de Cidadão como meio de autenticação primária.

Ano	Nº de registos
2014	291
2015	7.007
2016	10.921
2017	41.854
2018	182.205
Total	**242.278**

Figura 5: Evolução do número de Autenticações realizadas através da Chave Móvel Digital.(15)

2.2.1 *Principais funcionalidades*

O Autenticação.Gov é considerado o principal componente para autenticação, a nível nacional e internacional, sendo que as funcionalidades por ele introduzidas permitem a normalização do processo de autenticação e assinatura eletrónica.

Esta autenticação permite às entidades que a requisitam a recolha de informação do utilizador, sendo esta sempre autorizada pelo mesmo.

As principais funcionalidades do Autenticação.Gov são as seguintes:

1. **Identificação através do Cartão de Cidadão**

   O Autenticação.Gov tem como objetivo a identificação segura de um Cidadão, sendo esta baseada na credenciação do mesmo durante a emissão do Cartão de Cidadão, aliado à Federação de Identidades da Plataforma de Interoperabilidade da Administração Pública.

2. **Disponibilização de informação**

   A utilização do Cartão de Cidadão em conjunção com o Autenticação.Gov permite a recolha de informação do Cidadão, através de atributos e identificadores, como por exemplo *NIF* e *NomeCompleto*, através da utilização da Plataforma de Interoperabilidade.

3. **Single Sign-On (SSO)**

   Através do SSO é possível a navegação entre as várias entidades aderentes, com fornecimento dos atributos necessários para autenticação e garantia do nível de segurança proporcionado pelo Autenticação.Gov.

4. **Simplificação do processo de autenticação**

   O processo de autenticação é delegado ao Autenticação.Gov, sendo este responsável por assegurar a validade de certificados, obtenção dos atributos pedidos e a devolução dos mesmos à entidade que os solicitou.

5. **Normalização do processo de autenticação**

   O processo previamente descrito é realizado com auxilio de vários níveis de segurança e qualidade de serviço. Assim sendo, é garantida a utilização da estrutura da chave pública do Cartão de Cidadão, conhecida por PKI, com recurso a validação através de *Online Certificate Status Protocol (OCSP)* dos certificados de autenticação fornecidos.

6. **Transparência no processo de autenticação**

   Sendo o cidadão parte ativa na transmissão de atributos à entidade que o solicita, este tem de dar a sua explícita permissão, podendo negar acesso a certos atributos opcionais, ou até proceder ao cancelamento do processo de autenticação corrente.

2.2.2 *Workflow do Autenticação.Gov*

O workflow do processo de autenticação com o Autenticação.Gov cinge-se a dois possíveis cenários:

1. **Primeira autenticação**

   Este ocorre quando o cidadão/utilizador não possui sessão ativa no portal do Autenticação.Gov, quer por a sessão prévia ter expirado ou esta ser inexistente.

2. **Re-autenticação ou Renovação de uma autenticação prévia**

   Quando o cidadão/utilizador já teve sessão iniciada numa entidade utilizadora do Autenticação.Gov, este pode simplesmente confirmar a cedência de dados à plataforma CLAV, não havendo a necessidade de nova introdução de PIN de autenticação, excepto num caso explorado na secção 2.2.2.2.

2.2.2.1 *Primeira autenticação*

De acordo com o manual de Integração do Autenticação.Gov(6) providenciado pela *Agência para a Modernização Administrativa (AMA)*, existem 4 interações entre o *Autenticação.Gov* e a entidade/serviço a aceder, especificadas na figura 6.

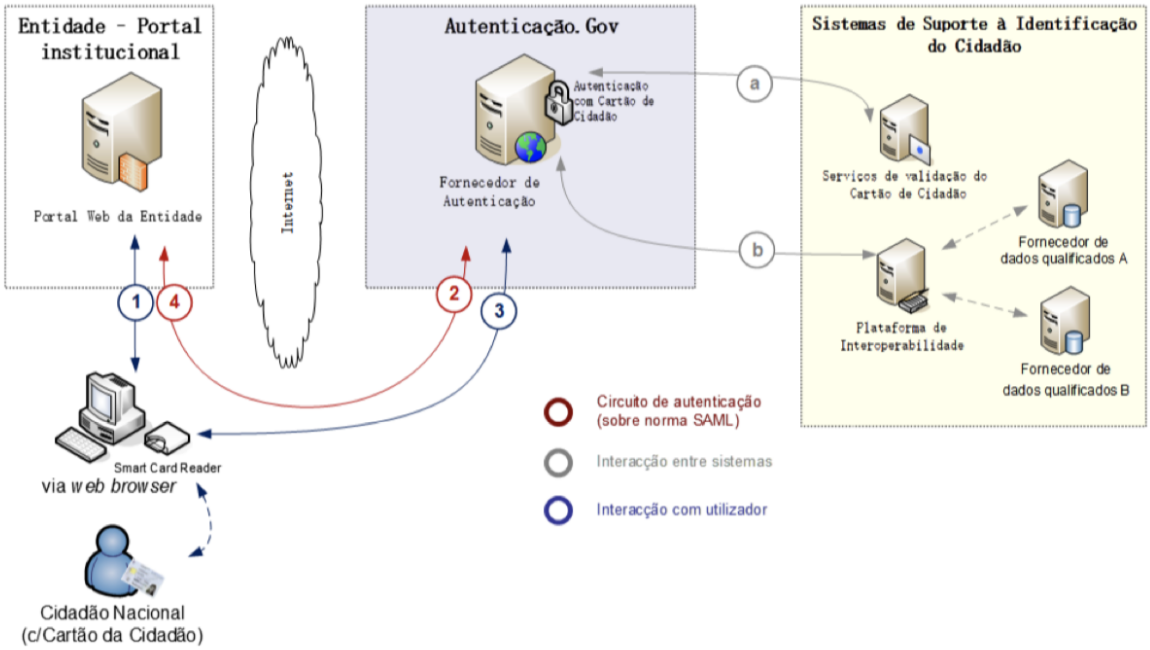


Figura 6: Workflow de autenticação via Cartão de Cidadão(6).

1. O utilizador pretende aceder à área privada do portal de uma entidade, na qual é necessário que comprove a sua identidade (neste caso a plataforma CLAV).

2. O portal da entidade delega a autenticação e redireciona o utilizador para o Auten-ticação.Gov, juntamente com um pedido de autenticação assinado digitalmente.

3. O Autenticação.Gov valida o pedido de autenticação recebido e solicita a autenticação do utilizador com recurso ao seu Cartão de Cidadão pedindo a inserção do seu PIN de autenticação. Durante este processo, o Autenticação.Gov efetua as seguintes operações internas:

   a) Valida as credenciais do utilizador com recurso à PKI do Cartão de Cidadão via OCSP.

   b) Obtém atributos que sejam solicitados pelo portal da entidade junto dos vários fornecedores de atributos qualificados. Esta operação é efetuada via Plataforma de Interoperabilidade. Este processo pode incluir a obtenção de dados da Fe-deração de Identidades ou de outras Entidades.

4. A identificação e atributos do utilizador são autenticadas e assinados digitalmente pelo Autenticação.Gov, após o que redirecciona o utilizador de volta ao portal da entidade original. Cabe à entidade a validação das credenciais do Autenticação.Gov e utilização dos atributos do cidadão.

2.2.2.2 *Renovação de autenticação*

Além das etapas numeradas entre **1** e **4** da figura 6, existe a possibilidade de o utilizador já ter sessão iniciada noutro portal da AP, como por exemplo, a Segurança Social Direta[2].

A utilização do Autenticação.Gov tem como principal função a simsimplificação do processo de autenticação nos vários portais da AP. Ou seja, este permite a autenticação de utilizadores entre diversos sites da AP ou entidades privadas, como por exemplo a plataforma CLAV, solicitando as credenciais do mesmo apenas uma vez, sendo estas revalidadas junto do Autenticação.Gov sem necessidade de nova inserção do PIN de autenticação.

Esta capacidade de re-autenticação sem nova inserção do PIN de autenticação é exemplificada na figura 7.

2 Mais informações em https://app.seg-social.pt/ptss/

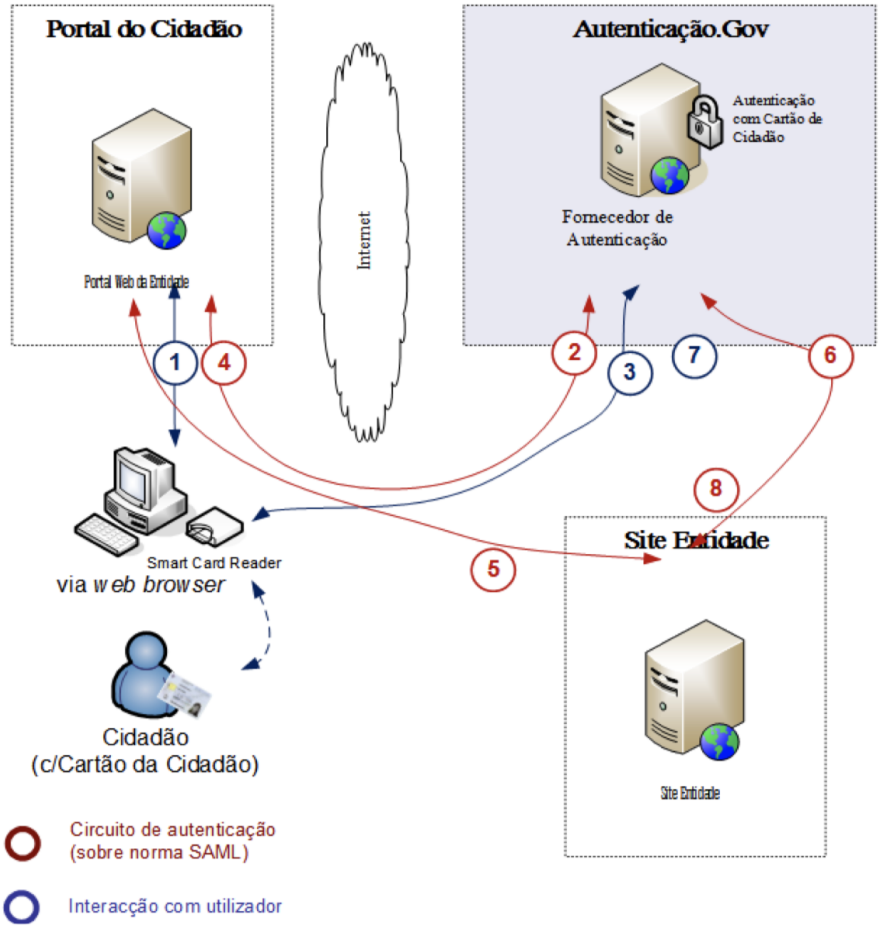


Figura 7: Workflow de re-autenticação via Cartão de Cidadão(6).

5. É feito o acesso a uma nova entidade, neste caso a plataforma CLAV, tendo o utilizador feito a autenticação na entidade prévia.

6. O site da entidade revalida a credencial eletrónica junto do Autenticação.Gov.

7. Cabe ao Autenticação.Gov re-emitir ou revalidar a credencial consoante os seguintes dois casos:

   a) Se forem solicitados os mesmos atributos da última autenticação e estes tenham sido obtidos com um nível de confiança igual ou superior, não será necessária nova introdução de PIN.

   b) Caso sejam pedidos atributos diferentes, então o Autenticação.Gov irá requisitar ao utilizador nova inserção de PIN.

8. O site de entidade recebe a resposta do Autenticação.Gov e autentica utilizador em questão.

Através desta implementação de renovação de autenticação, é assegurada a correta imple-mentação de um serviço SSO, referida na secção 2.2.

2.2.3 *Funcionamento do Autenticação.Gov*

De modo a utilizar o Autenticação.Gov é necessária a criação de um pedido baseado em SAML, assinado digitalmente com um certificado X.509 e encriptado com a nossa chave *Rivest-Shamir-Adleman (RSA)* privada, bem como a respetiva cadeia de autenticação.

Existem dois tipos de pedidos SAML utilizados pelo Autenticação.Gov:

- **SAML Request**

  Pedido de autenticação (*AuthnRequest*) em formato SAML, enviado ao Autenticação.Gov, no qual é enviada informação sobre a origem, assinaturas, atributos a requisitar, etc.

- **SAML Response**

  Resposta (*AuthnResponse*) em formato SAML ao pedido previamente referido, enviado em binding *HTTP-POST* para o Issuer do pedido de autenticação.

  Na resposta são enviados os dados previamente requisitados, bem como resposta de sucesso, insucesso ou cancelamento do pedido de autenticação por parte do utilizador.

Figura 8: Funcionamento do Autenticação.Gov na plataforma CLAV.

Consoante a resposta do Autenticação.Gov, o login do utilizador é autorizado ou negado, consoante exemplo na figura 8. Este apenas é negado caso o utilizador erre o PIN de autenticação, negue acesso aos dados pedidos, ou não tenha leitor smart-card no seu computador.

2.2.3.1 *AuthnRequest*

Um pedido de autenticação *(AuthnRequest)* é escrito na linguagem SAML, sendo esta baseada em *Extensible Markup Language (XML)*.

Como podemos verificar no exemplo em baixo, este pedido segue o formato standard do XML, tendo como elemento raíz o *AuthnRequest*.

Contidos dentro do *AuthnRequest* está o elemnto *Issuer*, o elemento *Signature* e por fim o elemento *Extensions*, sendo estes explorados posteriormente nesta secção.

```
<AuthnRequest
    xmlns="urn:oasis:names:tc:SAML:2.0:protocol"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    ID="_63f47093-6377-4771-a36e-8bc506cf9e64"
    Version="2.0"
    IssueInstant="2019-08-20T14:51:18.061Z"
    Destination="https://preprod.autenticacao.gov.pt/fa/Default.aspx"
    ProtocolBinding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
    AssertionConsumerServiceURL="http://clav-test.di.uminho.pt/api/users
    /callback"
    ProviderName="CLAV">
    <Issuer xmlns="urn:oasis:names:tc:SAML:2.0:assertion">
        ...
    </Issuer>
    <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
        ...
    </Signature>
    <Extensions>
        ...
    </Extensions>
</AuthnRequest>
```

O elemento de raiz, *AuthnRequest*, contém os seguintes atributos, sendo estes de caráter obrigatório para a correta criação de um pedido de autenticação SAML.

- **ID**

  Atributo baseado no tipo de dados *xs:ID*[3], sendo este gerado com recurso ao *UUID/v4*. Devido à especificação do Autenticação.Gov, este ID gerado tem de ser precedido de um "_".

3 Definido em: https://www.w3.org/TR/xmlschema-2/#ID

- **Version**

  Atributo responsável pela identificação da versão SAML utilizada neste pedido. Neste caso, o Autenticação.Gov está configurado para apenas funcionar com a versão 2.0.

- **IssueInstant**

  Atributo cujo valor identifica a data de emissão do pedido SAML em formato UTC[4].

- **Destination**

  Atributo responsável pelo URL para o qual o pedido SAML é enviado, sendo apenas possível um de dois URL:

  – **Ambiente de teste** `https://preprod.autenticacao.gov.pt/fa/Default.aspx`

    O ambiente de teste é utilizado para desenvolvimento de aplicações antes destas serem libertadas ao público, não sendo forçado o uso de *HTTPS* nem comunicação por *SSL/TLS*.

  – **Ambiente de produção** `https://autenticacao.gov.pt/fa/Default.aspx`

    Este ambiente é utilizado numa fase final da aplicação, ou seja, quando a mesma está pronta para uso do público geral, sendo nesta fase obrigatório o uso de *HTTPS* e comunicação com auxílio de *SSL/TLS*.

  No momento de escrita desta dissertação, a plataforma CLAV utiliza o ambiente de teste.

- **ProtocolBinding**

  Atributo que identifica o *binding* a utilizar aquando da criação do pedido SAML. No manual do autenticação(6) é especificado que apenas é utilizado o binding *HTTP-POST*.

- **AssertionConsumerServiceURL**

  Este atributo é povoado com o URL com o qual o Autenticação.Gov irá utilizar para enviar a resposta (*AuthnResponse*), ao pedido de autenticação.

  Em ambiente de produção, o URL em causa tem de utilizar *HTTPS*.

- **ProviderName**

4 Definido em: `http://www.w3.org/TR/xmlschema-2/#dateTime`

Nome do fornecedor de software, plataforma ou aplicação que está a requerir a autenticação via Autenticação.Gov.

Este valor tem de ser previamente acordado com a AMA durante a emissão da cadeira de autenticação e certificados X509.

Como foi referido no inicio desta secção, um pedido de autenticação *(AuthnRequest)* é composto por 3 elementos, todas incorporadas dentro de um *AuthnRequest*:

1. **Issuer**

   Neste elemento é enviada informação sobre quem está a requisitar a autenticação, neste caso a plataforma CLAV, sendo apenas necessário o envio do *URL* onde origina o pedido(6).

```
<Issuer xmlns="urn:oasis:names:tc:SAML:2.0:assertion">
    http://clav-auth.di.uminho.pt
</Issuer>
```

2. **Signature**

   Neste elemento é realizada a assinatura digital do pedido SAML, através de métodos de canonização e transformação especificados no manual de integração(6).

   Posteriormente à assinatura, é descrito qual o algoritmo utilizado para o cálculo do *DigestValue*, neste caso *SHA-1*, bem como o valor resultante deste processo.

   Por fim, é adicionado o certificado X.509 contendo a respetiva cadeia de autenticação fornecido pela AMA.

```
<Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
    <SignedInfo>
        <CanonicalizationMethod Algorithm="http://www.w3.org/2001/xml-exc-
            c14n"/>
        <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-
            sha1"/>
        <Reference URI="#_63f47093-6377-4771-a36e-8bc506cf9e64">
            <Transforms>
                <Transform Algorithm="http://www.w3.org/2000/09/xmldsig#
                    enveloped-signature"/>
                <Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#
                    "/>
            </Transforms>
            <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/
                >
```

```
            <DigestValue>
                msbCCbP1sltYLV...oaZ+nNmWwVHqE=
            </DigestValue>
        </Reference>
    </SignedInfo>
    <SignatureValue>
            gB656A8kkR7TujX8/xfAfYJm...2EX3iK7rXxAd5HAbvB6zV1BXvDYLJ6K+ZOBGF
    </SignatureValue>
    <KeyInfo>
        <X509Data>
            <X509Certificate>
                9AnXwiE5zOybPs8CAwEAAaOB...BlrBlEXYrXxFdIA7qATy
            </X509Certificate>
        </X509Data>
    </KeyInfo>
</Signature>
```

3. **Extensions**

   Este elemento contém os atributos que vamos requisitar ao Autenticação.Gov, aquando da autenticação de um utilizador na plataforma CLAV.

   Neste caso o seu *Número Identificação Civil (NIC)* e Nome Completo, sendo os atributos requisitados de caráter obrigatório devido ao atributo *isRequired* ser *true*.

```
<Extensions>
    <fa:RequestedAttributes xmlns:fa="http://autenticacao.cartaodecidadao.pt
       /atributos">
        <fa:RequestedAttribute
            Name="http://interop.gov.pt/MDC/Cidadao/NIC"
            NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
            isRequired="True"/>
        <fa:RequestedAttribute
            Name="http://interop.gov.pt/MDC/Cidadao/NomeCompleto"
            NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
            isRequired="True"/>
    </fa:RequestedAttributes>
</Extensions>
```

2.2.3.2 *AuthnResponse*

Após a recepção de um pedido *AuthnRequest*, o Autenticação.Gov processa a autenticação do utilizador e retorna o seguinte pedido como resposta.

Tal como o pedido de autenticação *AuthnRequest* explorado na secção 2.2.3.1, este é escrito na linguagem SAML.

Outras parecenças com o pedido de autenticação incluem o facto de também possuir o elemento *Issuer*, *Signature* e *Extensions*. Iremos explorar em detalhe os elementos que não foram incluídos na secção 2.2.3.1 bem como os que sofreram alterações quando comparados a um pedido de autenticação *AuthnRequest*:

```
<Response
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    ID="_3305700e-0ade-44b1-b81f-a10a5841d56b"
    InResponseTo="_63f47093-6377-4771-a36e-8bc506cf9e64"
    Version="2.0"
    IssueInstant="2019-08-21T17:52:40.4910867Z"
    Destination="http://clav-test.di.uminho.pt/api/users/callback"
    xmlns="urn:oasis:names:tc:SAML:2.0:protocol">
    <Issuer xmlns="urn:oasis:names:tc:SAML:2.0:assertion">
        ...
    </Issuer>
    <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
        ...
    </Signature>
    <Extensions>
        ...
    </Extensions>
    <Status>
        ...
    </Status>
    <Assertion
        Version="2.0"
        ID="_116a2041-ed1a-445c-8b3f-95baef3f5e00"
        IssueInstant="2019-08-21T17:52:40.4910867Z"
        xmlns="urn:oasis:names:tc:SAML:2.0:assertion">
        ...
    </Assertion>
</Response>
```

O elemento de raiz, *Response*, contém os seguintes atributos, sendo estes de caráter obrigatório numa resposta a um pedido de autenticação SAML.

- **ID**

  Atributo baseado no tipo de dados *xs:ID*[5]. Devido à especificação do Autenticação.Gov, este ID gerado tem de ser precedido de um "_".

- **InResponseTo**

  Atributo cujo valor reflete o *ID* do pedido de autenticação (não confundir com o *ID* da resposta ao pedido de autenticação).

- **Version**

  Atributo responsável pela identificação da versão SAML utilizada neste pedido. Neste caso, o Autenticação.Gov está configurado para apenas funcionar com a versão 2.0.

- **IssueInstant**

  Atributo cujo valor identifica a data de emissão da resposta SAML em formato UTC[6].

- **Destination**

  Atributo responsável pelo URL para o qual a resposta SAML é enviada. É identico ao *AssertionConsumerServiceURL* do pedido de autenticação exemplificado na secção 2.2.3.1.

Como foi referido no inicio desta secção, uma resposta a um pedido de autenticação *(Response)* é composto por 5 elementos, todas incorporadas dentro de um *Response*:

1. **Issuer**

   Neste elemento é enviada informação sobre quem está a responder ao pedido de autenticação, neste caso o URL do serviço responsável pelo Cartão de Cidadão.

   ```
   <Issuer xmlns="urn:oasis:names:tc:SAML:2.0:assertion">
       https://autenticacao.cartaodecidadao.pt
   </Issuer>
   ```

5 Definido em: https://www.w3.org/TR/xmlschema-2/#ID
6 Definido em: http://www.w3.org/TR/xmlschema-2/#dateTime

2. **Signature**

   Elemento idêntico ao referido na secção 2.2.3.1.

3. **Extensions**

   Neste elemento é devolvido o nível de confiança utilizado pela autenticação previamente realizada.

   Como no *AuthnRequest* da secção 2.2.3.1 não foi especificado nenhum nível de confiança, o Autenticação.Gov assume que se trata de nível 4, requerindo para tal a autenticação através de Cartão de Cidadão.

```
<Extensions>
    <fa:FAAALevel xmlns:fa="http://autenticacao.cartaodecidadao.pt/
        atributos">
        4
    </fa:FAAALevel>
</Extensions>
```

   Além do nível 4, este pode tomar os seguintes valores:

   a) **Nível 3**

      Autenticação com Chave Móvel Digital.

   b) **Nível 2**

      Autenticação com Chave Móvel Digital através do Twitter.

   c) **Nível 1**

      Autenticação com Utilizador/Palavra-passe e Redes Sociais.

4. **Status**

   Neste elemento é enviada informação relativa ao sucesso ou insucesso do pedido de autenticação.

   Caso este tenha sido realizado com sucesso é enviado o seguinte elemento de *Status*:

```
<Status>
    <StatusCode Value="urn:oasis:names:tc:SAML:2.0:status:Success"/>
</Status>
```

Caso contrário, alem do atributo de estado de primeiro nível (representado pelo *StatusCode)*, é enviado um segundo nível designado por estado subordinado.

Este apenas é obrigatório no caso em que a autenticação falhe, sendo povoado com o motivo pelo qual esta falhou.

```
<Status>
    <StatusCode Value="urn:oasis:names:tc:SAML:2.0:status:Responder">
        <StatusCode Value="urn:oasis:names:tc:SAML:2.0:status:AuthnFailed"
            />
    </StatusCode>
</Status>
```

a) **Estados de primeiro nível**

i. **urn:oasis:names:tc:SAML:2.0:status:Success**

Este estado é retornado quando a operação de autenticação é realizada com sucesso.

ii. **urn:oasis:names:tc:SAML:2.0:status:Requester**

Este estado é retornado quando ocorre uma falha com o fornecedor de serviços, neste caso a plataforma CLAV e a autenticação não é realizada.

iii. **urn:oasis:names:tc:SAML:2.0:status:Responder**

Este estado é retornado quando ocorre uma falha com o Autenticação.Gov e a autenticação não é realizada.

b) **Estados subordinados**

i. **urn:oasis:names:tc:SAML:2.0:status:AuthnFailed**

Ocorre quando a autenticação falhou por algum motivo alheio ao Autenticação.Gov. Pode ser por um destes vários fatores:

A. Utilizador negou leitura do Cartão de Cidadão.

B. Utilizador errou o PIN de autenticação.

C. Utilizador negou a recolha dos atributos pedidos.

D. Utilizador não tem leitor smart-card.

E. Utilizador não tem plugin do Autenticação.Gov instalado.

F. etc.

ii. **urn:oasis:names:tc:SAML:2.0:status:InvalidAttrNameOrValue**

Ocorre quando é requisitado um valor ou atributo inexistente ou inválido.

iii. **urn:oasis:names:tc:SAML:2.0:status:RequestDenied**

Ocorre quando o pedido de autenticação se encontra correto, mas por algum motivo o Autenticação.Gov decidiu optar por não lhe responder.

iv. **urn:oasis:names:tc:SAML:2.0:status:undefined**

Ocorre quando o erro contemplado não se encontra nos listados previamente.

5. **Assertion**

Este elemento contem uma asserção SAML, ou seja, um pacote de informações de segurança.

Esta asserção especifica que a resposta foi emitida por uma entidade num determinado momento e atesta a identidade da entidade da mesma, desde que as condições especificadas de validação tenham sido satisfeitas.

```
<Assertion>
    <AttributeStatement>
        <Attribute
            Name="http://interop.gov.pt/MDC/Cidadao/NIC"
            NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
            d4p1:AttributeStatus="Available"
            xmlns:d4p1="http://autenticacao.cartaodecidadao.pt/atributos">
            <AttributeValue xsi:type="xsd:string">
                12345678
            </AttributeValue>
        </Attribute>
        <Attribute
            Name="http://interop.gov.pt/MDC/Cidadao/NomeCompleto"
            NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
            d4p1:AttributeStatus="Available"
            xmlns:d4p1="http://autenticacao.cartaodecidadao.pt/atributos">
            <AttributeValue xsi:type="xsd:string">
                OCTAVIO JOSE AZEVEDO MAIA
            </AttributeValue>
```

```
        </Attribute>
    </AttributeStatement>
</Assertion>
```

A asserção apenas pode conter um elemento *AttributeStatement*, sendo que este é povoado com os atributos requisitados no pedido de autenticação *AuthnRequest* especificado na secção 2.2.3.1.

Neste caso, foi requisitado o valor do Número de Identificação Cívil (**12345678**) e o Nome Completo do cidadão (**OCTAVIO JOSE AZEVEDO MAIA**).

2.2.4 *Especificação do Autenticação.Gov*

Após ter sido explorado na secção 2.2.3 o funcionamento do Autenticação.Gov, nesta secção iremos explicar os diversos métodos que permitem a correta criação dos pedidos de autenticação em formato SAML, quer seja por auxílio a métodos de encriptação, assinatura ou comunicação.

1. **SAML 2.0**

   O formato de dados trocados entre o Autenticação.Gov e as entidades é baseado em SAML 2.0, de forma a assegurar a autenticidade e integridade de todas as transações.

   A utilização do SAML HTTP-Post Binding associado ao SAML Web Browser SSO Profile permite que a autenticação seja efetuada pelo browser do utilizador, sem necessidade de ligação física entre a plataforma CLAV e o Autenticação.Gov.

2. **SSL e HTTPS**

   A comunicação entre o Autenticação.Gov e a plataforma CLAV é efetuada sobre HTTPS em canal cifrado – *Secure Sockets Layer (SSL)* ou *Transport Layer Security (TLS)*.

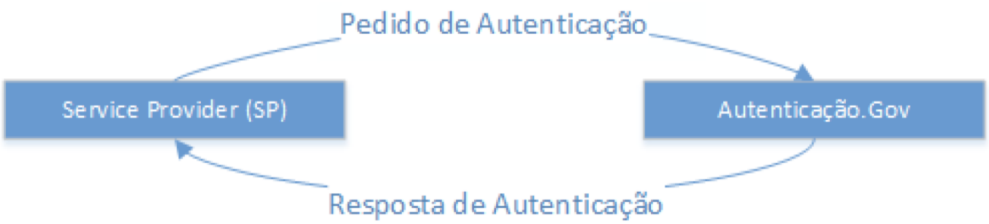


Figura 9: Comunicação entre o Autenticação e o Fornecedor de Serviços (plataforma CLAV)

.

O Autenticação.Gov responde à plataforma CLAV com informação autorizada pelo utilizador. A resposta inclui os atributos solicitados no pedido de autenticação. Esta ligação é também efetuada sobre HTTP em canal cifrado – cifrado – SSL ou TLS.

A utilização de canais cifrados, associado ao formato específico SAML garante que a troca de dados segue as seguintes considerações:

a) **Privacidade de dados**

A utilização de canais cifrados garante que os dados do utilizador se mantêm privados, impedindo a sua visualização por terceiros (ex. visualização de dados por sniffer de rede);

b) **Integridade de dados**

O protocolo SAML, através de assinatura digital nos pedidos e respostas de autenticação, garante a integridade de dados de modificações não autorizadas (ex. ataque por *Man-in-the Middle*)[7].

3. **Par de chaves RSA**

O par de chaves RSA é utilizado para a assinatura do pedido SAML, através da chave privada do remetente e da chave pública imbutida no certificado X.509.

Esta assinatura permite verificar a validade do pedido de autenticação SAML recebido pelo Autenticação.Gov

4. **Certificado X.509**

Este certificado tem como função identificar a entidade responsável pelo pedido SAML. Como cada entidade possui o seu certificado único, esta é facilmente identificável.

Após esta breve explicação de todos os elementos necessários para a criação de um pedido de autenticação SAML, iremos explorar mais em detalhe cada um deles nas secções seguintes.

7 Para mais informações: https://pt.wikipedia.org/wiki/Ataque_man-in-the-middle

2.2.4.1 *SAML 2.0*

O SAML define um padrão standard para a troca de informação segura entre diversas entidades na Web.

Mais precisamente, o SAML define um framework XML para a troca de asserções de informação entre entidades.

> "...to define, enhance, and maintain a standard XML-based framework for creating and exchanging authentication and authorization information."
>
> – Security Assertion Markup Language V2.0 Technical Overview(9)

A versão 2.0 do SAML foi aprovada pela *Organization for the Advancement of Structured Information Standards (OASIS)* em 2005, modificando o standard de tal forma, que implementações baseadas na versão prévia (versão 1.1) eram incompatíveis com o novo standard.

Na especificação técnica do SAML existem dois *providers*:

- **Identity provider**

  Faculta a autenticação, verificando a identidade do utilizador, procedendo ao envio de informação para o *Service provider*, juntamente com os níveis de acesso do utilizador (Autenticação.Gov).

- **Service provider**

  Requisita a autenticação ao *Identity provider* de modo a proceder à autorização do utilizador (Plataforma CLAV).

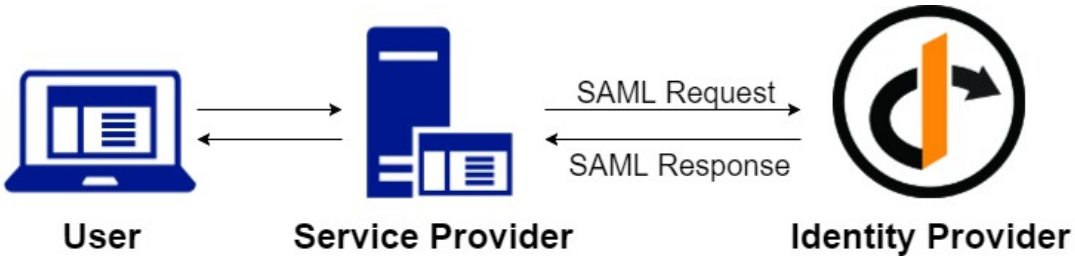


Figura 10: Interação entre o utilizador e os respetivos *Providers*. (4)

Devido à especificação do SAML, este permite a utilização de SSO, ou seja, após um único login do utilizador, essas mesmas credenciais podem ser reutilizadas para proceder ao *login* noutros *Service Providers*.

De modo a realizar comunicação entre os *providers* é utilizado um documento XML, conhecido por *SAML Assertion*. Existem 3 tipos de *Assertions*:

1. **Authentication**

   Fornece identificação do utilizador em questão, providenciando dados como a hora a que o login foi realizado e que tipo de autenticação foi utilizada (por exemplo: Cartão de Cidadão).

2. **Attribution**

   Transmite atributos SAML (um atributo SAML trata-se de um formato de dados responsável pela transmição de informação sobre o utilizador) para o *Service Provider*.

3. **Authorization**

   Reporta se o utilizador está autorizado a utilizar o serviço em questão, ou se este foi negado autorização pelo *Identity Provider*.

Em suma, o SAML funciona através do partilha de informação sobre utilizadores, login e atributos entre ambos os *Providers*.

Após ser realizado um login com o auxílio do *Identity Provider*, este pode enviar atributos SAML para o *Service Provider* quando o utilizador tenta aceder a certos serviços. Por ventura, o *Service Provider* requesita autorização e autenticação ao *Identity Provider*.

2.2.4.2 *SSL e HTTPS*

De modo a tornar a comunicação entre a plataforma CLAV e o Autenticação anónima, bem como diminuir a possibilidade de modificação de dados não autorizadas, é necessária a implementação de um mecanismo de cifragem.

Para tal, no manual de integração do Autenticação.Gov é recomendada a utilização de *SSL V3+* ou *TLS 1.0+*(6), tendo sido optada a implementação de *SSL V3+* na plataforma CLAV.

O protocolo SSL (Secure Sockets Layer), possui a simples tarefa de encriptar dados na Internet. Foi desenvolvido pela *Netscape* em 1995, com o intuito de garantir a privacidade, autenticação e integridade dos dados nas comunicações sobre a Internet. O SSL é o predecessor da encriptação TLS utilizada hoje em dia.

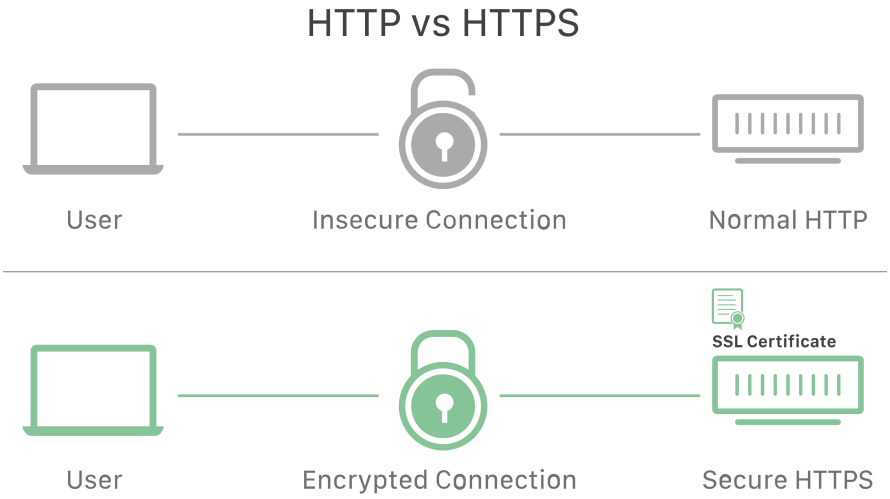


Figura 11: Figura representativa da diferença entre comunicação via HTTP e HTTPS.

Mas como é que o protocolo SSL funciona?

De modo a garantir um alto nível de privacidade, o SSL encripta todos os dados que são transmitidos pela Internet. Isto significa que de qualquer tentativa de intercepção destes dados irá resultar um conjunto de carateres aleatórios, sendo estes quase impossíveis de desencriptar.

Quando corre um acesso a uma página web configurada com protocolo SSL, este inicia um processo designado por "*handshake*"entre os dois dispositivos, Utilizador e Servidor, de modo a garantir e verificar a identidade de ambos.

Além deste processo, o SSL também realiza a assinature digital dos dados transmitidos, de modo a garantir a integridade dos mesmos, verificando que estes não foram de alguma forma indevida modificados antes de chegar ao destino.

Como é realizado o *handshake* entre cliente e servidor?
O processo de "SSL *handshake*"pode ser resumido a 8 simples passos:

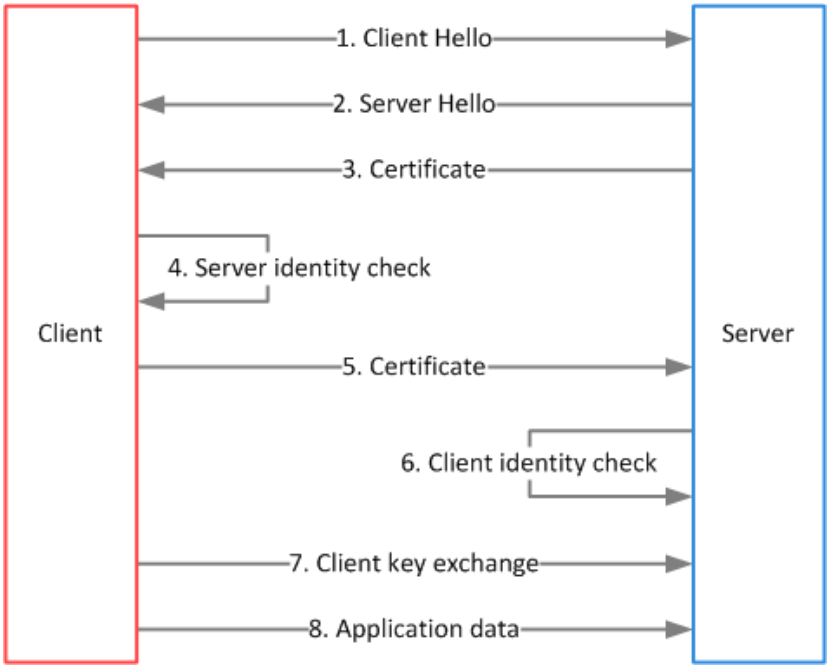


Figura 12: Resumo do processo de *handshake* entre cliente e servidor.

1. **Client Hello**

   Neste primeiro passo, o cliente envia ao servidor um *"ClientHello"* de modo a propor as seguintes opções para uma conexão SSL:

   a) Protocolo.

   b) Versão do protocolo.

   c) Cifra (Algoritmo de encriptação).

   d) Data específica relativa à sessão em causa.

   e) Outra informação que o servidor possa precisar.

2. **Server Hello**

   De seguida, o servidor envia para o cliente um *"ServerHello"* de modo a confirmar as opções da conexão SSL. É de referir que é neste passo que pode ocorrer a primeira negação da conexão SSL devido aos seguintes fatores:

   a) **Incompatibilidade de protocolo**

      Ocorre quando a versão do protocolo a utilizar proposta pelo cliente não é suportada pelo servidor, ou vice-versa.

b) **Incompatibilidade de cifra**

Ocorre quando a cifra proposta pelo cliente não é suportada pelo servidor, ou vice-versa.

3. **Envio do certificado pelo servidor**

   Neste passo ocorre o envio do certificado do servidor para posterior verificação pelo cliente.

4. **Verificação do certificado**

   Nesta fase ocorre a verificação da identidade do servidor por parte do cliente, através do certificado SSL do servidor. Caso o certificado do servidor seja inválido ou tenha expirado ocorre a negação da ligação SSL.

5. **Envio do certificado pelo cliente**

   Neste passo ocorre o envio do certificado do cliente para posterior verificação pelo servidor.

6. **Verificação do certificado**

   Nesta fase ocorre a verificação da identidade do cliente por parte do servidor, através do certificado SSL do cliente. Caso o certificado do cliente seja inválido ou tenha expirado ocorre a negação da ligação SSL.

7. **Troca de chaves de cliente**

   Caso o servidor confirme a identidade do cliente, o cliente irá gerar uma chave para a sessão em curso, sendo esta encriptada com a chave pública do certificado SSL do servidor e enviada para o servidor.

8. **Envio de dados**

   Após o envio da chave de sessão para o servidor, é iniciado o envio de dados para o mesmo. Cada pacote é encriptado utilizando a chave de sessão, sendo que após a recepção do mesmo pelo servidor, este é desencriptado pelo mesmo e por fim processado.

Com a utilização de *SSL V3* podemos garantir a anonimidade das comunicações entre a plataforma CLAV e o Autenticação.Gov, bem como a eliminação de qualquer tentativa de ataque baseada em "*Man-in-the-middle*", além de garantir a cifragem dos dados.

2.2.4.3 *Par de chaves públicas/privadas*

O Autenticação.Gov utiliza um mecanismo de autenticação de pedidos baseado em pares de chaves públicas e privadas. Para tal, é necessário gerar um par de chaves baseado no formato RSA, do qual resultam duas chaves:

- **Chave privada**

  Chave de conhecimento privado, ou seja, apenas o criador da chave possui informação sobre a mesma. Esta é composta por dois números primos aleatórios de comprimento substancial.

- **Chave pública**

  Chave de conhecimento público, disponível para qualquer utilizador, não possuindo qualquer tipo de informação sensível. Esta é o resultado da multiplicação dos dois números primeiros da chave privada, sendo quase impossível a obtenção da combinação original.

Na autenticação com o Cartão de Cidadão, é enviado um pedido para o Autenticação.Gov, sendo este assinado digitalmente com auxílio da chave privada. Como o Autenticação.Gov possui acesso à chave pública, é-lhe possível verificar instantaneamente a validade do pedido em causa, sendo este processado caso seja válido, ou negado caso contrário.

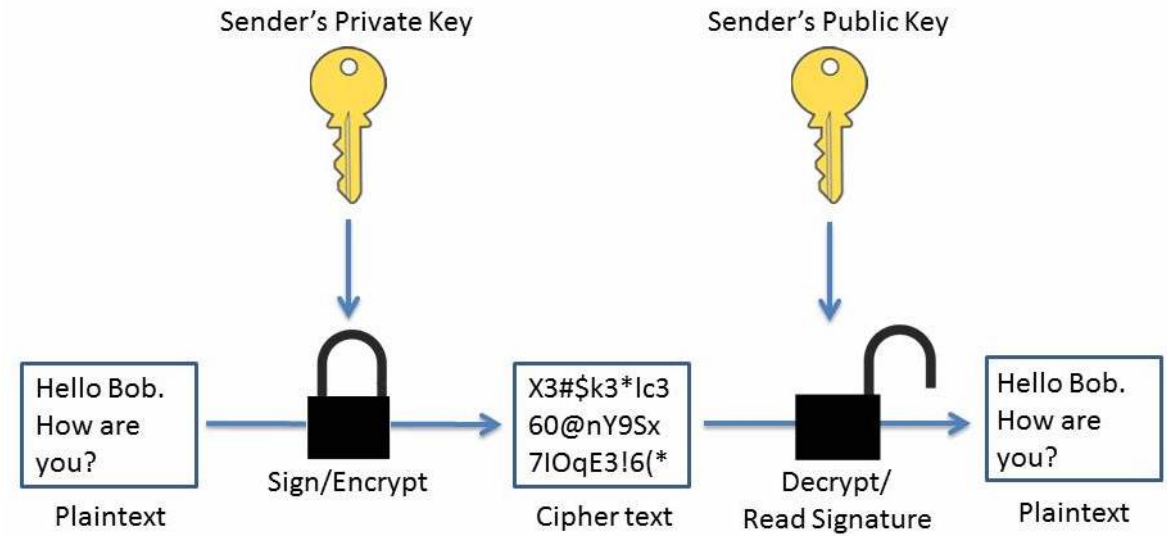


Figura 13: Exemplo da assinatura e leitura através de par de chaves.

Na figura 13 podemos verificar que o funcionamento da assinatura digital baseado em pares de chaves segue os seguintes passos:

1. É criado um texto/pedido de autenticação/etc.

2. Esse pedido previamente criado é assinado com a chave privada do remetente.

3. Após a recepção do mesmo pelo destinatário, é feita a leitura com o auxílio da chave pública do remetente presente no certificado X509 do mesmo.

Segue-se um exemplo de uma chave RSA privada, sendo esta de maior comprimento quando comparada com a chave pública, devido a conter dois números primos (como foi referido no início desta secção).

Exemplo de chave RSA privada:

```
-----BEGIN RSA PRIVATE KEY-----
MIICXAIBAAKBgQCqGKukO1De7zhZj6+H0qtjTkVxwTCpvKe4eCZ0FPqri0cb2JZfXJ/DgYSF6vUp
wmJG8wVQZKjeGcjDOL5UlsuusFncCzWBQ7RKNUSesmQRMSGkVb1/3j+skZ6UtW+5u09lHNsj6tQ5
1s1SPrCBkedbNf0Tp0GbMJDyR4e9T04ZZwIDAQABAoGAFijko56+qGyN8M0RVyaRAXz++xTqHBLh
3tx4VgMtrQ+WEgCjhoTwo23KMBAuJGSYnRmoBZM3lMfTKevIkAidPExvYCdm5dYq3XToLkkLv5L2
pIIVOFMDG+KESnAFV7l2c+cnzRMW0+b6f8mR1CJzZuxVLL6Q02fvLi55/mbSYxECQQDeAw6fiIQX
GukBI4eMZZt4nscy2o12KyYner3VpoeE+Np2q+Z3pvAMd/aNzQ/W9WaI+NRfcxUJrmfPwIGm63il
AkEAxCL5HQb2bQr4ByorcMWm/hEP2MZzROV73yF41hPsRC9m66KrheO9HPTJuo3/9s5p+sqGxOlF
L0NDt4SkosjgGwJAFklyR1uZ/wPJjj611cdBcztlPdqoxssQGnh85BzCj/u3WqBpE2vjvyyvyI5k
X6zk7S0ljKtt2jny2+00VsBerQJBAJGC1Mg5Oydo5NwD6BiROrPxGo2bpTbu/fhrT8ebHkTz2epl
U9VQQSQzY1oZMVX8i1m5WUTLPz2yLJIBQVdXqhMCQBGoiuSoSjafUhV7i1cEGpb88h5NBYZzWXGZ
37sJ5QsW+sJyoNde3xH8vdXhzU7eT82D6X/scw9RZz+/6rCJ4p0=
-----END RSA PRIVATE KEY-----
```

De seguida temos um exemplo de uma chave pública, de menor comprimento quando comparada com a chave privada (devido a ser o produto dos dois números primos da chave privada).

Exemplo de chave RSA pública:

```
-----BEGIN PUBLIC KEY-----
MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCqGKukO1De7zhZj6+H0qtjTkVxwTCpvKe4eCZ0
FPqri0cb2JZfXJ/DgYSF6vUpwmJG8wVQZKjeGcjDOL5UlsuusFncCzWBQ7RKNUSesmQRMSGkVb1/
3j+skZ6UtW+5u09lHNsj6tQ51s1SPrCBkedbNf0Tp0GbMJDyR4e9T04ZZwIDAQAB
-----END PUBLIC KEY-----
```

2.2.4.4 *Certificado X.509*

O Autenticação.Gov requer o fornecimento de um certificado no pedido SAML, de modo a validar as origens do pedido, bem como confirmar a integridade e validade do mesmo.

O formato deste certificado é designado por *X.509*(19), um padrão criado pela *International Telecommunication Union (ITU)* em 1988.

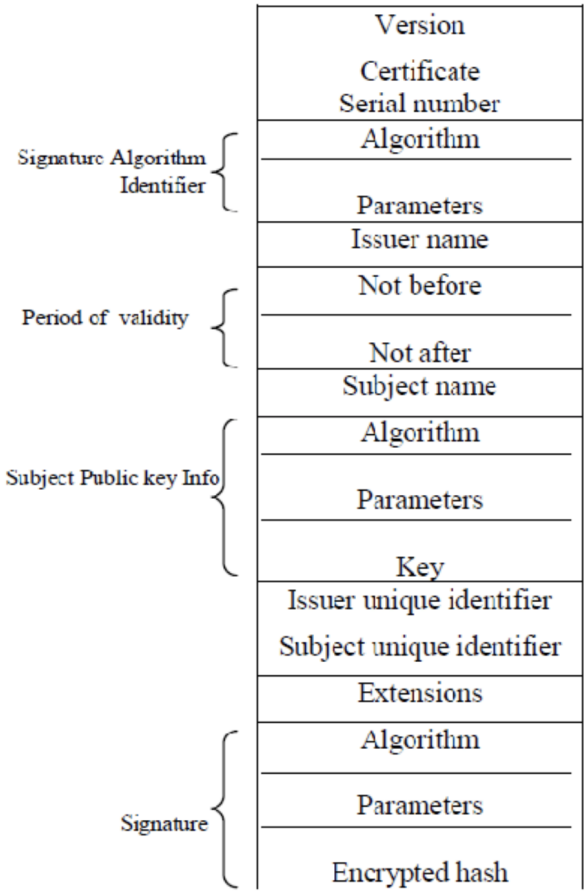


Figura 14: Formato de um certificado X.509 v3.

Este pode conter várias informações além da chave RSA pública (ver figura 14), sendo que ao abrigo do Autenticação.Gov, os certificados X.509 por ele utilizados contêm a seguinte informação:

- Chave pública e respetivo algoritmo.
- Nome da Entidade.
- Entidade emissora.

- Data de emissão.
- Validade do certificado.
- Informações de contacto.
- Assinatura e respetivo algoritmo.

Com o auxílio desta informação, é possível verificar e validar o certificado, como tendo sido emitido por uma entidade certificada, sendo este passo crucial para os pedidos de autenticação com o Cartão de Cidadão.

De modo a garantir a validade do certificado, as seguintes regras têm de ser obedecidas:

1. O certificado tem de possuir uma entidade emissora válida (neste caso, a plataforma CLAV é a entidade emissora certificada pela AMA).
2. O certificado tem de possuir uma data de validade válida e não pode estar expirado.
3. O certificado não pode estar presente na lista de certificados revogados.
4. Todos os certificados da sua cadeia de certificação têm de ser válidos.

Um objeto contendo as caraterísticas previamente descritas pode ser representado através do padrão *X.509*, sendo este utilizado como suporte para a PKI (Public Key Infrastucture).

A estrutura definida pela PKI é utilizada de forma universal para a autenticação de utilizadores através de certificados de chave pública.(5)

A utilização destes certificados, auxiliados pela utilização de um canal de comunicação seguro via *HTTPS* e *SSL* (explorado em detalhe na secção 2.2.4.2) garantem a autenticidade e confidencialidade necessárias à utilização de serviços sensíveis, como por exemplo, a autenticação com o auxílio do Cartão de Cidadão.

Exemplo de Certificado X.509:

```
subject=/emailAddress=it@ama.pt/C=PT/L=Lisboa/O=AMA/OU=IT/CN=Autenticacao.Gov
issuer=/emailAddress=it@ama.pt/C=PT/L=Lisboa/O=AMA/OU=IT/CN=Autenticacao.Gov
-----BEGIN CERTIFICATE-----
MIIDzDCCArSgAwIBAgIGAWHCrFAKMA0GCSqGSIb3DQEBCwUAMHMxGDAWBgkqhkiG
9w0BCQETCWl0QGFtYS5wdDELMAkGA1UEBhMCUFQxDzANBgNVBAcTBkxpc2JvYTEM
```

MAoGA1UEChMDQU1BMQswCQYDVQQLEwJJVDEeMBwGA1UEAxMVQXV0ZW50aWNhY2Fv
LkdvdiBSYWl6MB4XDTE4MDIwNjEyMzg0OFoXDTI5MDIwNTEyMzg0OFowczEYMBYG
CSqGSIb3DQEJARMJaXRAYW1hLnB0MQswCQYDVQQGEwJQVDEPMA0GA1UEBxMGTGlz
Ym9hMQwwCgYDVQQKEwNBTUExCzAJBgNVBAsTAklUMR4wHAYDVQQDExVBdXRlbnRp
Y2FjYW8uR292IFjBkMB8GA1UdIwQYMBaAFLLBjPTQe0ItzZ3Cda8123axE-23971
iHk+GwTe8PDpMB0GA1UdDgQWBBSywYz00HtCLc2dwoh5PhsE3vDw6TASBgNVHRMB
Af8ECDAGAQH/AgEBMA4GA1UdDwEB/wQEAwICBDANBgkqhkiG9w0BAQsFAAOCAQEA
PIRDKpYrNQRd2/5/+Pm/xN1KXbk3uZUNPsL0T2Hz/RJoX7X3gg0+d1i9cem3izNP
hQXuh5GO4z+hTIBDfnpvPnjVbYZTZaApkO/GmdC6VpXBVfR4H8A7z1CzOQ52F81n
WHYfgpUXQfyCz1KPNw7ba4LYdiTaAg8Yyd6BmGb0ud5Pq235AzpbvJs9I7HniGMD
eVLH6asnt4RUkaJrIelrqSHC1ZJNXHA75LlQHkaaxG8mWnRmb4ikmaR923hca6oQ
1JyGrOvrVLTVKdNgy1wKSyA47HZzHLMto5EBxj/j5xKJ040MNN2N1Isg+kBEqkt+
pAR+Z3kctgfUVHton0cwJQ==
-----END CERTIFICATE-----

2.2.5 *Incongruências no Autenticação.Gov*

Durante o desenvolvimento desta dissertação foram encontradas algumas incongruências na especificação do Autenticação.Gov:

1. **Assinatura no pedido *AuthnRequest***

   Na página 48 do manual de integração do Autenticação.Gov(6), é referido que a assinatura do pedido de autenticação deverá usar o algoritmo *SHA-1* como digest. A utilização de tal algoritmo é desaconselhada, visto este ser considerado como *deprecrated*[8]. Este apenas deve ser utilizado para verificação de *time-stamps*, geração de *Hash-based Message Authentication Code (HMAC)* entre outros usos extremamente limitados.

   É importante referir que qualquer tentativa de utilizar outra função para digest, tal como *SHA-256*, *SHA-512*, entre outras, resulta na criação de um pedido SAML correto, mas rejeitado pelo Autenticação.Gov. Podemos assim concluir que a atual implementação do Autenticação.Gov apenas permite a utilização de *SHA-1*.

   Para tal, é recomendada a mudança imediata do algoritmo utilizado para o digest para algo mais seguro, tal como as funções da família *SHA-2*, por exemplo, *SHA-256*.

8 Para mais informações aceder ao URL seguinte: https://csrc.nist.gov/Projects/Hash-Functions/NIST-Policy-on-Hash-Functions

2. **Certificados para encriptação do tráfego**

   Na página 52 do manual de integração do Autenticação.Gov(6), é referido que as comunicações entre o utilizador e o Autenticação.Gov deverão ser cifradas através da utilização de *SSLv3* (ou superior), ou *TLS 1.0* (ou superior).

   Como acontece com o algoritmo de encriptação, a utilização de *SSLv3* é desaconselhada, visto este ser considerado como *deprecated*[9]. É importante referir que nesta dissertação foi utilizado um certificado *SSLv3* devido à facilidade de obtenção do mesmo, sendo que a alteração para *TLS* pode ser feita através da obtenção de um novo certificado, não sendo necessária qualquer reestruturação da solução já implementada.

   Em suma, é recomendado a criação de um período de transição, no qual os utilizadores do Autenticação.Gov possam alterar o certificado utilizado para uma vertente do *TLS*, como por exemplo, o *TLS 1.3*[10].

3. **Definição de ID nos pedidos**

   Cada pedido de autenticação *AuthnRequest* possui um identificador único, designaod por *ID*. Este campo é gerado através do *UUID/v4*, sendo sempre gerados identificadores totalmente aleatórios.

   Por norma, estes identificadores gerados podem começar com qualquer letra ou número, sendo que durante o desenvolvimento da autenticação com o Cartão de Cidadão, apenas um número pequeno de pedidos de autenticação eram de facto gerados com sucesso. Isto devia-se ao caso de o *ID* ter como primeiro caractere uma letra.

   Em suma, apenas gerava um pedido correto quando era iniciado por um número. A solução proposta pela AMA foi preceder todos os identificadores de um "_", tornando todos os identificadores gerados válidos.

   É importante referir que no manual de integração esta solução não está referenciada em nenhum local, sendo aconselhada acrescentar esta mesma informação na próxima versão do mesmo.

Além das incongruências e vulnerabilidades descritas previamente, é possível encontrar exemplos desatualizados, como por exemplo, pedidos de autenticação. Para tal, é aconselhada uma revisão ao documento, sendo que nesta se devem refletir os pontos previamente descritos, de modo a garantir uma maior segurança e fiabilidade da plataforma Autenticação.Gov.

9 Para mais informações aceder ao URL seguinte: https://tools.ietf.org/html/rfc7568
10 Para mais informações aceder ao URL seguinte: https://tools.ietf.org/html/rfc8446

2.3 CARTÃO DE CIDADÃO

O Cartão de Cidadão (CC) é um documento de identificação da cidadania portuguesa, desenvolvido durante o XVII Governo Constitucional de Portugal[11], tendo a primeira emissão do mesmo ocorrido em 2006/2007 numa fase experimental na Região Autónoma dos Açores, seguido da emissão a nível nacional em 2008.

Figura 15: Exemplo de um Cartão de Cidadão(1).

O objetivo do Cartão de Cidadão não se cinge a uma simples substituição *Bilhete de Identidade (BI)*, mas sim à agregação de vários documentos pessoais do cidadão, nomeadamente:

- Cartão de contribuinte.
- Cartão de eleitor.
- Cartão de beneficiário da Segurança Social.
- Cartão de utente do Serviço Nacional de Saúde.

O Cartão de Cidadão foi desenvolvido com as várias normas internacionais em mente, para que este seja reconhecido como um documento de identificação e de viagem.

Dito isto, é importante referir que o Cartão de Cidadão é mais do que um simples documento. Este foi concebido para ser um documento inteligente, ou seja, um *smartcard*, sendo dotado de um chip embutido na sua construção plástica.

A invenção do *smartcard* pode ser atribuida a Helmut Gröttrup e Jürgen Dethloff[12] em 1968 pela criação de um chip capaz de caber num bolso, sendo adaptado para um cartão plástico por Michel Ugon em 1977.

11 Mais informação em https://pt.wikipedia.org/wiki/Cart~ao_de_cidad~ao.
12 Mais informação em http://www.historyofinformation.com/detail.php?id=2317

Desde 1977 ocorreu uma evolução deste tipo de cartões, podendo os mesmos ser divididos em quatro categorias:

- Cartões com microprocessador.
- Cartões de memória.
- Cartões óticos.
- Cartões de banda magnética.

O Cartão de Cidadão está inserido na primeira categoria, Cartões com microprocessador, sendo alvo de uma explicação mais detalhada na secção 2.3.1.

Sendo utilizado como documento físico, este permite a um cidadão identificar-se presencialmente de forma segura. Como documento tecnológico, permite ao seu portador identificar-se perante serviços informatizados e autenticar documentos eletrónicos.

Outro propósito da criação do Cartão de Cidadão foi melhorar o nível de segurança associado aos cartões de identificação, de forma a diminuir e evitar falsificações dos mesmos.

O chip presente no Cartão de Cidadão permite guardar dados pessoais do titular, garantindo a privacidade e segurança dos mesmos. Por exemplo, impede acessos indevidos a dados médicos do titular por parte de trabalhadoras da Segurança Social, visto cada entidade só ter acesso aos dados que lhe dizem respeito, de modo a evitar o uso indevido dos dados do titular, bem como uso indevido de poder.

2.3.1 *Especificação do Cartão de Cidadão*

O Cartão de Cidadão segue o padrão designado por *ID-1*[13], sendo este baseado na norma *ISO/IEC 7810* que especifica as dimensões do Cartão de Cidadão, sendo estas de 8,56cm de comprimento e 5,398cm de largura. Além da norma previamente descrita, o Cartão de Cidadão implementa a norma *ISO 7816-2*[14] que define a localização e funcionamento do chip do cartão.

O chip presente no Cartão de Cidadão tem de obedecer a inúmeras especificações, sendo que para tal possui as seguintes caraterísticas:

1. Possui uma memória *Electrically-Erasable Programmable Read-Only Memory (EEPROM)* com 64Kb de memória ou superior.

13 Mais informações em https://www.iso.org/standard/31432.html
14 Mais informações em https://www.iso.org/standard/45989.html

2. É baseado na plataforma *Java Card*, sendo portanto multi-aplicacional.
3. Suporta a versão mais recente da plataforma *Java Card*.
4. Possui mecanismos capazes de realizar a gestão dinâmica da sua memória, através de "garbage collection"da *Java Virtual Machine (JVM)*, bem como proteção da mesma.
5. Possui mecanismos capazes de gerir o espaço de armazenamento através de desfragmentação.
6. Possui mecanismos capazes de gerar números "verdadeiramente"aleatórios.
7. Suporta vários códigos PIN baseados na norma *ISO/IEC 7816-4*[15]:
   a) PIN da autenticação.
   b) PIN da morada.
   c) PIN da assinatura digital.
8. Possui mecanismos de bloqueio caso erre na escrita do código PIN.
9. Possui mecanismos de desbloqueio através de código *Pin Unlock Key (PUK)*.
10. Possui mecanismos de geração de novo código PIN.
11. Possui criptografia interna com as seguintes caraterísticas:
    a) Assinatura e verificação baseadas em RSA de 1024bits.
    b) Assinatura eletrónica baseada na norma *CWA-14169*[16].
    c) Suporte para *Message-Digest algorithm 5 (md5)*, SHA-1 e SHA-256.
    d) Suporte para *Data Encryption Standard (DES)* e *Triple Data Encryption Standard (TDES)*[17].
    e) Suporte para *Message Authentication Code (MAC)*.
    f) Suporte para várias normas *Public-Key Cryptography Standards (PKCS)*, nomedamente *PKCS#1* e*PKCS#15*.
12. Suporte para leitores de cartões baseados na norma *Chip Authentication Program (CAP)*.
13. Capacidade de defesa contra ataques a nível de hardware.

15 Mais informações em https://www.iso.org/standard/54550.html
16 Mais informações em ftp://ftp.cen.eu/CEN/Sectors/TCandWorkshops/Workshops/eSIGN_CWAs/cwa14169-00-2004-Mar.pdf
17 Mais informações em https://en.wikipedia.org/wiki/Triple_DES

2.4 API DE DADOS

No âmbito do projeto CLAV é disponibilizada uma API pública de modo a serem feitas *queries* à base de dados.

De modo a estas chamadas, bem como controlar uso indevido da mesmas, foi necessário implementar autenticação e controlo através de *JSON Web Token (JWT)*[18].

A essência do JWT assenta no facto do token ser ou não válido, ou seja, se este já expirou ou foi revogado. A implementação deste token foi feita de forma a ser possível monitorizar quem está a fazer chamadas à API, podendo a qualquer altura revogar autorização para tal, bem como a auto-revogação após 30 dias.

Estes cuidados ajudam a evitar problemas, tais como ataques à API através de sucessivas chamadas num curto espaço de tempo, o que provoca uma sobrecarga no sistema.

Um token JWT pode ser definido como um *standard RFC 7519*(10)(16)[19] desenvolvido para a troca de pedidos em ambientes cujo espaço é limitado.

2.4.1 *Funcionamento do JSON Web Token*

Um *JSON Web token* é composto por 3 elementos distintos (quebras de linha inseridas para melhorar a leitura), separados por um ponto (.) entre cada elemento:

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6Ik
9jdMOhdmlvIE1haWEiLCJudW1BbHVubyI6NzEzNjI9.
SneQiuAGUW9aTpxlNNbMkEoYNj7v4-Sw_5jl134-hosk

Embora seja impossível retirar qualquer tipo de informação visível do *token* exemplificado anteriormente, este contem dados extremamente úteis, tendo sido gerado através da seguinte forma:

- Header (representado a vermelho).

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

18 Mais informação em https://jwt.io
19 Mais informação em https://tools.ietf.org/html/rfc7519

Todos os *JSON Web Token* possuem um header. Este elemento estabelece qual o algoritmo utilizado, se o JWT foi assinado ou encriptado e por norma, como proceder ao parse do resto do JWT.

O único atributo obrigatório num JWT é o *alg*, sendo que em JWT que não foram encriptados este valor é **none**, sendo que neste exemplo foi utilizada a encriptação **HS256**, ou seja, HMAC com o auxílio de *SHA-256*.

Existem tambem atributos opcionais, tais como *typ*, cuja função é distinguir entre o JWT e outros tipos de dados que possam ser transmitidos no mesmo formato[20], sendo que neste exemplo é transmitido um *JWT* e *cty*, que apenas é utilizado quando existem aninhamento de JWT dentro de outro(s) JWT.

Após o uso de *Base64URL enconding* sobre o algoritmo **HS256**, a informação acima representada é transcrita sobre a forma da seguinte string:

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9

- Payload (representado a verde).

```
{
  "sub": "1234567890",
  "name": "Octavio Maia",
  "numAluno": 71369
}
```

O payload é o elemento responsável pelo armazenamento de todo e qualquer dado do utilizador. Como o header, este elemento segue o formato *JSON*, embora que todos os atributos presentes no payload sejam de natureza opcional, ao contrário do header que possui pelo menos um atributo de cariz obrigatório.

De acordo com a especificação do payload, existem 7 atributos registados, sendo que neste exemplo apenas existe um, designado por *sub*. Este atributo contem informação sobre quem se trata a informação contida neste JWT, sendo que para tal efeito, este atributo é de cariz único na aplicação em questão.

Os atributos designados por *name* e *numAluno* são designado por atributos não registados, podendo conter qualquer tipo de informação representável sobre a forma de string.

No âmbito do projeto CLAV são utilizados outros atributos registados, sendo o de maior importância o atributo *exp*, cuja função é representar uma data

20 Tipos de dados transmissíveis: http://www.iana.org/assignments/media-types/media-types.xhtml

específica, seguindo o formato *"seconds since epoch"*[21], após a qual o JWT é considerado inválido.

A informação acima representada anteriormente é transcrita sobre a forma da seguinte string:

eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6Ik
9jdMOhdmlvIE1haWEiLCJudW1BbHVubyI6NzEzNjl9

- Assinatura (representado a azul).

```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  secretKey
)
```

Embora a definição de JWT apenas abranja os dois elementos previamente estudados, o terceiro elemento tem como função garantir que o JWT não foi indevidamente alterado.

Por norma é utilizado uma variante do algoritmo *Base64*, designada por *Base64-URL*. Para encriptar a assinatura, são concatenadas 3 strings que foram sujeitas ao codificação via *Base64-URL*.

```
[Base64-URL header].[Base64-URL payload].[secretKey]
```

O resultado desta concatenação e posterior encriptação via **HS256** é a seguinte string:

SneQiuAGUW9aTpxlNNbMkEoYNj7v4-Sw_5jl134-hosk

2.5 Síntese

Neste capítulo foi explorado o estado da arte atual, sendo o foco principal a vertente da autenticação da plataforma CLAV, sendo o foco a autenticação através de Cartão de Cidadão utilizando o Autenticação.Gov, bem como a autenticação local através de username e password e a gestão dos utilizadores.

No próximo capítulo será descrita a solução proposta para os problemas apresentados neste capítulo, através de diagramas de fluxo e especificação técnica da mesma.

21 Definido pela *Portable Operating System Interface (POSIX)* em http://pubs.opengroup.org/onlinepubs/9699919799/basedefs/V1_chap04.html#tag_04_15

3

SOLUÇÃO

Neste capítulo iremos explorar a solução encontrada para os vários desafios e necessidades encontradas ao longo desta dissertação.

3.1 AUTENTICAÇÃO LOCAL

3.1.1 *Especificação*

De modo a satisfazer as necessidades da autenticação local de utilizadores, através de username e password, foi decidida a implementação de uma base de dados em *MongoDB*.

A criação de um utilizador na plataforma CLAV obriga ao preenchimento dos seguintes dados:

1. **Nome**

   Qualquer combinação de números e letras.

2. **Email**

   Considera-se um email válido qualquer combinação de letras e números, seguida de um "@"com terminação em "."seguido de letras.

3. **Nível de utilizador**

   Durante a fase de registo é obrigatória a seleção do nível de utilizador do mesmo, sendo esta utilizada para verificação de nível de acesso dentro da plataforma CLAV.

   a) Administrador de Perfil Tecnológico

   b) Administrador de Perfil Funcional

   c) Utilizador Validador

   d) Utilizador Avançado

   e) Utilizador Decisor (AD)

f) Utilizador Decisor

g) Utilizador Simples

h) Representante Entidade

4. **Entidade**

   A escolha de uma entidade ao qual o utilizador pertence é de caratér obrigatório, não sendo permitido o registo sem a mesma ser selecionada. Para tal efeito, é carregada a lista de entidades já registadas no sistema das quais o utilizador escolherá uma.

5. **Password**

   Qualquer combinação de números e letras.

Além da obrigatoriedade dos dados prévios, foi também decidido que todos os dados relativos aos utilizadores iriam ser guardados numa coleção MongoDB com o nome "*users*".

3.1.2 *Workflow da autenticação*

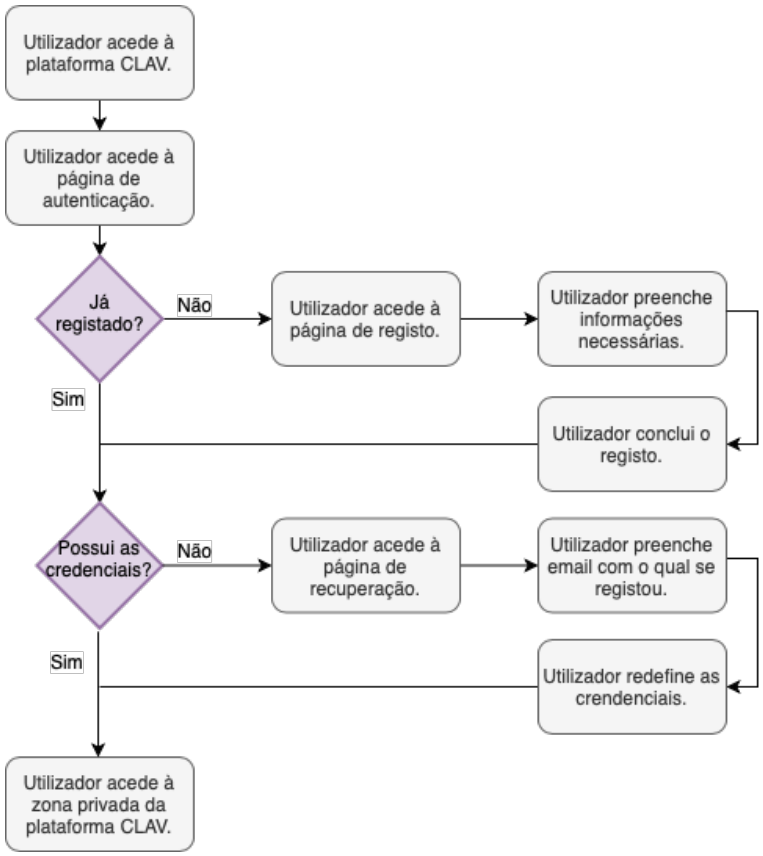


Figura 16: Workflow da autenticação local.

3.2 AUTENTICAÇÃO ATRAVÉS DE CARTÃO DE CIDADÃO

3.2.1 *Especificação*

A autenticação através de Cartão de Cidadão é baseada na autenticação local, havendo distinção entre os utilizadores registados via username e password, e aqueles registados através de Cartão de Cidadão.

1. **Nome**

   Qualquer combinação de números e letras.

2. **Email**

   Considera-se um email válido qualquer combinação de letras e números, seguida de um "@"com terminação em "."seguido de letras.

3. **Nível de utilizador**

   Durante a fase de registo é obrigatória a seleção do nível de utilizador do mesmo, sendo esta utilizada para verificação de nível de acesso dentro da plataforma CLAV.

   a) Administrador de Perfil Tecnológico
   b) Administrador de Perfil Funcional
   c) Utilizador Validador
   d) Utilizador Avançado
   e) Utilizador Decisor (AD)
   f) Utilizador Decisor
   g) Utilizador Simples
   h) Representante Entidade

4. **Entidade**

   A escolha de uma entidade ao qual o utilizador pertence é de caratér obrigatório, não sendo permitido o registo sem a mesma ser selecionada.

3.2.2 *Workflow da autenticação*

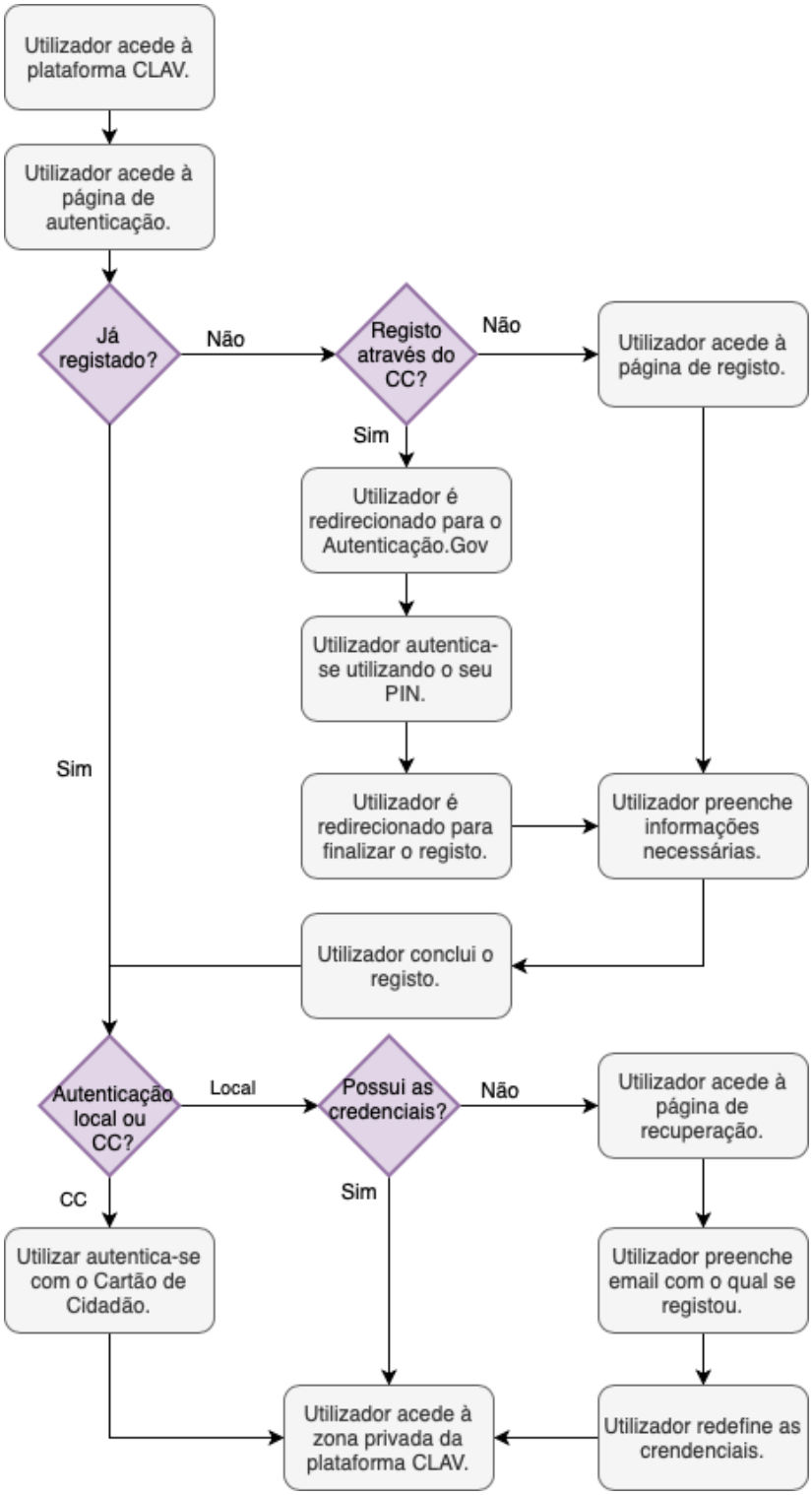


Figura 17: Workflow da autenticação através de Cartão de Cidadão.

3.3 PROTEÇÃO DA API DE DADOS

3.3.1 *Especificação*

De modo a fornecer uma camada de proteção à API de dados, é necessário identificar e autenticar os pedidos que a ela chegam.

Para tal, a solução encontrada foi a implementação de duas funções de middleware responsáveis pela seguinte função:

1. **Verificação da origem do pedido**

   Função de middleware cujo funcionamento é baseado em tokens JWT. Esta função é executada sempre que há um pedido à API e verifica se o pedido obedece aos seguintes critérios:

   a) **Presença do token**

      Todo e qualquer pedido à API de dados da plataforma CLAV, seja um *GET*, *POST*, entre outros, tem de ser enviado com um token baseado em JWT.

   b) **Validade do token**

      Após receção do token, este irá ser descodificado e irá ocorrer a verificação da validade do mesmo, através da utilização da assinatura, que irá verificar se o mesmo não foi comprometido.

2. **Verificação do nível de acesso**

   Caso o token seja válido e de facto contem o identificador do utilizador, é verificado o nível de acesso do mesmo. Caso este esteja a tentar aceder a funções da API que necessitam de um nível superior ao fornecido, o pedido é negado, caso contrário é permitido o acesso à informação pedida.

3.3.2 *Workflow da autenticação*

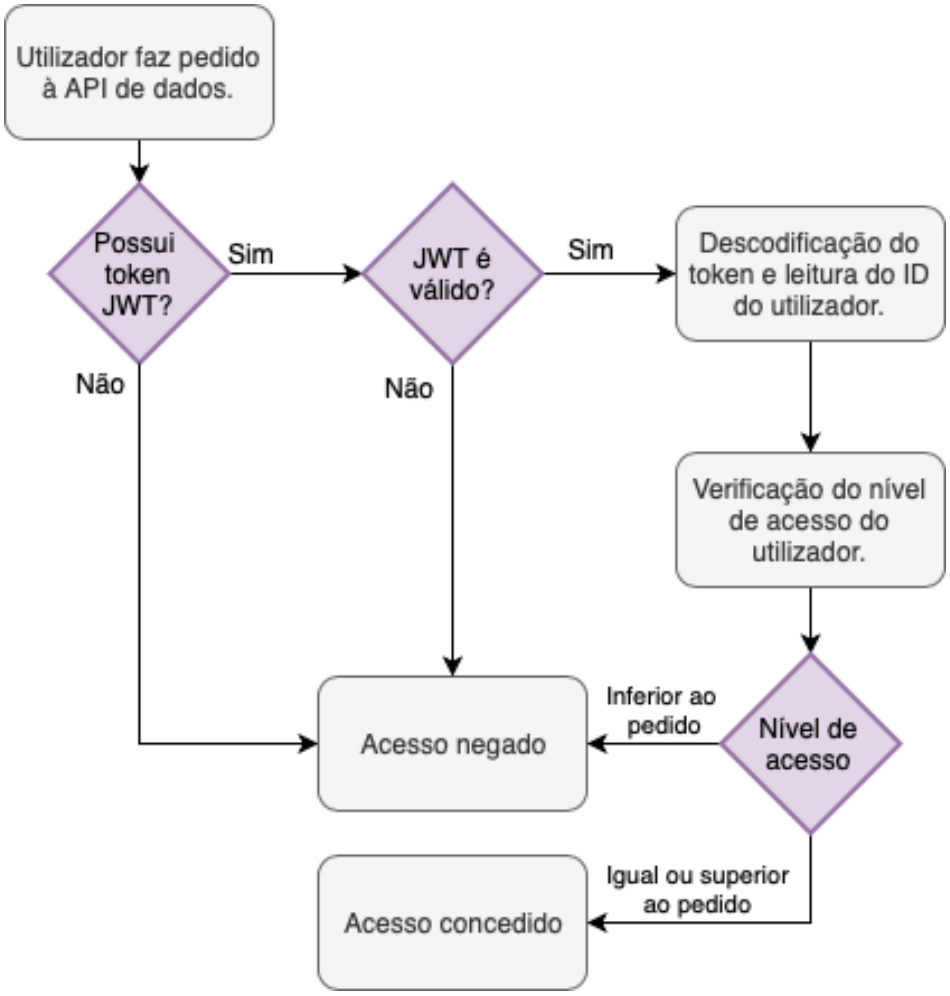


Figura 18: Workflow da autenticação de pedidos à API de dados.

3.4 GESTÃO DE UTILIZADORES

3.4.1 *Especificação*

De modo a fornecer uma solução para a gestão de utilizadores, foi sugerida a criação e implementação de uma página que correspondesse a um painel de gestão de utilizadores.

Este painel de gestão terá de suportar as seguintes funcionalidades:

1. Listagem de todos os utilizadores registados na plataforma CLAV.
2. Filtragem de utilizadores através de campo de pesquisa.
3. Ordenação de utilizadores através dos seguintes parâmetros:
   a) Nome.
   b) Entidade.
   c) Email.
   d) Nível de utilizador.

Além das funcionalidades previamente descritas, este painel de gestão terá de permitir a edição dos utilizadores em causa, permitindo alterar os parâmetros previamente listados, bem como suportar a desactivação de um utilizador, sendo a este interdito o acesso à plataforma CLAV enquanto não for desbloqueado.

3.4.2 *Workflow da edição de um utilizador*

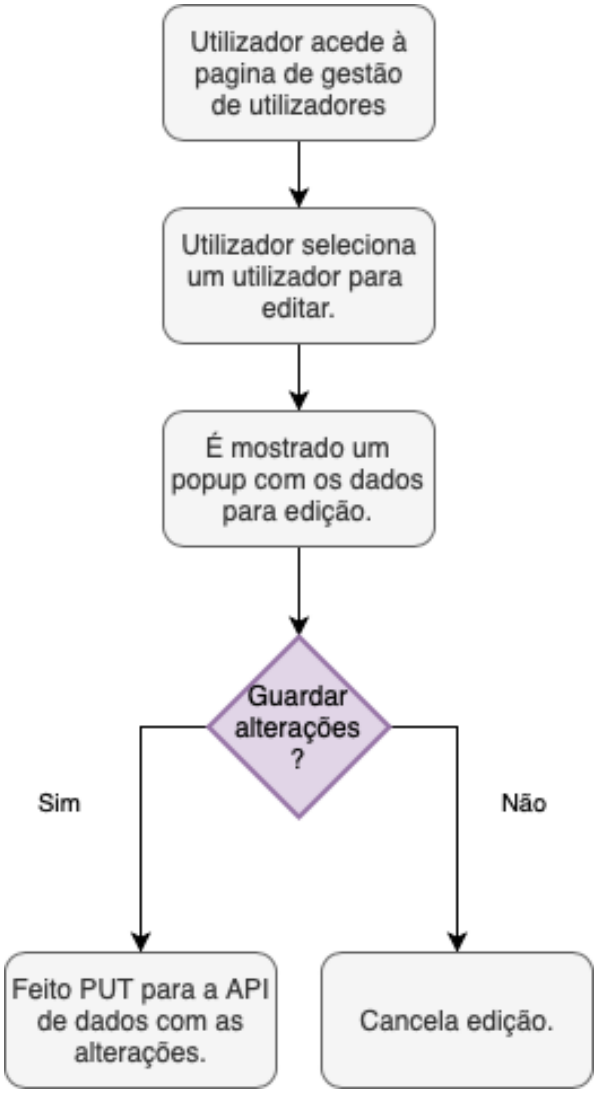


Figura 19: Workflow da edição de um utilizador na plataforma CLAV.

3.5 MÉTRICA DA PLATAFORMA

3.5.1 *Especificação*

Além das medidas especificadas na secção 3.3, foi sugerida a criação de uma página capaz de identificar a quantidade de pedidos processados pela plataforma CLAV. Para tal, é necessário distinguir entre os possíveis tipo de métodos HTTP existentes na API de dados:

- **GET**.
- **POST**.
- **PUT**.
- **DELETE**.

Além de ser guardado em base de dados o tipo de pedido feito, irá ser guardado a API de qual o pedido originou, quer seja ela a API dos utilizadores, das entidades, classes, etc, de forma a ser possível visualizar graficamente qual é a API mais utilizada.

Este recolher de dados permite também evitar utilizações indevidas da API de dados pública. Por exemplo, um elevado acesso à API pode levar a uma resposta extremamente demorada por parte do servidor, bem como o "crash"total da aplicação devido a esgotamento de recursos. Estes ataques são extremamente comuns e devido à implementação de métrica, é possível prevenir os mesmos de modo a eliminar esse risco.

3.5.2 *Workflow da incrementação da métrica na plataforma*

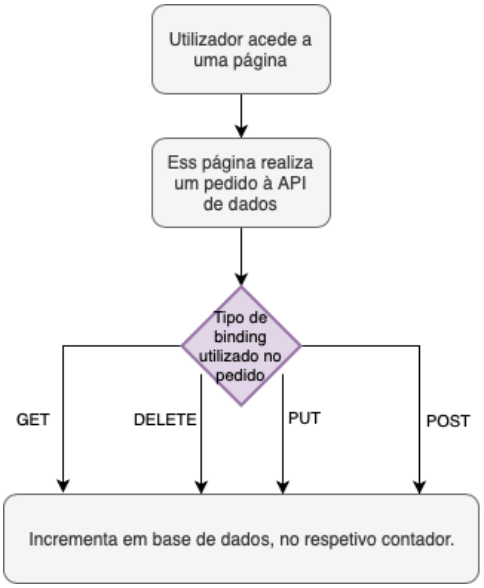


Figura 20: Workflow da incrementação de métrica na plataforma CLAV.

3.6 SÍNTESE

Uma das principais necessidades desta dissertação recaiu na documentação de todo o processo de desenvolvimento.

Esta documentação tem como objetivo explicar de forma explícita e com alto detalhe toda a arquitetura aplicacional implementada no projeto CLAV, sendo que para tal foram utilizados diagramas de fluxo que explicam de forma clara esta mesma solução.

O principal foco deste capítulo baseou-se na autenticação de utilizadores, quer por autenticação local, bem como por Cartão de Cidadão através do Autenticação.Gov e da protecção da API de dados disponibilizada ao público.

No seguinte capítulo irá ser descrita a implementação utilizada nas vertentes da plataforma CLAV.

4

IMPLEMENTAÇÃO

Neste capítulo iremos discutir a implementação das várias soluções descritas no capítulo 3. Para tal há que fazer a distinção entre as duas vertentes da plataforma CLAV:

- **Frontend**

  O frontend da plataforma CLAV é baseado na linguagem *JavaScript*, sendo utilizado o framework *Vue* para a criação das páginas e interfaces responsáveis pela interação com o utilizador da plataforma, com o auxílio do *Vuetify*.

- **Backend/API de dados**

  O backend da plataforma CLAV foi desenvolvido em *Node.JS*, utilizando uma API REST para comunicação com o frontend. Este utiliza duas base de dados, cada uma com funções distintas na plataforma:

  - **MongoDB**

    A base de dados não relacional, implementada em *MongoDB* é responsável pelo armazenamento de informação relativa aos utilizadores, chaves API, métricas entre outros.

  - **GraphDB**

    A base de dados baseada em grafos, implementada em *GraphDB* é responsável pelo armazenamento de informação relativa às entidades, legislações, bem como a Lista Consolidada. Devido a se usar uma base de dados baseada em grafos, esta permite cruzar informação de forma extremamente rápida e com pouco impacto no tempo de execução das queries, permitindo assim uma solução flexível, capaz de responder a pedidos específicos de forma rápida.

Esta decisão de separar o frontend e backend cinge o primeiro a um papel de simples interface com o utilizador, devoluto de qualquer ”lógica”aplicacional, ou seja, incapaz de realizar alterações em base de dados ou realizar pedidos à API.

Para obter tais informações, tem de comunicar com o backend, realizando pedidos conforme necessário, processando a resposta dada de modo a interagir com o utilizador.

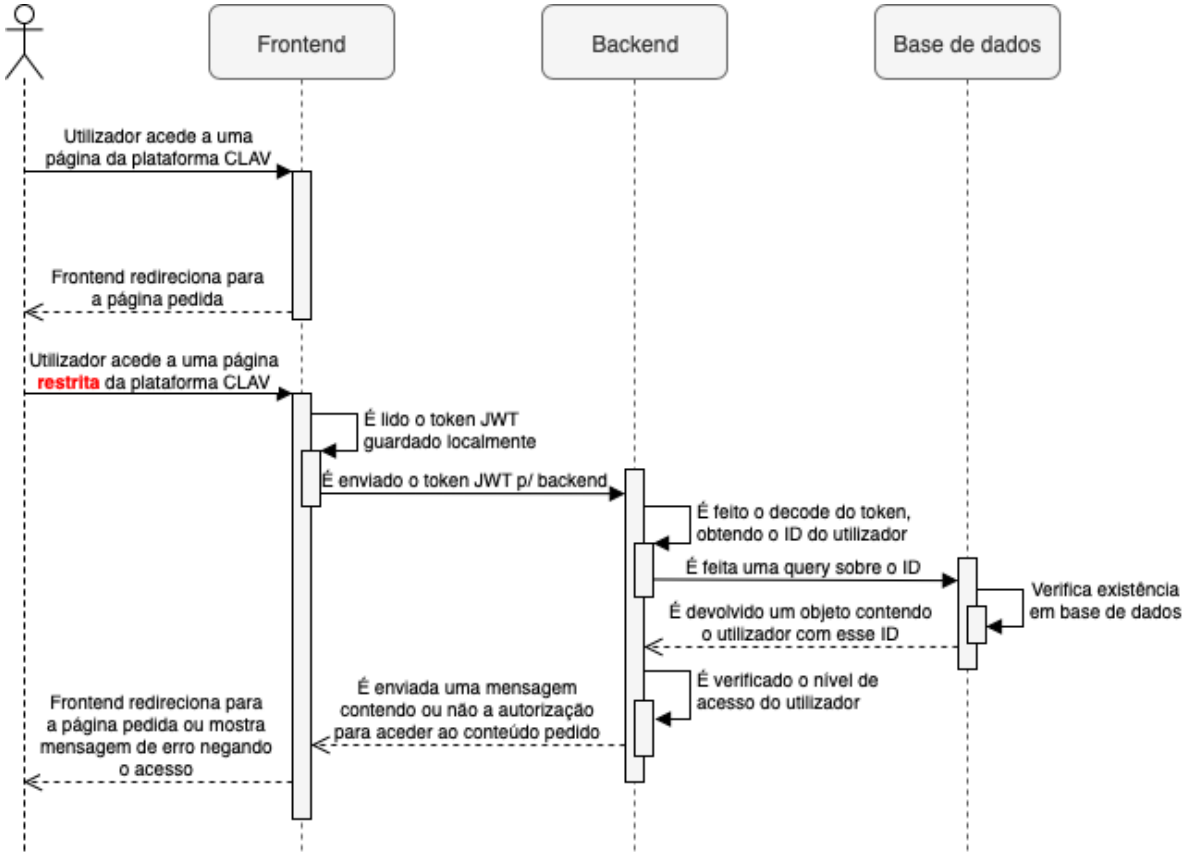


Figura 21: Diagrama de sequência ilustrativo da comunicação entre frontend e backend na plataforma CLAV.

Como podemos ver no diagrama de sequência da figura 21, o frontend serve apenas para interação com o utilizador, sendo que grande porção da lógica aplicacional ocorre no backend, que após terminar a query ou pedido em questão, retorna para o frontend mensagens de sucesso, erro, etc.

Toda e qualquer tarefa, quer seja de registo, login, acesso a informação sobre uma entidade, etc, é realizada sobre uma API desenvolvida especificamente para o frontend e backend comunicarem entre si de forma independente e totalmente modular, permitindo correr em servidores diferentes, sendo para tal utilizado o mecanismo *Cross-Origin Resource Sharing (CORS)*.

4.1 ENCRIPTAÇÃO DE INFORMAÇÃO SENSÍVEL

De modo a solucionar os problemas referidos na secção 2.1.1, foi necessário proceder ao desenvolvimento de uma função criptográfica que não só solucionasse o problema relacionado com ataques via *rainbow tables*, mas também fosse capaz de acompanhar os avanços tecnológicos a nível computacional.

Para tal, Niels Provos e David Mazières desenvolveram a função de *hashing* designada por **bcrypt**(17). Esta foi apresentada em 1999 na *The Advanced Computing Systems Association (USENIX)*, sendo uma função baseada na cifra *Blowfish*[1], desenvolvia por Bruce Schneier e apresentada em 1993.

Além de incorporar um *salt* para proteger contra ataques via *rainbow tables*, esta foi desenvolvida para ser uma função adaptável, sendo que o número de iterações pode ser incrementado de modo a aumentar o número de ciclos máquina necessários para o cálculo do *hash*.

Algorithm 1 Pseudo código do algoritmo *bcrypt*.

```
1: function BCRYPT(cost, salt, pwd)
2:     state ← EksBlowfishSetup(cost, salt, key)
3:     ctext ← OrpheanBeholderScryDoubt
4:     repeat(64)
5:         ctext ← EncryptECB(state, ctex)
6:     return Concatenate(cost, salt, ctext)
```

Através do pseudo código acima descrito, é possível verificar que o algoritmo *bcrypt* devolve uma string composta pelo custo, *salt* e o hash da password.

Utilizando como exemplo a string "octavio"com um fator de custo 12, o algoritmo *bcrypt* retorna o seguinte:

$2y$12$au1FX9q7Ju67N3INnoBo3uYqKrkYRLPFH1.ycZ4GU9qHuo6c2FaN6

Esta string pode ser dividida em 4 secções:

- Versão do *bcrypt* utilizada (representada a vermelho).
- Fator de custo, valor entre 1 e 31 (representada a verde).
- String correspondente ao *salt* (representada a azul).
- Password após *hash* da mesma(representada a laranja).

1 Cifra assimétrica utilizada em criptografia.

A combinação destas duas técnicas faz com que, até à data de publicação desta dissertação, o algoritmo *bcrypt* ainda não tenha sido quebrado, sendo que se trata do algoritmo escolhido por sistemas operativos como o *OpenBSD*[2] e *SUSE Linux*[3] para o *hashing* das suas passwords.

Esta fama levou a que o *bcrypt* seja considerado por muitos, o standard da indústria em termos de encriptação de passwords.

4.1.1 *Funcionamento do bcrypt*

De modo a proporcionar uma forma de aumentar o tempo computacional o algoritmo *bcrypt*, Niels Provos e David Mazières basearam-se na cifra *Blowfish* já existente, criando assim uma versão designada por *Eksblowfish* (***Expensive key schedule** Blowfish*).

Algorithm 2 Pseudo código do algoritmo *EksBlowfish*.

```
1: function EksBlowfishSetup(cost, salt, key)
2:     state ← InitState()
3:     state ← ExpandKey(state, salt, key)
4:     repeat(2^cost)
5:         state ← ExpandKey(state, 0, salt)
6:         state ← ExpandKey(state, 0, key)
7:     return state
```

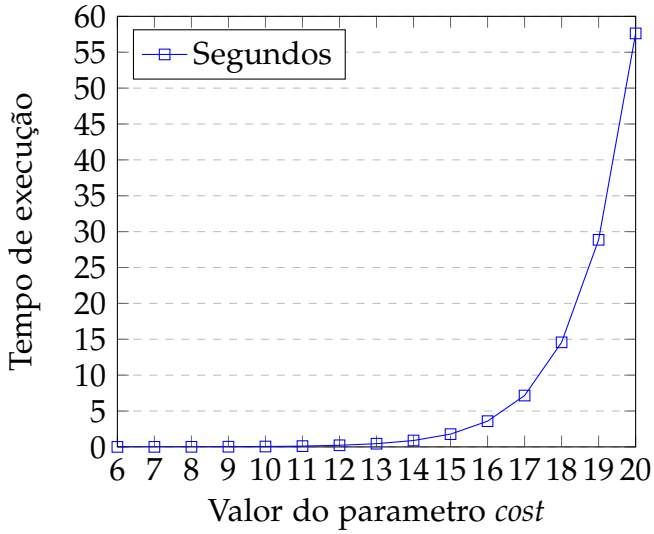
Esta alteração à cifra *Blowfish* não foi feita com o intuito de a tornar criptograficamente mais forte, mas sim alterar a forma como a *key* é calculada.

Tendo em consideração que a *key* é um valor aleatório variável entre 1 e 72 bytes, inclusive, esta alteração torna o cálculo da *key* demoroso a nível computacional (**Expensive**).

De modo a prolongar o tempo de execução desta função, é aplicada a expansão da *key* 2^{cost} vezes. Esta alteração é a maior distinção entre o algoritmo *Blowfish* original e o *Eksblowfish*, sendo a causa do *bcrypt* ser considerado um algoritmo adaptável.

2 Distribuição Unix open-source https://www.openbsd.org
3 Distribuição Unix open-source desenvolvida para soluções enterprise https://www.suse.com/

No gráfico seguinte, podemos ver uma relação entre o fator de custo e o tempo de execução do algoritmo *bcrypt*[4].



4.1.2 *bcrypt no mundo real*

Como foi mencionado nas secções prévias, um dos factos de o *bcrypt* ainda ser utilizado deve-se não só à sua segurança, mas ao facto de acompanhar as evoluções tecnológicas a nível de processamento.

Quando foi originalmente lançado em 1999, devido ao tempo de execução do algoritmo *bcrypt* ser adaptável, começou por ser utilizado um fator de custo de 6, pois este era o valor para o qual resultava um tempo de execução de aproximadamente 250ms (considerado por muitos o *standard*).

Devido a enormes avanços tecnológicos, hoje em dia um fator de custo 6 na máquina previamente descrita leva a um tempo de execução de apenas 3.6ms, ou seja, aproximadamente 70 vezes menor do que em 1999.

Para combater esta crescente incessável de poder computacional, o fator de custo deve ser ajustado de acordo com o hardware atual, sendo que hoje em dia o fator de custo mais comum é de 12 a 14, o que nosso teste levou a um tempo de execução entre 226ms e 890ms.

De modo a exemplificar a importância do fator de custo, vamos imaginar o seguinte cenário:

- Aplicação com 1000 utilizadores, cujas passwords estão contidas num dos conjuntos em baixo especificados.

4 Teste efetuado com processador Intel Core i7-4770K 4C/8T

- Conjunto de passwords mais usadas:
  1. 100 passwords.
  2. 1000 passwords.
  3. 10000 passwords.
- É permitido tentativas de autenticação ilimitadas, eliminando assim qualquer defesa contra ataques *bruteforce*.

Quanto tempo demoraria a um atacante, com um processador idêntico ao descrito anteriormente, testar todas as passwords para os 1000 utilizadores?

Tempo necessário para testar todas as passwords para os 1000 utilizadores.			
Fator de custo	100 passwords	1000 passwords	10000 passwords
6	6 minutos	1 hora	10 horas
7	12 minutos	2 horas	21 horas
8	23 minutos	4 horas	2 dias
9	48 minutos	8 horas	3 dias
10	92 minutos	15 horas	6 dias
11	3 horas	30 horas	13 dias
12	6 horas	3 dias	1 mês
13	12 horas	5 dias	2 meses
14	1 dia	10 dias	3 meses
15	2 dias	21 dias	7 meses
16	4 dias	1 mês	1 ano
17	1 semana	3 meses	2 anos
18	2 semanas	6 meses	5 anos
19	1 mês	11 meses	9 anos
20	2 meses	2 anos	18 anos

Tabela 3: Tempo necessário para testar combinações de passwords para 1000 utilizadores.

Facilmente percebemos que fatores de custo como 10 são demasiado baixos para a atualidade, sendo imperativo a escolha de um fator de custo equilibrado. Outro facto a ter em consideração é este teste ter sido feito com um processador *mainstream*, sendo que o mesmo se encontra 4 gerações atrasado em comparação com a atual 9ª geração de processadores Intel.

Outro problema que tem ganho tração nos últimos anos, é o aparecimento de soluções especializadas sobre a forma de GPU, *Field-programmable gate array (FPGA)*(20)(14)

e *Application-specific integrated circuit (ASIC)*, capazes de poder computacional extremamente superior, quando comparados com o CPU utilizado para os testes anteriores.

Um FPGA é um circuito integrado que tem a possibilidade de ser reprogramado através de *bitstreams*, de modo a ser utilizado em aplicações diferentes, como por exemplo, cálculos de hash baseados no algoritmo SHA-1, SHA-256, etc. Enquanto que um ASIC, embora também seja um circuito integrado, apenas consegue fazer uma função e não pode ser reprogramado, no entanto oferece performance muito superior a um FPGA.

Na tabela 4 exploramos a performance em hashes por segundo (H/s) entre CPU[5], GPU[6] e FPGA[7] relativamente ao algoritmo *bcrypt*.

Cálculo de hashes por segundo (H/s) em diverso hardware.					
Tipo	Modelo	Custo 6	Custo 12	Consumo	Preço
CPU	Xeon E3-1240	6210 H/s	50 H/s	300W	262$
GPU	GTX 750Ti	1920 H/s	15 H/s	300W	120$
FPGA	zedboard	6511 H/s	51.95 H/s	4.2W	319$
FPGA	Virtex-7	51437 H/s	410.4 H/s	20W	3495$

Tabela 4: Comparação entre hardware no cálculo de H/s, eficiência e custo.(20)

Mais uma vez é possível identificar outro um problema que não foi previamente considerado: como ajustar o fator de custo para este tipo de hardware especializado, como por exemplo FPGA e ASIC?

Infelizmente a resposta é que este ajuste é deveras impossível. O simples aumento do fator de custo é impensável, pois iria implicar tempos exponenciais de computação em CPU mainstream.

Felizmente, tal aplicação de hardware pare este efeito ainda não foi detetada no mundo real e não aparenta qualquer problema de momento.

5 CPU fabricado pela Intel, desenvolvido na arquitetura *Sandy Bridge*.
6 GPU fabricada pela NVIDIA, desenvolvida na arquitetura *Maxwell*.
7 Ambos os FPGA utilizados para comparação são desenvolvidos pela Xilinx.

4.2 AUTENTICAÇÃO DE UTILIZADORES

A autenticação e gestão de utilizadores é feita através da combinação do middleware[8] designado por *Passport* e a base de dados não-relacional[9] implementada em *MongoDB*.

Devido à informação sensível que pode ser guardada na mesma, campos como a password são, naturalmente, encriptados recorrendo à técnica discutida na secção 4.1.

O *Passport* é um middleware de autenticação, desenvolvido para *NodeJS*, com mais de 500 estratégias[10] diferentes de autenticação. Foi criado para solucionar um único problema: **autenticar pedidos**.

Devido à natureza do *Passport*, este é extremamente fácil de implementar numa dada aplicação, devido ao seu forte encapsulamento e ao facto de delegar qualquer funcionalidade, que não seja a autenticação, para a aplicação.

Devido à natureza das aplicações Web modernas, a autenticação pode ser feita por diversas estratégias. As estratégias exploradas neste projeto recairam sobre a escolha mais "tradicional", uma autenticação local com campos para *username* e *password* e a autenticação com o auxílio da ferramenta Autenticação.Gov.

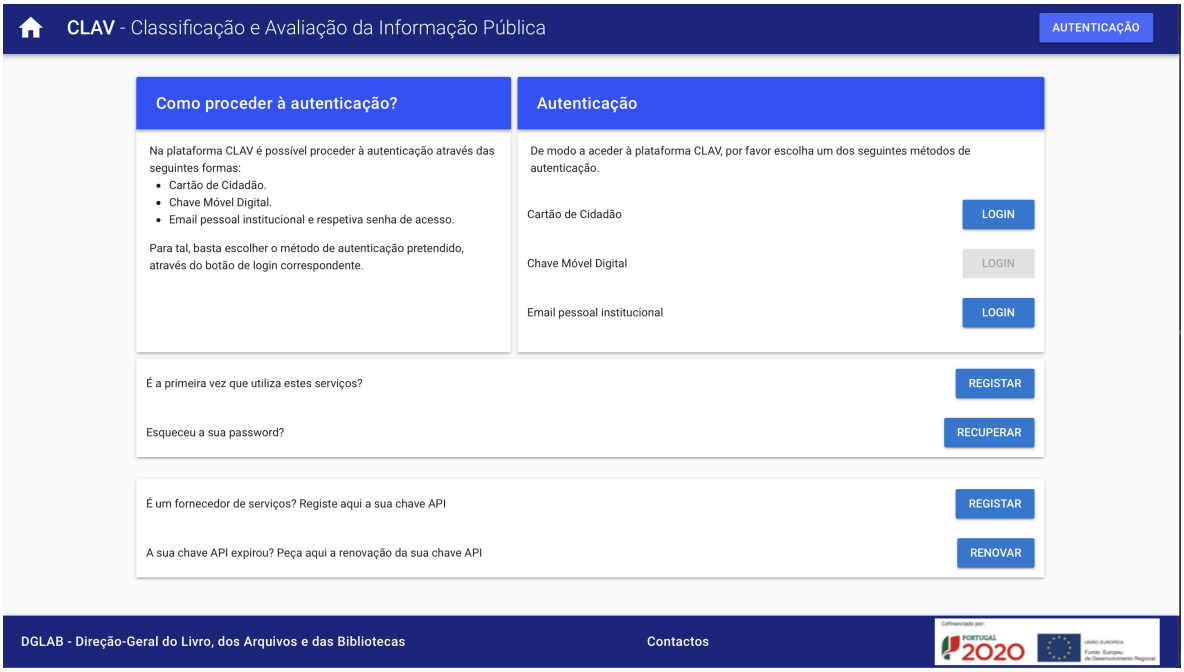


Figura 22: Página principal de autenticação na plataforma CLAV.

8 Middleware é um software que funciona como intermediário entre dois programas.
9 Estilo de base de dados livres de esquema, capazes de maior escalabilidade que as base de dados tradicionais.
10 Uma estratégia pode ser interpretado como um mecanismo único de autenticação.

Embora exista um aumento exponencial de implementações de autenticação baseadas em redes sociais, através de protocolos desenvolvidos em acordo com o *OAuth*[11], tais como *Google+* e *Facebook*, foi decidido desde uma fase inicial que tais estratégias não seriam consideradas devido ao caráter profissional da plataforma.

Devido a cada aplicação possuir necessidades diferentes, o *Passport* guarda cada estratégia de autenticação em módulos independentes, deixando assim a cargo do programador que estratégias empregar, sem criar dependências desnecessárias.

4.2.1 *Autenticação através de username e password*

Nesta secção iremos explorar a implementação da autenticação através de username a password, designada por autenticação local. Começaremos por analisar a página de registo, seguida de uma análise do diagrama de sequência da mesma, explicando em detalhe o processo de registo do início ao fim.

4.2.1.1 *Registo*

O registo local, através de username e password foi implementado com o objetivo de ser intuitivo e simples.

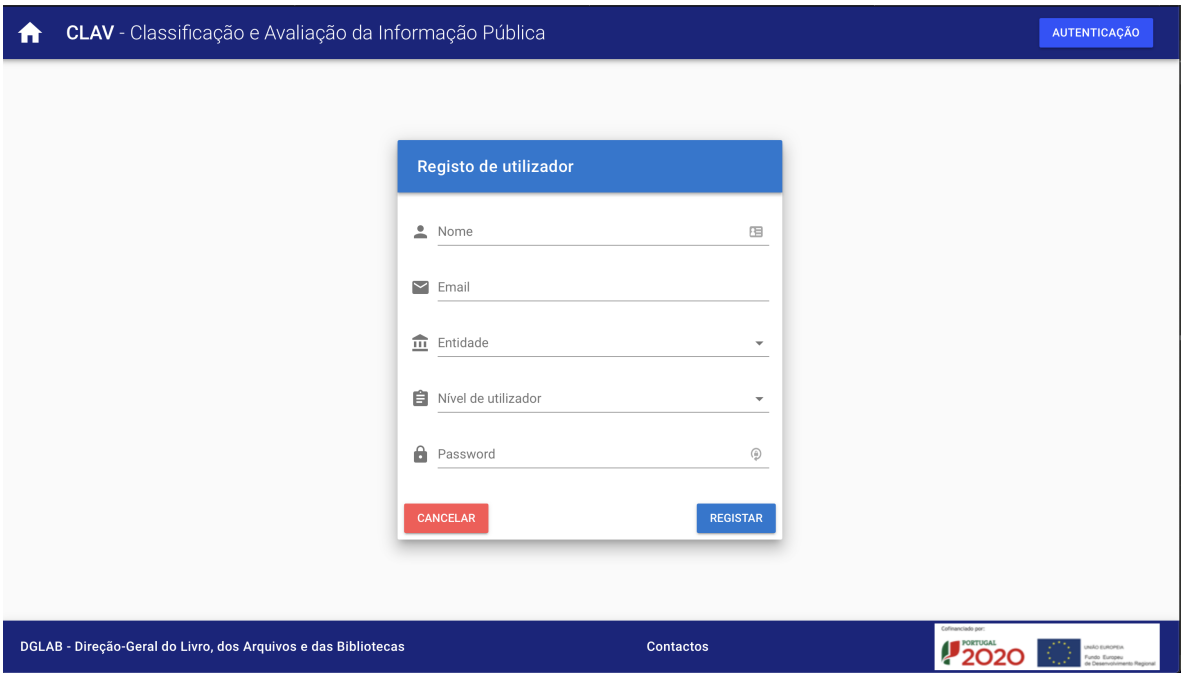


Figura 23: Página de registo na plataforma CLAV.

11 Protocolo open-source que permite autenticação de forma standard e segura, entre aplicações desktop, web e mobile.

Para tal é utilizado um form, onde é necessário o preenchimento da seguinte informação (como foi especificado na secção 3.1.1):

1. Nome.
2. Email.
3. Entidade.
4. Nível de utilizador.
5. Password (encriptada através de *bcrypt*, explicado na secção 4.1).

Após submeter o pedido de registo pode ocorrer um dos seguintes cenários:

- **Sucesso**

  Ocorre quando o utilizador fornece todos os dados necessários para o seu registo.

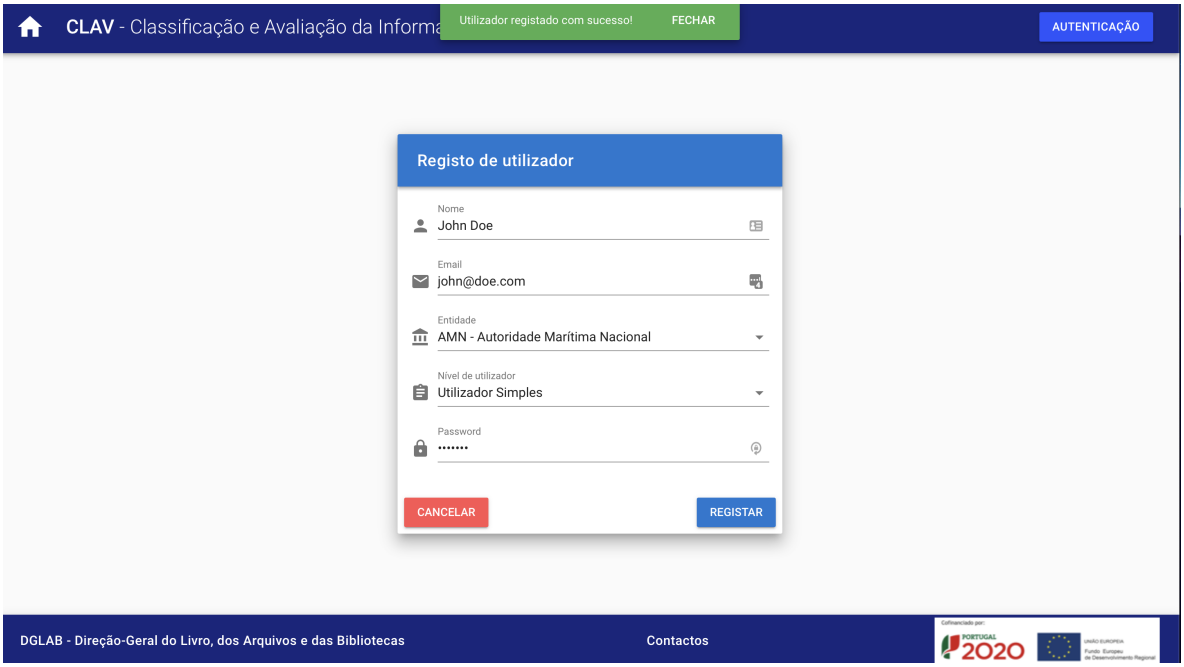


Figura 24: Página quando ocorre um registo com sucesso na plataforma CLAV.

- **Insucesso**
  - **Dados insuficientes**

    Ocorre quando o utilizador não preencheu todos os campos necessários para o registo.

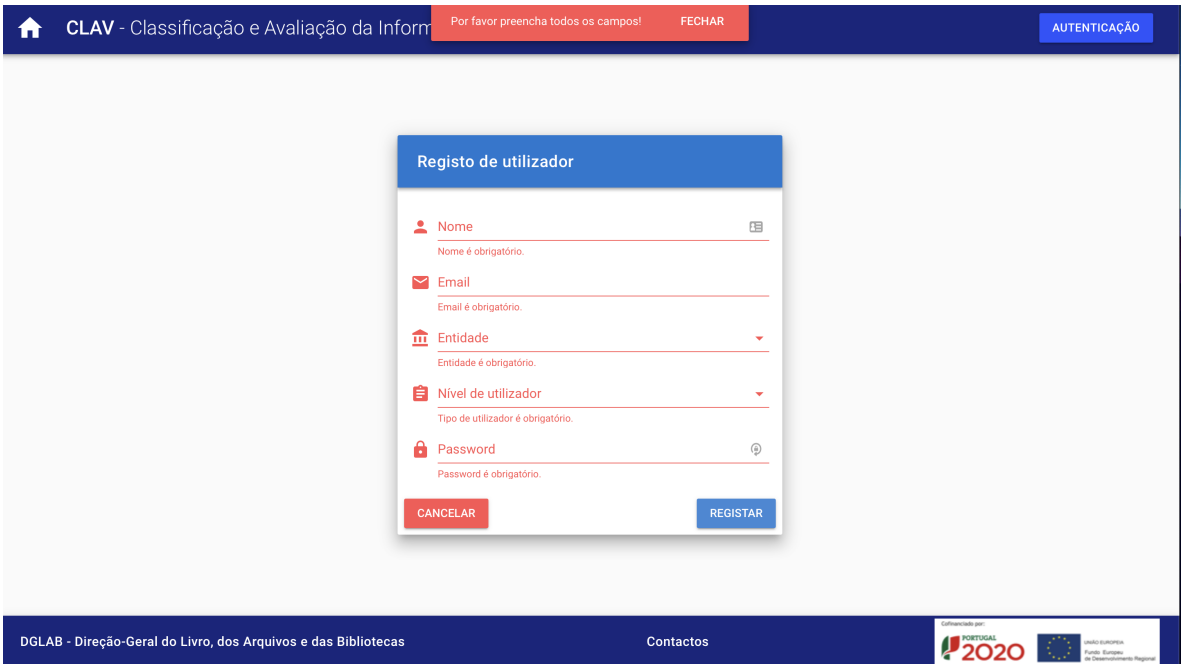


Figura 25: Página quando ocorre um erro de registo na plataforma CLAV.

– **Email já registado**

Ocorre quando o utilizador fornece um email já atribuido a outro utilizador.

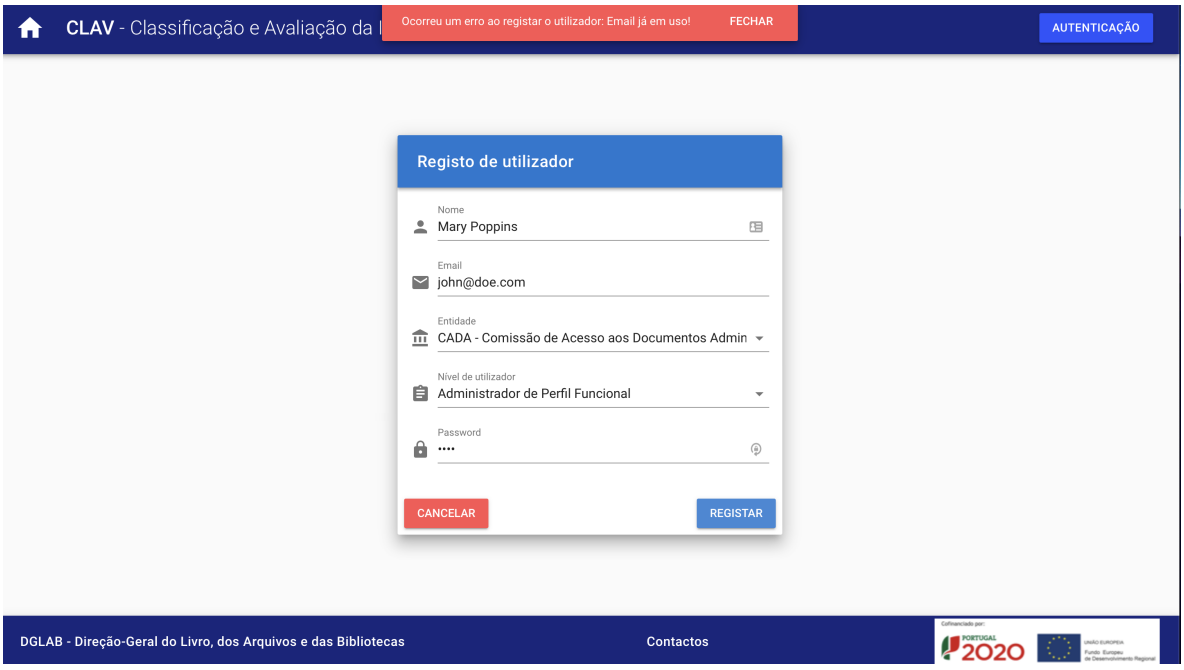


Figura 26: Página quando ocorre um erro de registo na plataforma CLAV.

O comportamento da função de registo pode ser explicado pelo diagrama de sequência da figura 27.

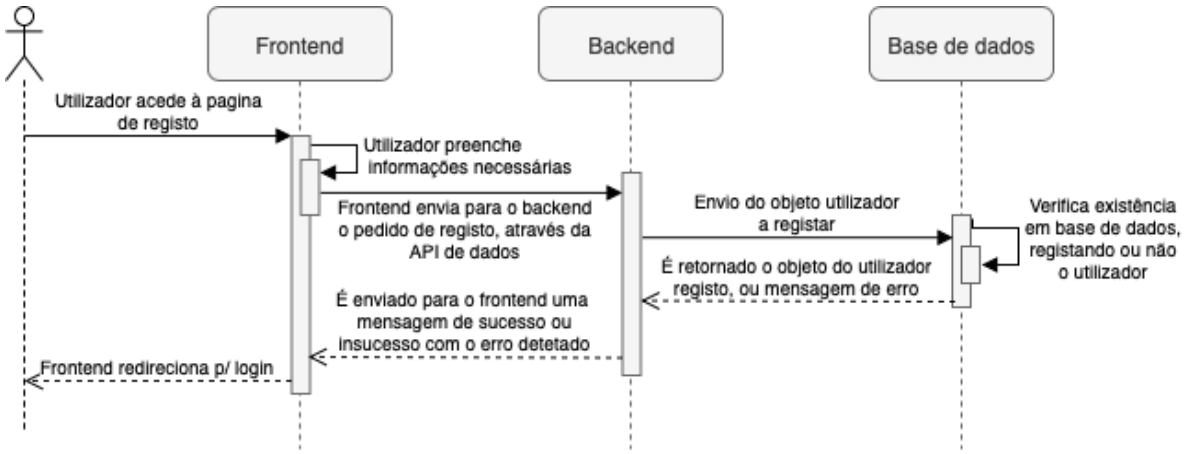


Figura 27: Diagrama de sequência correspondente ao processo de registo na plataforma CLAV.

4.2.1.2 *Login*

Tendo em consideração a página de registo local, foi implementada uma solução similar, apenas com os campos necessários para login, ou seja, email e password.

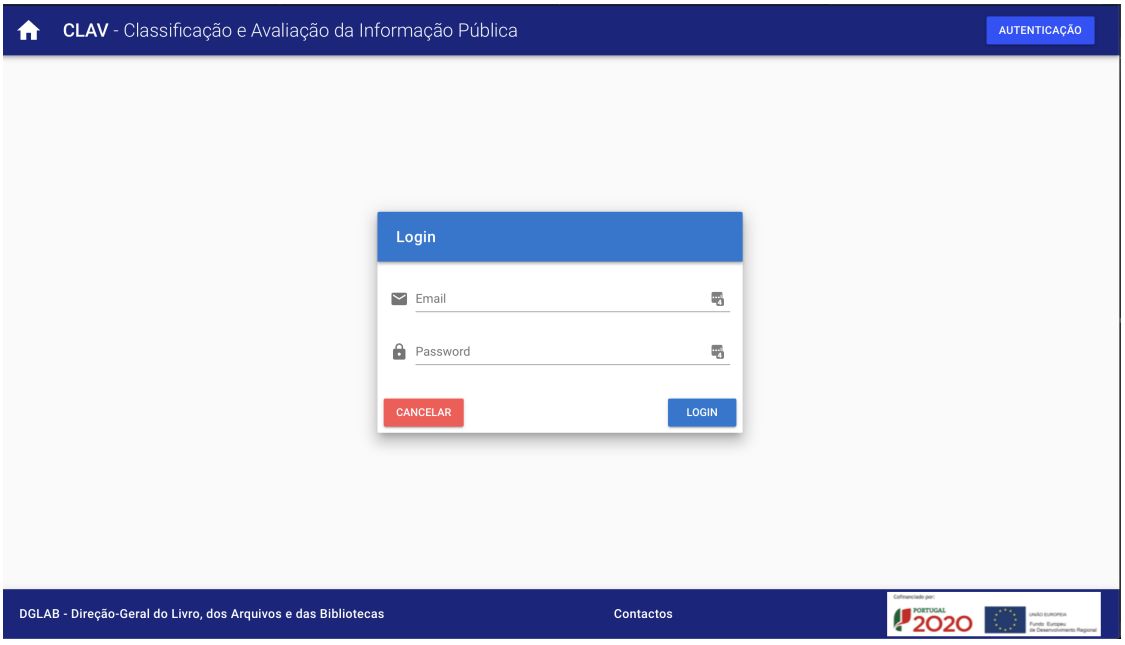


Figura 28: Página de login através de username e password na plataforma CLAV.

Após submeter o pedido de login pode ocorrer um dos seguintes cenários:

- **Sucesso**

  Ocorre quando o utilizador fornece uma combinação de email e password correta.

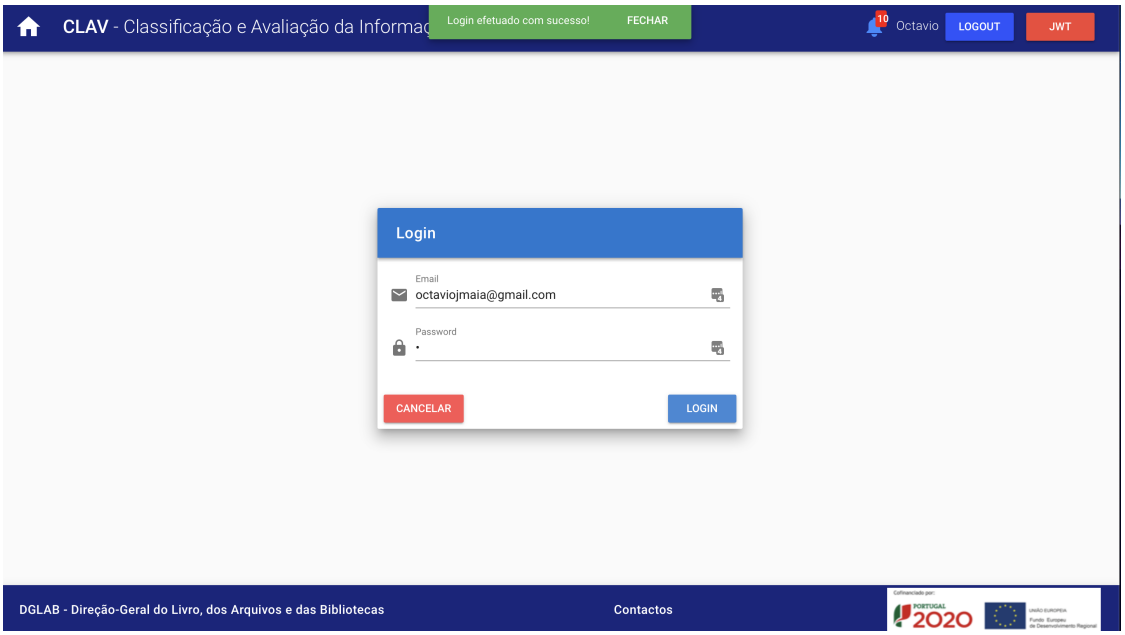


Figura 29: Página quando ocorre um login com sucesso na plataforma CLAV.

- **Insucesso**
  - **Dados insuficientes**

    Ocorre quando o utilizador não preencheu todos os campos necessários para o login.

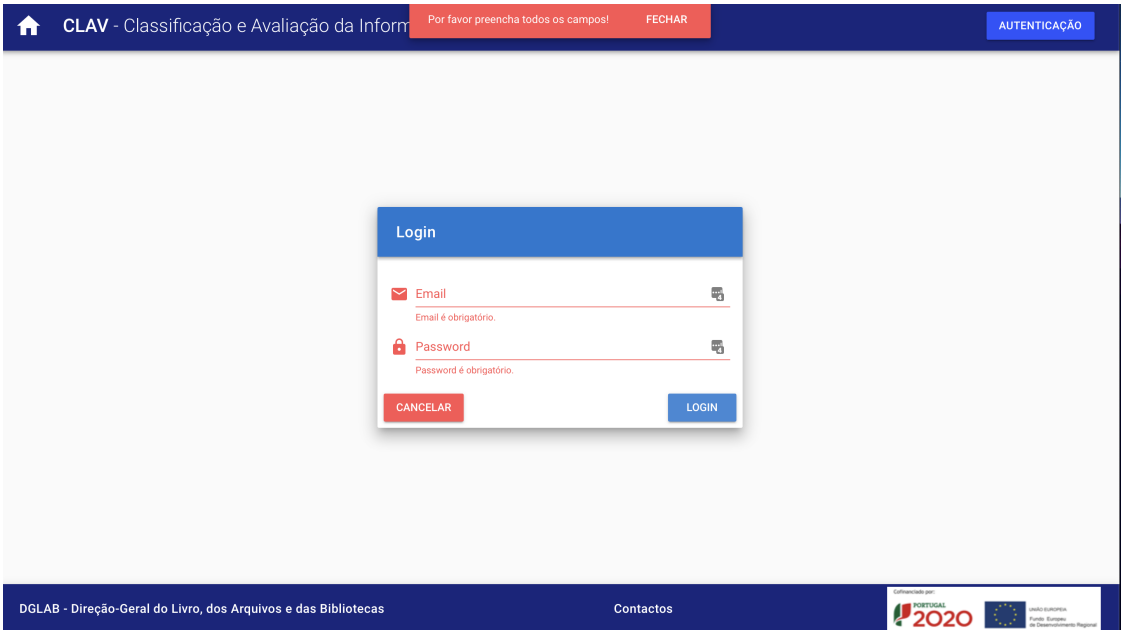


Figura 30: Página quando ocorre um erro de login na plataforma CLAV.

– **Dados incorretos**

Ocorre quando o utilizador fornece uma combinação de email e password incorreta.

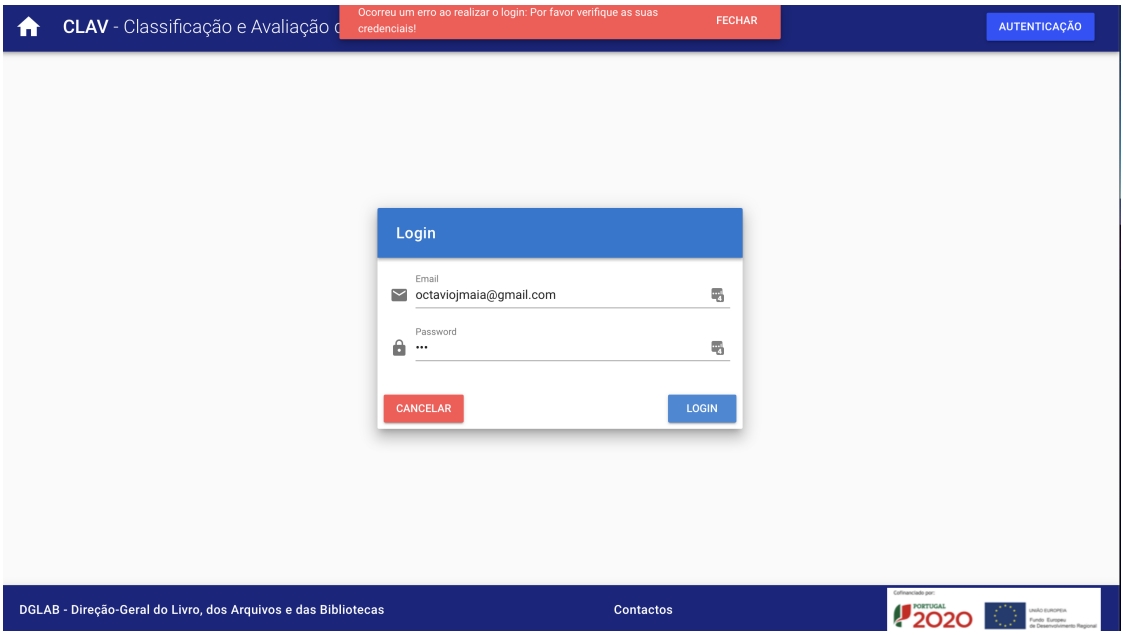


Figura 31: Página quando ocorre um erro de login na plataforma CLAV.

O comportamento da função de login pode ser explicado pelo diagrama de sequência da figura 32.

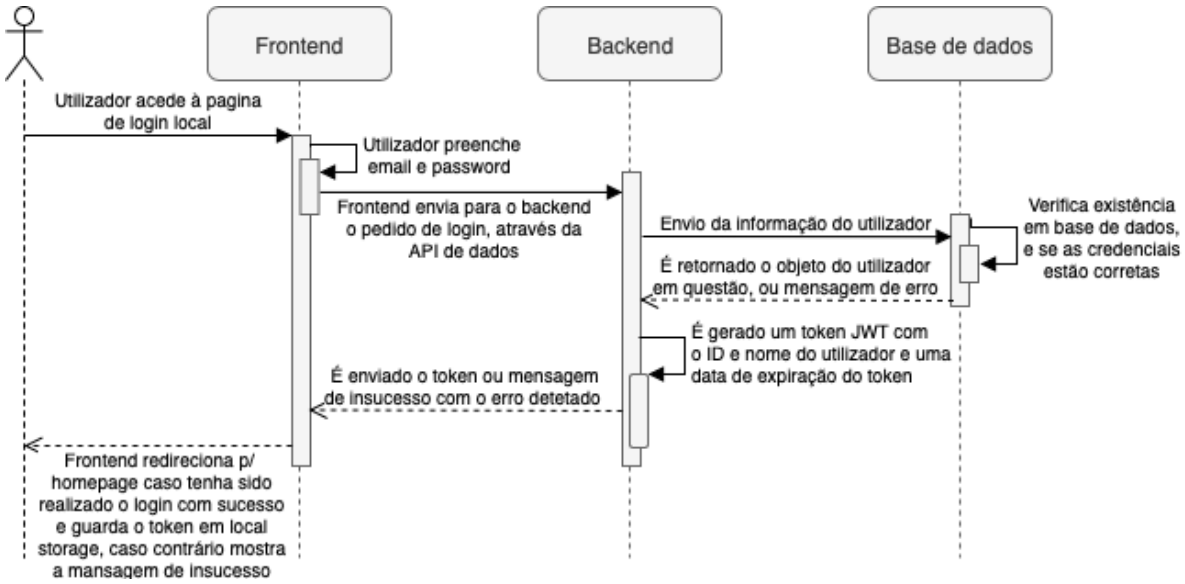


Figura 32: Diagrama de sequência correspondente ao processo de login local na plataforma CLAV.

4.2.1.3 *Recuperação de credenciais*

Caso o utilizador se esqueça das suas credenciais, existe uma página de recuperação, indicada na figura 33. Nesta página apenas é necessária a inserção do email utilizado para registo.

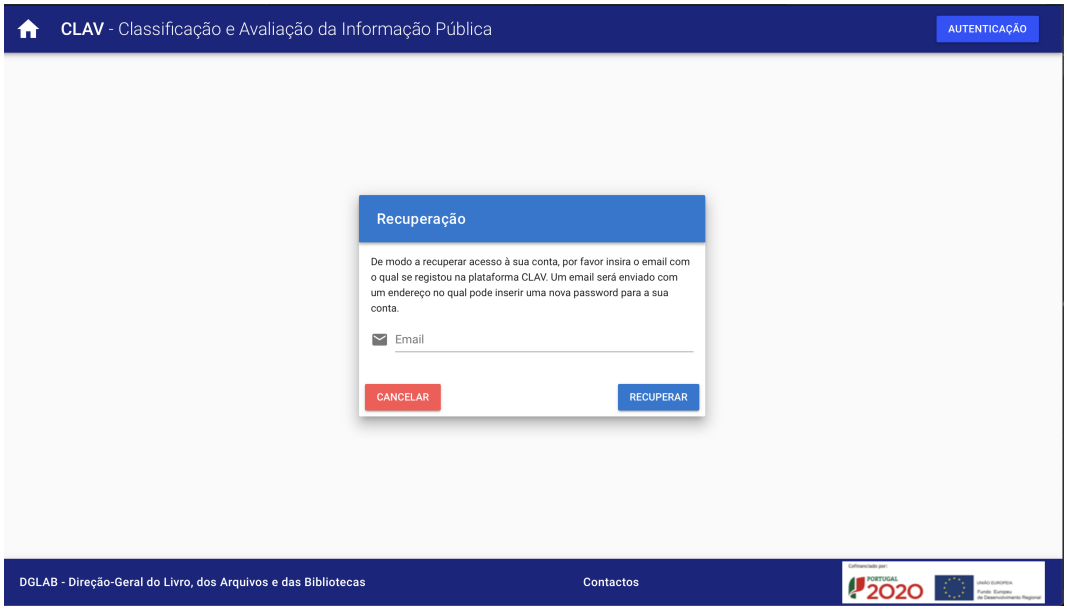


Figura 33: Página de recuperação de credenciais na plataforma CLAV.

Após inserção do email e subsequente pedido de recuperação de credenciais, pode ocorrer um dos seguintes cenários:

- **Sucesso**
  Ocorre quando o utilizador fornece um email presente em base de dados.

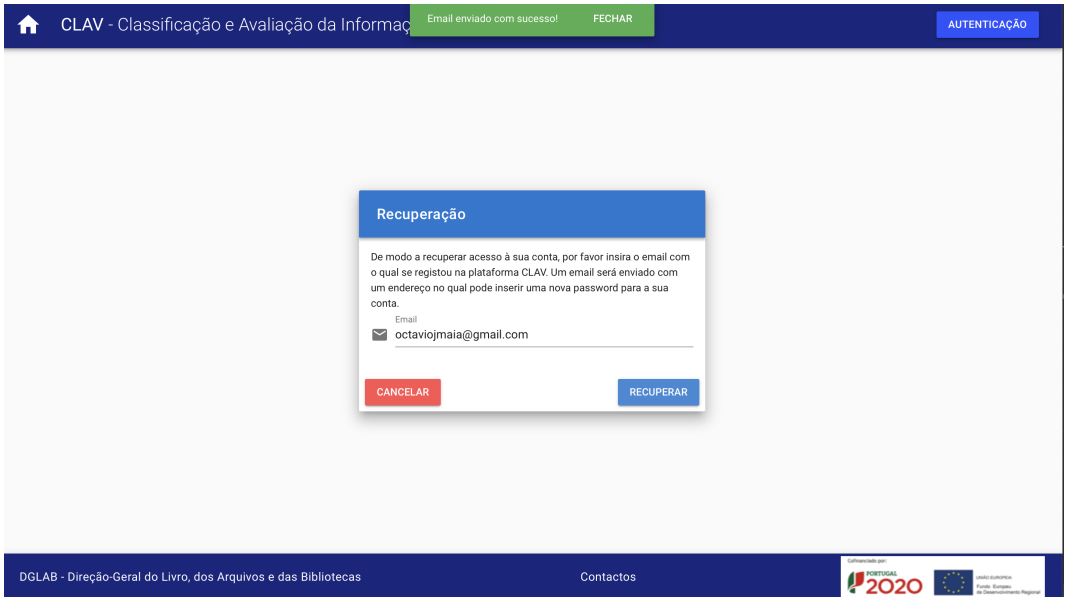


Figura 34: Mensagem de aviso do envio de email de recuperação.

De seguida, é enviado um email para esse destinatário, com um link para recuperação da password. Por questões de segurança este link é válido por 30 minutos, sendo que após esse montante de tempo, o mesmo expira.

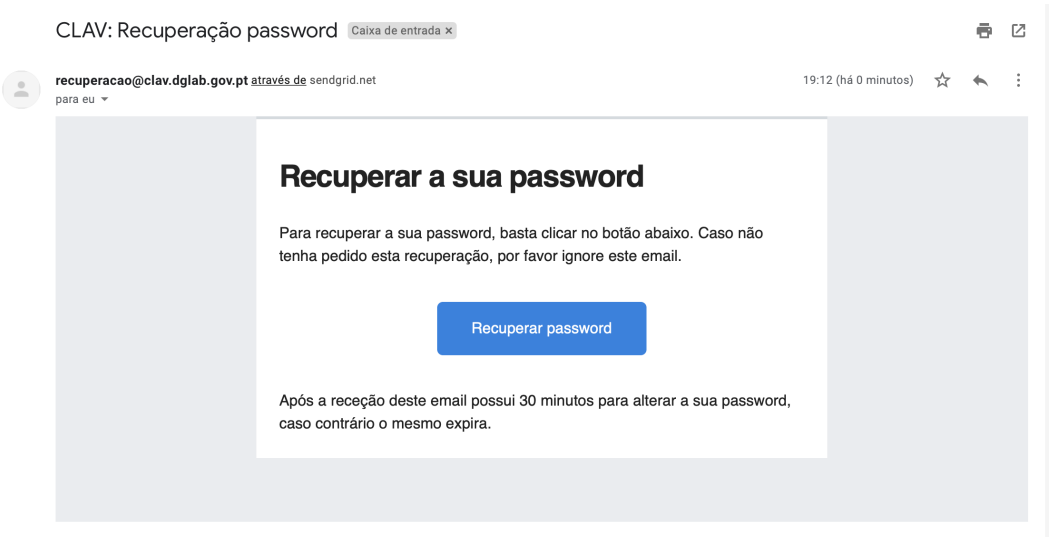


Figura 35: Email de recuperação enviado pela plataforma CLAV.

Após aceder ao link de recuperação, é possível alterar a sua password, exemplificado na figura 36.

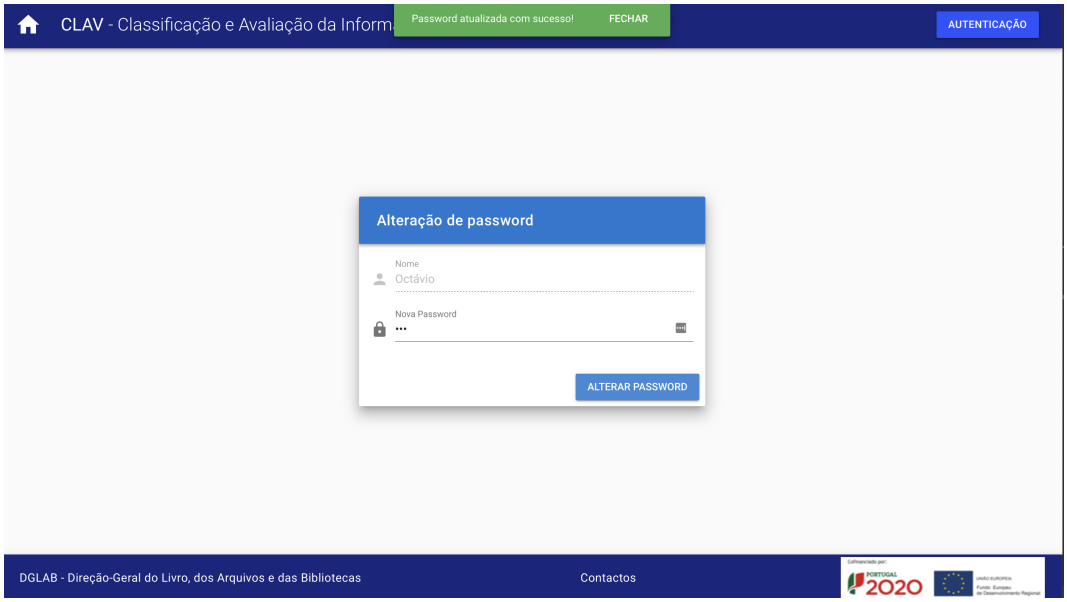


Figura 36: Mensagem de sucesso após alteração de password na plataforma CLAV.

- **Insucesso**
  - **Dados incorretos**

    Ocorre quando o utilizador insere um email que não se encontra presente em base de dados.

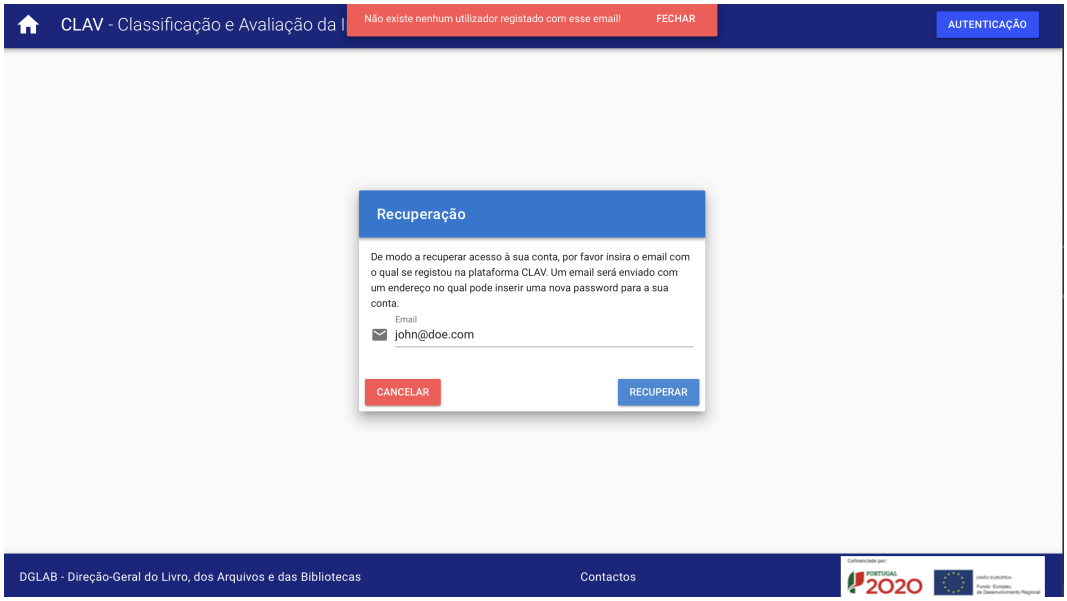


Figura 37: Mensagem de erro após inserção de email não existente na plataforma CLAV.

  - **Registo através de Cartão de Cidadão**

    Ocorre quando o utilizador insere um email associado a um registo com o Cartão de Cidadão.

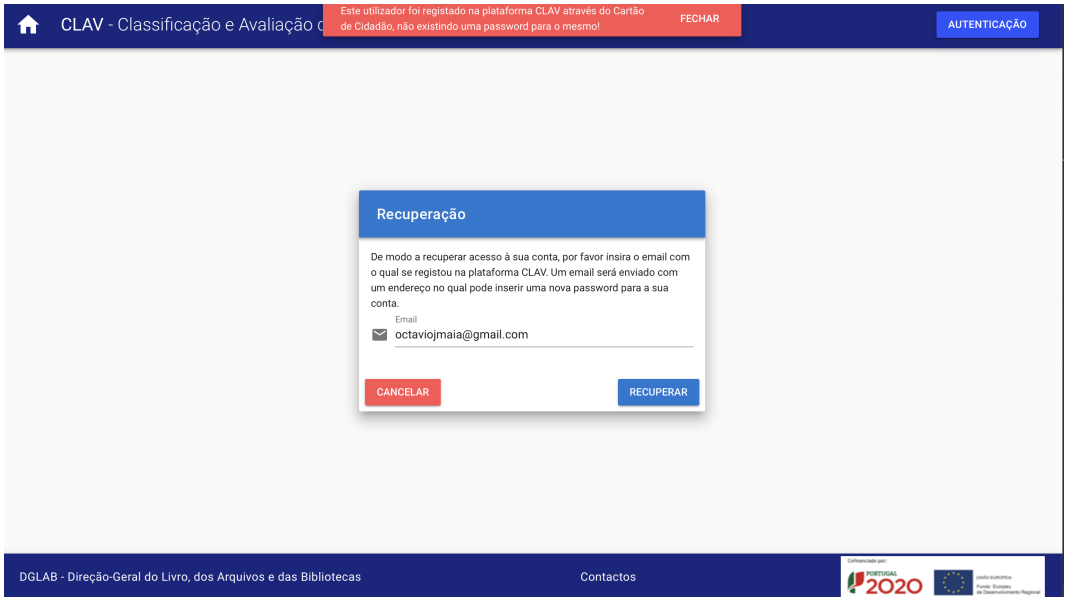


Figura 38: Mensagem de erro após tentativa de recuperação de uma conta registada com Cartão de Cidadão.

O comportamento da função de recuperação pode ser exemplificado pelo seguinte diagrama de sequência:

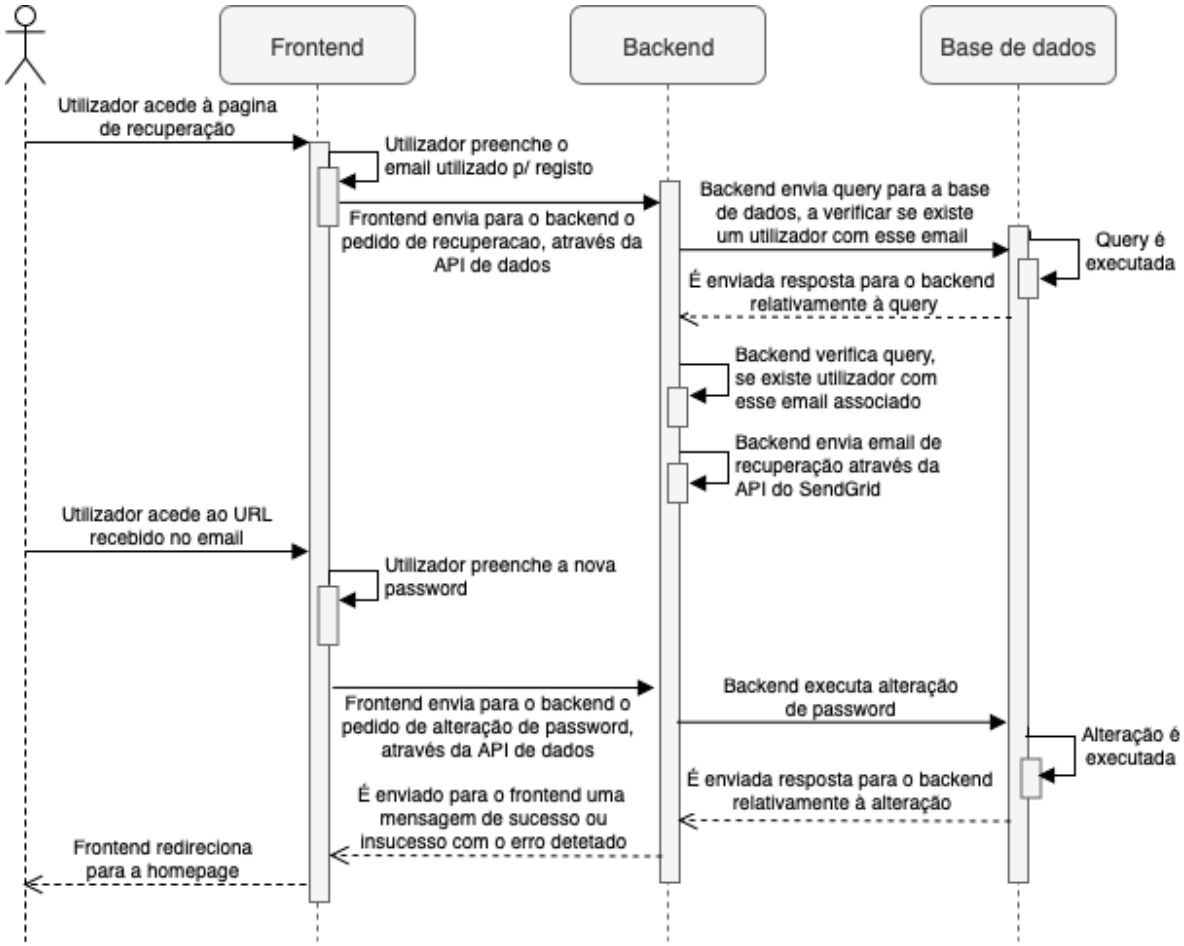


Figura 39: Diagrama de sequência relativo ao processo de recuperação na plataforma CLAV.

4.2.2 *Autenticação através de Cartão de Cidadão*

Nesta secção iremos explorar a implementação da autenticação com o Cartão de Cidadão através da ferramenta Autenticação.Gov. Iremos começar por relatar o processo de autenticação, seguido do registo de um novo utilizador, bem como o diagrama de sequência deste processo.

Após aceder ao Autenticação.Gov somos abordados com os dados que a plataforma CLAV está a requisitar, tendo este sido abordados na secção 2.2.3.1.

Figura 40: Página principal do Autenticação.Gov.

De seguida é pedido o PIN de autenticação do nosso Cartão de Cidadão.

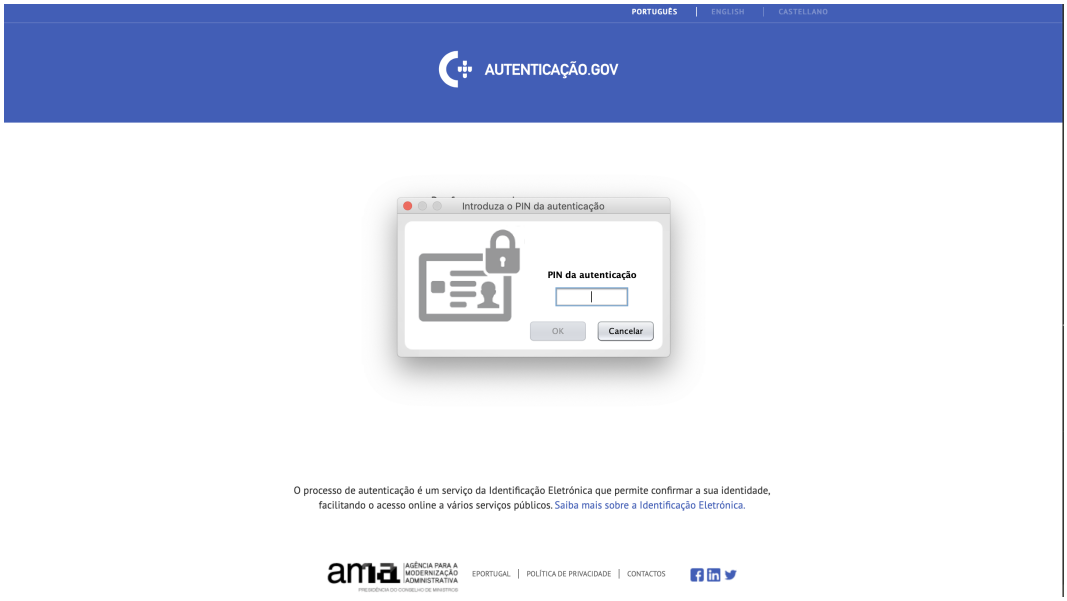


Figura 41: Pedido de inserção do PIN de autenticação por parte do Autenticação.Gov.

Após esta etapa podem ocorrer dois cenários:

- **Sucesso**

  Ocorre quando o utilizador fornece o PIN de autenticação correto.

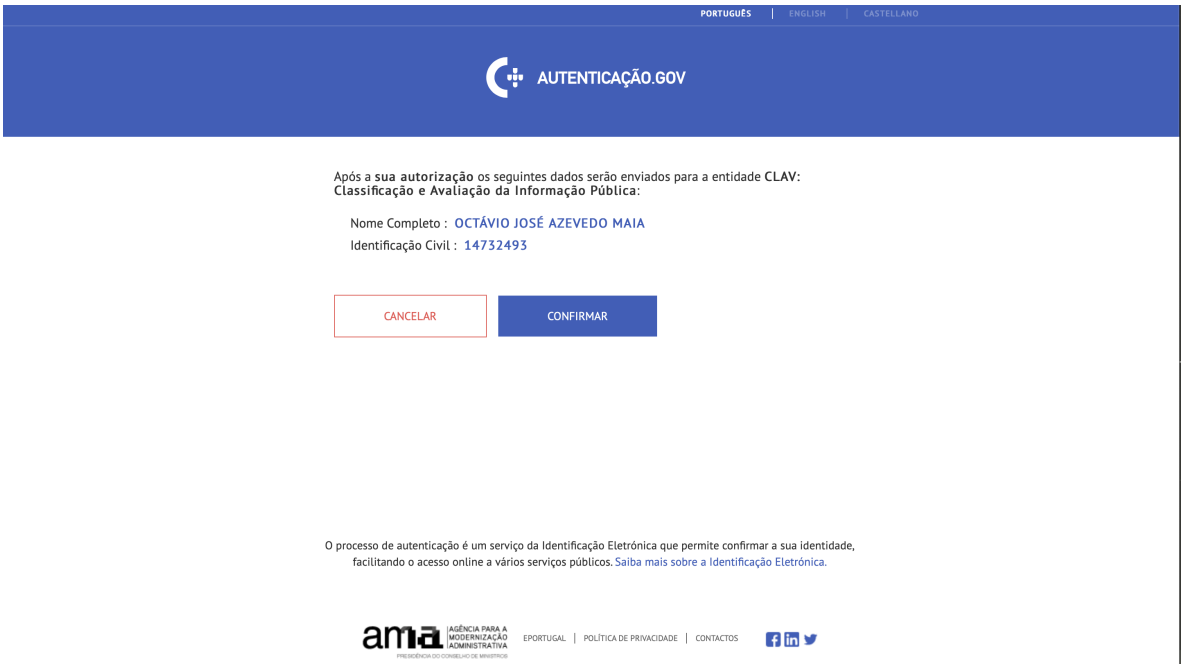


Figura 42: Listagem dos dados requisitados pela plataforma CLAV.

Após correta inserção do PIN de autenticação, é necessário verificar se já existe algum utilizador registado com o Número de Identificação Civil (NIC) devolvido pelo Autenticação.Gov.

  – **Já existe utilizador registado**

    É iniciada a sessão com o utilizador correspondente a esse NIC seguido do redireccionamento do utilizador para a página principal.

  – **Não existe utilizador registado**

    É feito o redireccionamento para a página de registo de um novo utilizador.

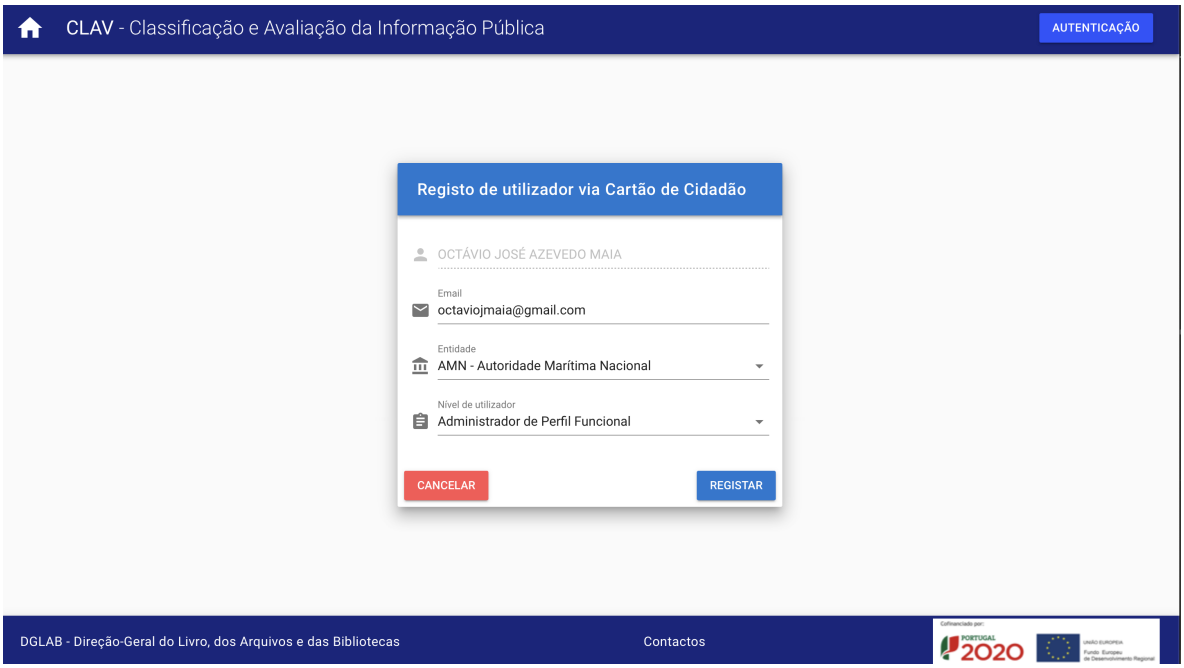


Figura 43: Página de registo de um novo utilizador através do Cartão de Cidadão.

- **Insucesso**

  Ocorre quando o utilizador introduziu um PIN de autenticação incorreto ou negou o acesso à informação requisitada.

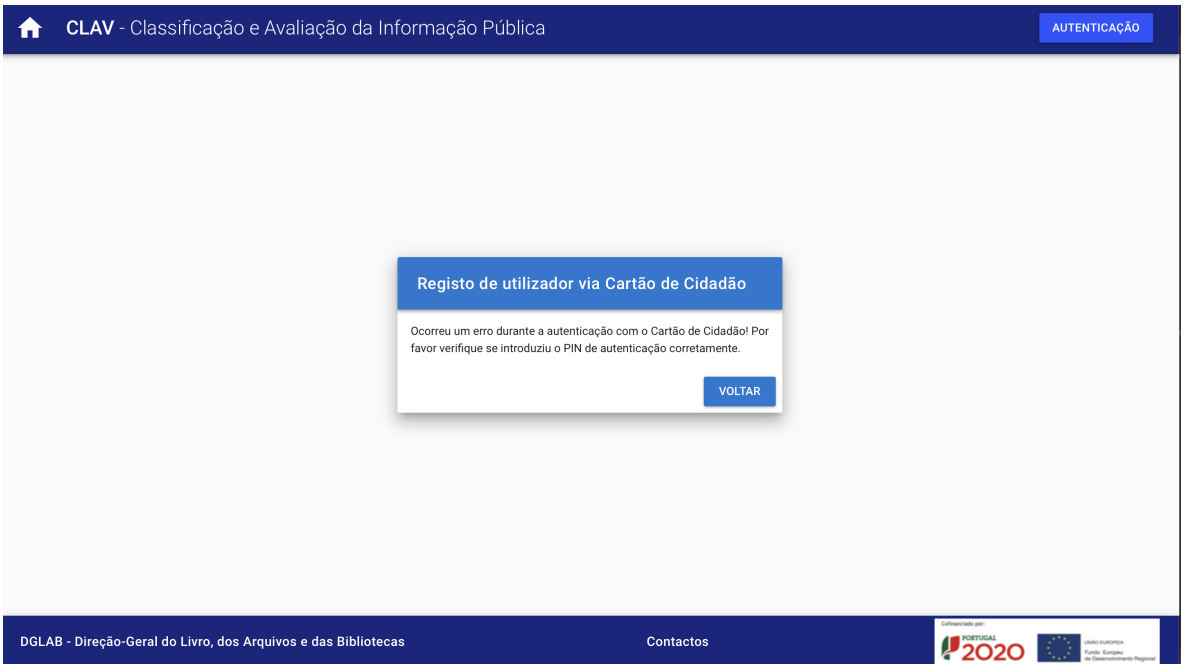


Figura 44: Página de erro resultante de um PIN incorreto ou negação da leitura dos dados pretendidos.

O comportamento da função de registo e login via Cartão de Cidadão pode ser exemplificado pelo seguinte diagrama de sequência:

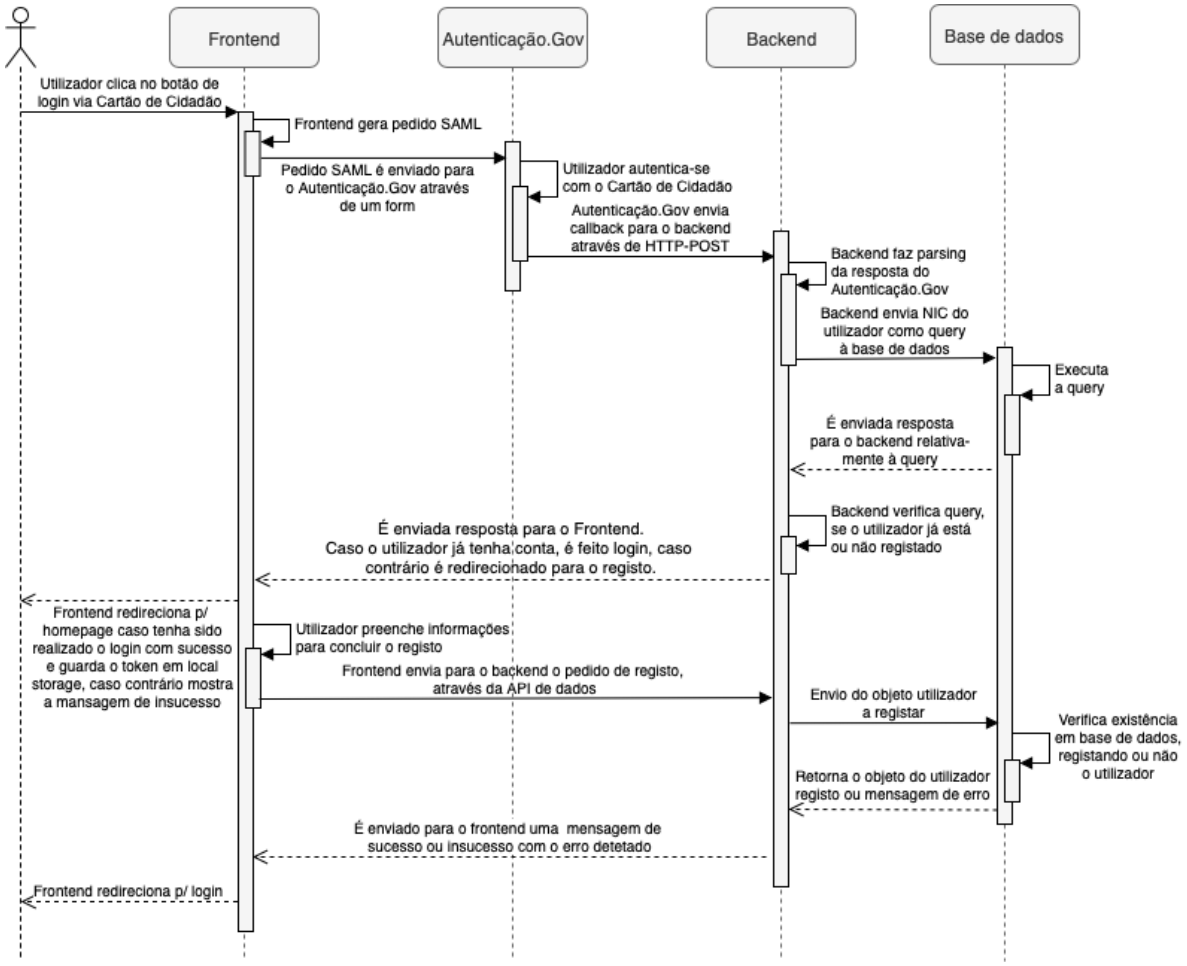


Figura 45: Diagrama de sequência relativo ao processo de registo e login através de Cartão de Cidadão na plataforma CLAV.

4.3 GESTÃO DE UTILIZADORES

Com o previsto aumento exponencial do número de utilizadores da plataforma CLAV, foi necessário proceder à implementação de um mecanismo capaz e gerir, editar e possivelmente desativar certos utilizadores. Para tal, foi desenvolvida a seguinte página, de acordo com as especificações documentadas na secção 3.4.

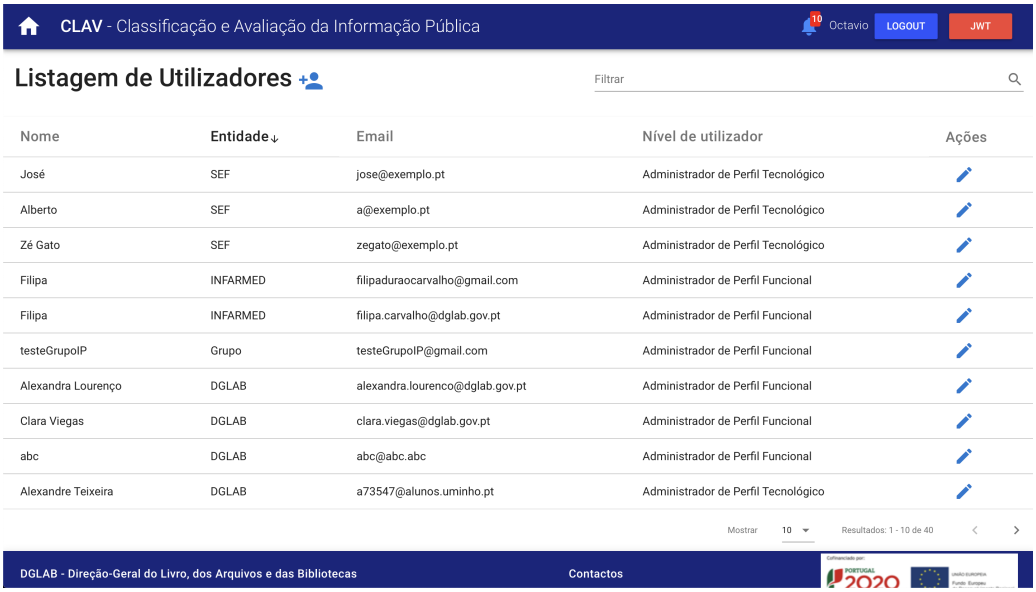


Figura 46: Página de listagem de todos os utilizadores da plataforma CLAV.

Após listar todos os utilizadores é possivel ordenar através de nome, entidade, email ou nível de utilizador, bem como editar o utilizador correspondente através do icone da direita (representado por um lápis).

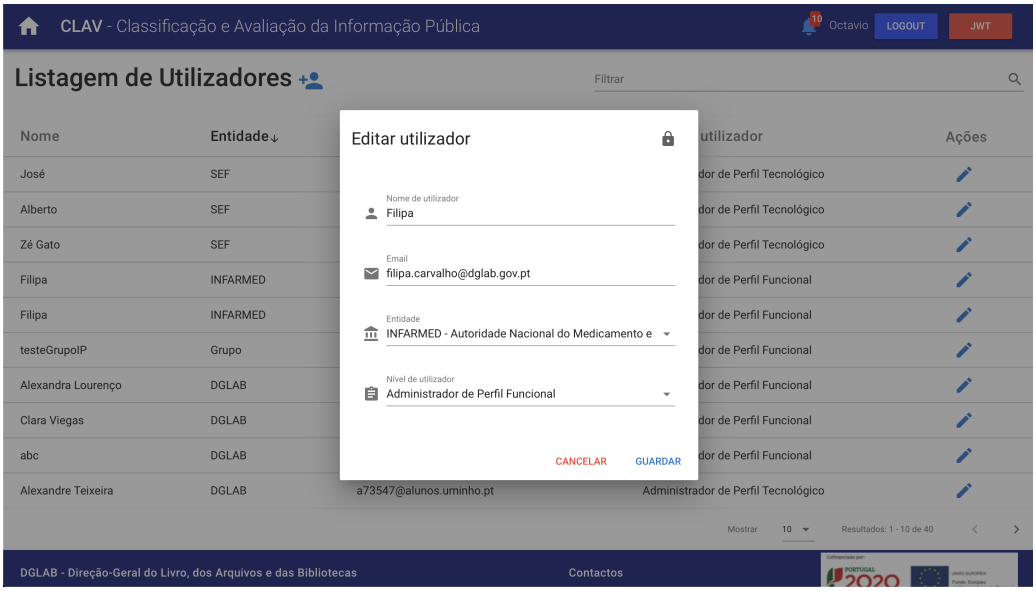


Figura 47: Página de edição de um utilizador da plataforma CLAV.

Após clicar no icon de edição é possivel editar os seguintes campos:

- Nome.
- Email.
- Entidade.
- Nível do utilizador.

O comportamento da função de listagem e edição de utilizadores pode ser exemplificado pelo seguinte diagrama de sequência:

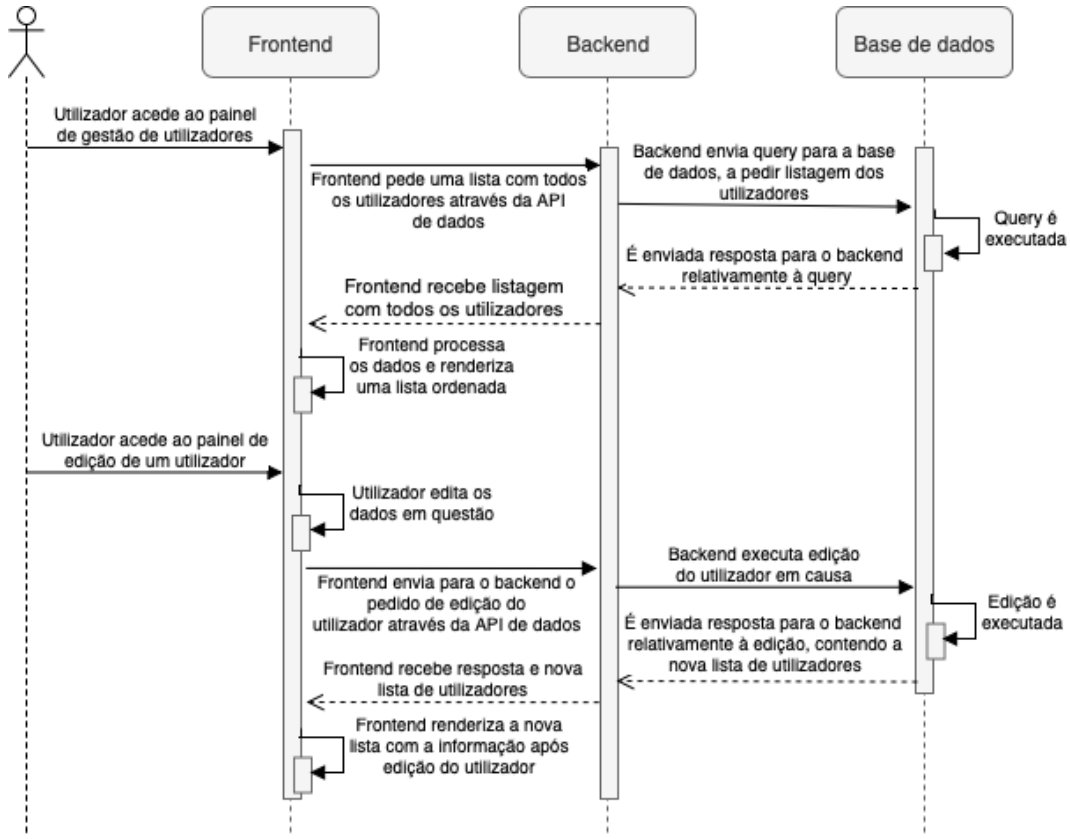


Figura 48: Diagrama de sequência relativo à gestão e edição de utilizadores na plataforma CLAV.

4.3.1 *Armazenamento e validação de sessões*

De modo a fazer uma gestão de utilizadores mais aprofundada, foi necessário implementar métodos capazes de não só limitar o número de utilizadores ativos a cada dado momento, mas também manter as sessões iniciadas no caso de falha do servidor.

Numa primeira fase do projeto CLAV foi implementada uma solução baseada em *JSON Web Token*, que será explicada na subsecção 4.3.1.1. Embora esta se tenha provado eficaz, a mesma não era capaz de manter as sessões iniciadas no caso de uma falha súbita do servidor, ou seja, se o servidor por algum motivo sofresse de uma falha de energia e tivesse de reiniciar, devido às sessões estarem guardadas localmente, as mesmas eram perdidas.

Outro fator decisivo contra esta implementação assenta no facto de não ser possível recorrer ao paralelismo da aplicação através do *nginx*[12], pois a comunicação entre processos paralelos não permite a partilha do token guardado localmente.

Para tal, foi necessário recorrer a outra abordagem, o armazenamento de sessões em base de dados, neste caso em *MongoDB*.

Esta segunda implementação em *MongoDB*, embora totalmente funcional, teria de ser abandonada devido à divisão da aplicação especificada no capítulo 4. Assim sendo, as secções 4.3.1.1 e 4.3.1.2 podem ser encaradas como processos iterativos na cadeia de desenvolvimento da plataforma CLAV, sendo a secção 4.3.1.3 representativa do estado atual do projeto.

4.3.1.1 *Validação através de JSON Web Token*

Durante o desenvolvimento do projeto CLAV foi discutida a implementação de um mecanismo capaz de terminar sessões automaticamente, de modo a diminuir o número de sessões ativas, bem como evitar possíveis falhas de segurança.

Para tal foi implementado, juntamente com o *Passport*, uma estratégia capaz de lidar com JWT, representada através do seguinte pseudo código.

Algorithm 3 Pseudo código da autenticação via *JSON Web Token*.

```
1: function JwtStrategy(opts, jwt_payload, done)
2:     user ← findUserByJwt(jwt_payload.id)
3:     if user ≠ invalid then
4:         return done(null, user)
5:     else
6:         return done(null, false)
```

12 Mais informação em https://www.nginx.com

Através do uso do *JSON Web Token*, é possível povoar o atributo *exp* com uma data de expiração, após a qual o token se torna inválido, terminando assim a sessão do utilizador.

Para tal, quando a estratégia de login local do *Passport* é invocada, é necessário proceder à assinatura de um token JWT e atribuí-lo à sessão do utilizador atual.

Algorithm 4 Pseudo código da atribuição de um JWT à sessão.

```
1: function LOGIN(user)
2:     token ← jwt.sign(user, jwt.secret, {
3:         expiresIn : jwt.expiration
4:     }
5:     session.token ← token
```

Os valores de *secret* e *expiration* previamente utilizados são provenientes de um ficheiro externo, no qual designamos a duração (em segundos) em que a sessão está ativa, bem como a chave secreta utilizada para a assinatura.

```
jwt: {
    secret: 'ihfH6Cv4LVD1YoG...vY8XdVch5ebGHTdPMDGHy',
    expiration: 3000 //in seconds
}
```

Após corretamente atribuída a sessão a um utilizador, cada vez que o mesmo pretende aceder a páginas restritas, como por exemplo, a adição de entidades à plataforma CLAV é realizada uma chamada ao *middleware* de autenticação *isLoggedIn*, o qual verifica se existe um utilizador com sessão iniciada, bem como se o token do mesmo ainda não expirou, sendo esta lógica representada através do seguinte pseudo código.

Algorithm 5 Pseudo código da função de middleware *isLoggedIn*.

```
1: function ISLOGGEDIN(req, res, next)
2:     if req.isAuthenticated() then
3:         result ← jwt.verify(session.token, jwt.secret)
4:         if result ≠ expired then
5:             return next()
6:         else
7:             return err
```

Embora esta implementação do *JSON Web Token* estivesse 100% funcional, a mesma foi abandonada da gestão de utilizadores por uma alternativa similar, o armazenamento de sessões ativas diretamente na base de dados em *MongoDB*, sendo apenas utilizado o JWT na autenticação de pedidos à API.

4.3.1.2 *Armazenamento em MongoDB*

O armazenamento de sessões em *MongoDB* permite uma maior flexibilidade comparativamente com o armazenamento local.

Em primeiro lugar é possível recorrer ao paralelismo da aplicação via *nginx* ou outras soluções similares, pois cada *thread* pode correr independentemente umas da outras e a comunicação entre processos paralelos passa a ser inexistente. Para validar sessões cada *thread* faz uma *query* à base de dados, verificando se a sessão do utilizador é válida.

Permite também que na ocorrência de uma falha súbita do sistema, as informações relativas às sessões não se percam devido a estarem armazenadas em base de dados. Assim sendo, após o *boot* do sistema, os utilizadores continuam com as sessões ativas, algo que não aconteceria se fosse utilizado armazenamento local e validação via JWT.

Um possível problema desta solução seria que iríamos delegar o armazenamento das sessões para o lado do servidor, ou seja, se por ventura existissem milhões de sessões armazenadas, iria ser ocupado um tamanho substancial de memória em disco. Para evitar tal problema, é utilizado o módulo ***connect-mongo***[13] que permite a auto remoção de sessões expiradas, bem como a atribuição de um tempo máximo após o qual a sessão expira.

Para atingir este objetivo, apenas é necessário adicionar os campos abaixo mencionados à sessão do *ExpressJS*.

```
app.use(session({
    ...
    autoRemove: 'interval',
    autoRemoveInterval: 15, //minutes
    store: new MongoStore({
        url: dataBases.userDB,
        ttl: 1800 //seconds
    })
}));
```

Esta configuração do módulo ***connect-mongo*** faz com que a cada 15 minutos seja invocada a função de remoção de sessões expiradas, sendo que cada sessão apenas é válida durante 1800 segundos, ou seja, 30 minutos.

Em suma, o armazenamento de sessões em base de dados possui todas as qualidades da validação via *JSON Web Token*, oferecendo uma maior flexibilidade e parametrização, sem impacto na performance do servidor.

13 https://github.com/jdesboeufs/connect-mongo

4.3.1.3 *Nova técnica de validação através de JSON Web Token*

Devido à separação do frontend e backend em aplicações distintas, explicada na secção 4, a existência de uma sessão em base de dados deixa de ser possível. Isto deve-se ao facto do utilizador ter de iniciar sessão no servidor responsável pelo frontend da aplicação, sendo os dados introduzidos enviados para o backend, seguido da resposta do mesmo para o frontend (explicado na figura 32).

Devido ao envio da informação para outro servidor, esta informação e posterior sessão são armazenadas no servidor do backend e não do frontend. Caso a separação do frontend e backend não tivesse ocorrido, ambos correriam no mesmo servidor e este problema não existiria.

Para tal, foi necessário implementar um novo método de validação de sessões, fortemente baseado na autenticação em tokens discutida na secção 4.3.1.1.

A função de login foi alterada de modo a enviar os seguintes parâmetros:

- **Token JWT**

  Em vez de guardar toda a informação do utilizador como o algoritmo 4 foi decidido que este iria apenas conter o ID do utilizador, sendo realiza uma query à base de dados quando é necessário obter informações sobre o mesmo (o token criado possui 8h de validade).

  Este comportamento pode ser exemplificado pelo algoritmo 6.

Algorithm 6 Pseudo código da atribuição de um JWT à sessão.

```
1: function LOGIN(user)
2:     token ← jwt.sign(id : user.¡d, jwt.secret, {
3:         expiresIn : 8h
4:     }
5:     name ← user.name
6:     entidade ← user.entidade
7:     res.send(token, name, entidade)
```

- **Nome do utilizador.**
- **Entidade do utilizador.**

Ao receber o token JWT, o nome e a entidade do backend, estes são guardados em local storage através do *Vuex Store*. Sempre que uma página é acedida, é verificado se o token ainda não expirou de modo a terminar automaticamente as sessões.

Com esta nova técnica de validação e armazenamento de sessões através de tokens JWT é possível separar as aplicações em servidores distintos, mantendo também a sessão iniciada em caso de falha de servidor.

4.4 AUTENTICAÇÃO DE PEDIDOS À API DE DADOS

Nesta secção iremos explorar a implementação da autenticação de pedidos realizados à API de dados descrita na secção 3.3.

A API de dados da plataforma CLAV foi desenvolvida com o intuito de fornecer uma API pública, sem autenticação, e outra carente de autenticação, a ser utilizada por fornecedores de serviços que queiram desenvolver aplicações ou outros serviços que requerem a obtenção de dados da plataforma.

Assim sendo, nesta secção iremos analisar a página de registo de uma chave API para os fornecedores de serviços, bem como a renovação da mesma, e como esta autenticação é processada no backend.

4.4.1 *Registo*

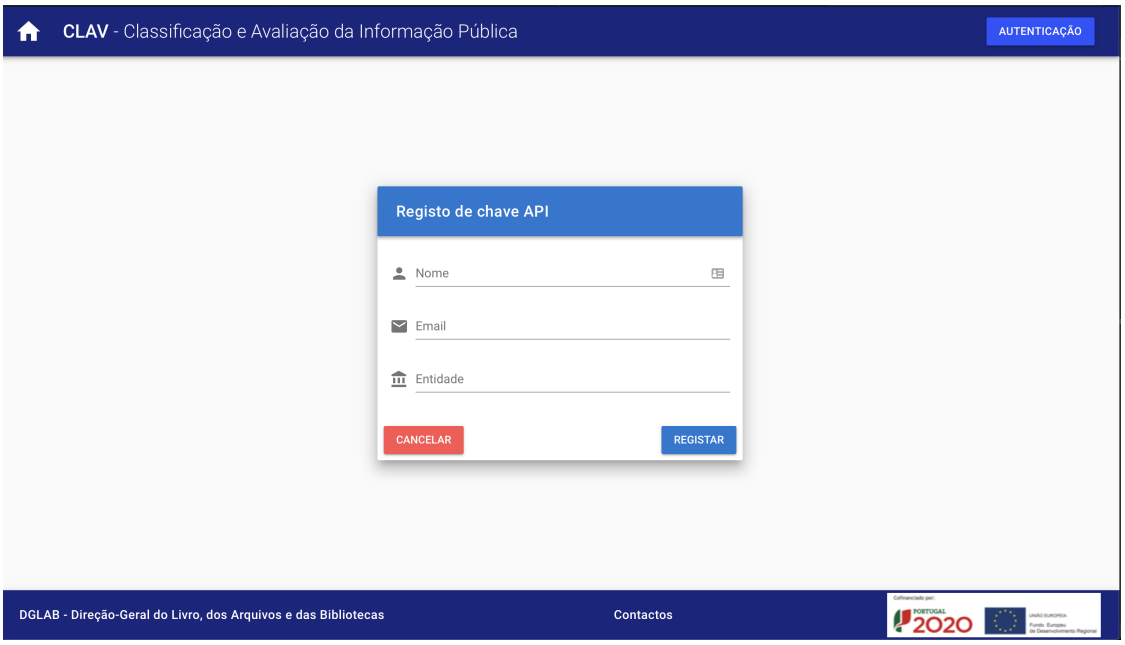


Figura 49: Página de registo de uma chave API na plataforma CLAV.

O registo de uma chave API por parte de um fornecedor de serviços requer o preenchimento da seguinte informação:

1. Nome.
2. Email (este tem de ser válido de modo a permitir a renovação da chave API).
3. Entidade (informação da entidade que está a requisitar a chave API).

Após submeter o pedido de registo de uma chave API é enviado para o email especificado a informação relativa à chave API emitida para a entidade/fornecedor de serviços em questão.

Figura 50: Email de registo contendo uma chave API da plataforma CLAV.

Como podemos verificar na figura 50, a chave API possui uma validade de 30 dias, após a qual é necessário renovar a mesma de modo a continuar a utilizar a API. Além disso, é enviada a chave em formato de um token JWT, bem como a maneira correta de enviar o token para processamento pelo backend a cada pedido realizado.

4.4.2 *Renovação*

Como explicado na secção 4.4.1, a chave API apenas possui uma validade de 30 dias. Esta foi implementada de modo a diminuir potenciais abusos ou pedidos excessivos por parte de um fornecedor de serviços, bem como estudar a utilização da chave API, identificando a origem dos pedidos que mais tráfego geram, entre outras medidas.

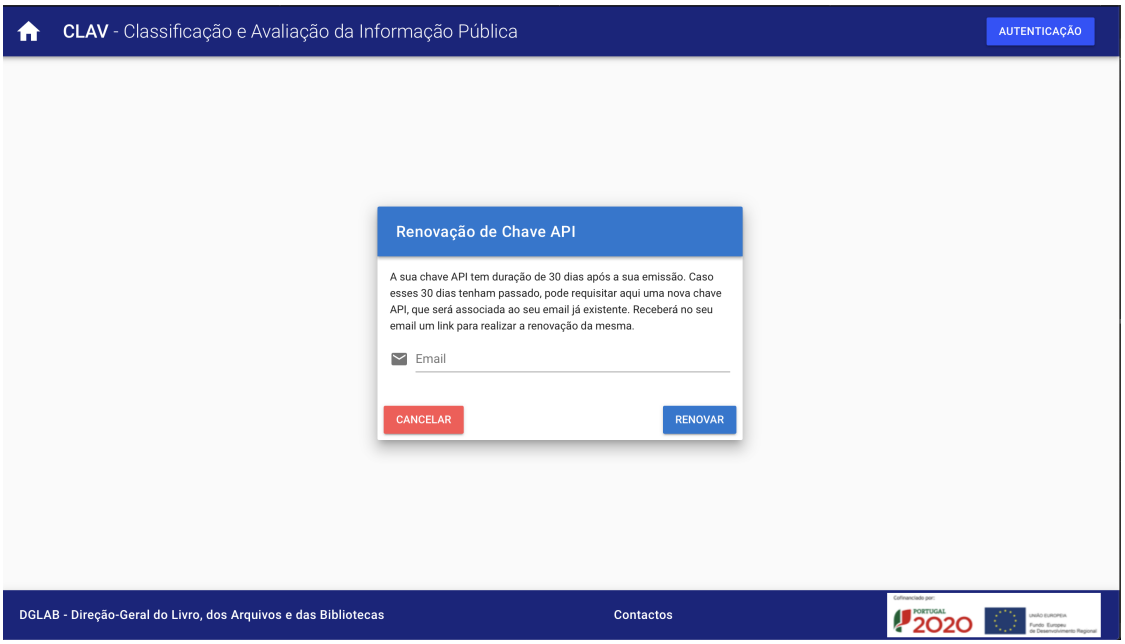


Figura 51: Página de renovação de uma chave API na plataforma CLAV.

De modo a renovar a chave API apenas é necessário introduzir o email utilizado para o registo da mesma, podendo ocorrer um de dois cenários:

- **Sucesso**

  Ocorre quando o utilizador insere um email associado a uma chave API.

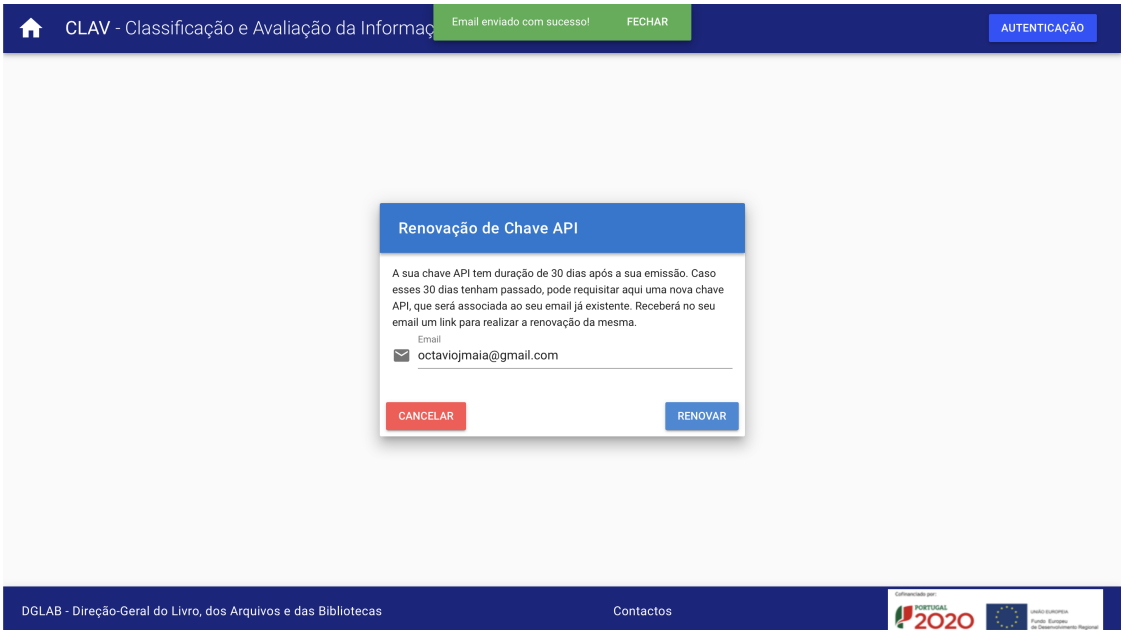


Figura 52: Mensagem de sucesso após fornecer um email correto para renovação de uma chave API na plataforma CLAV.

Após ser introduzido um email correto é enviado um email contendo um URL para renovação da chave API em causa.

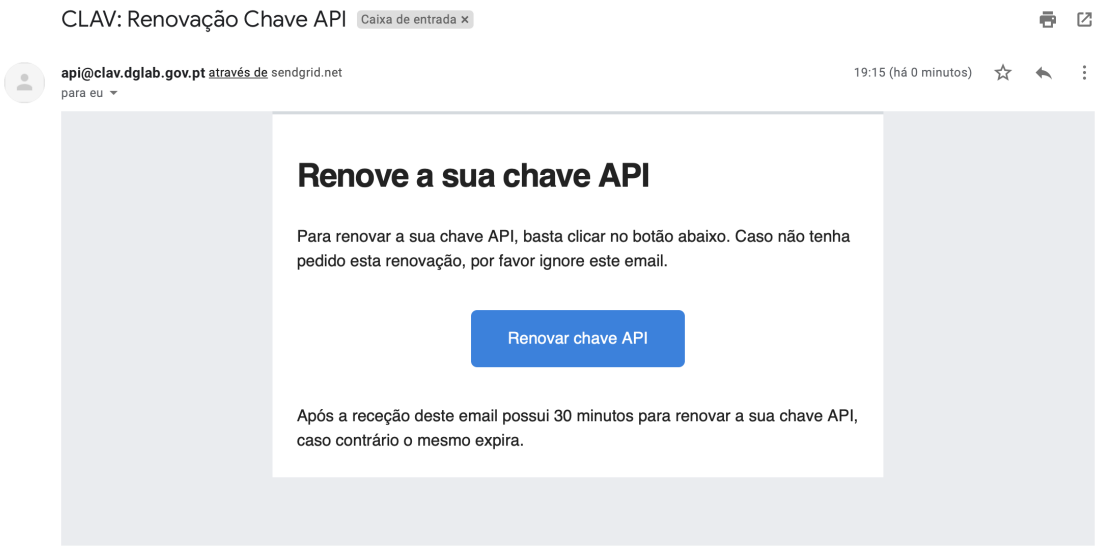


Figura 53: Email de renovação contendo a nova chave API da plataforma CLAV.

Após recepção do email, basta aceder ao URL presente no email e será enviada uma nova chave API, sendo a chave anterior desativada.

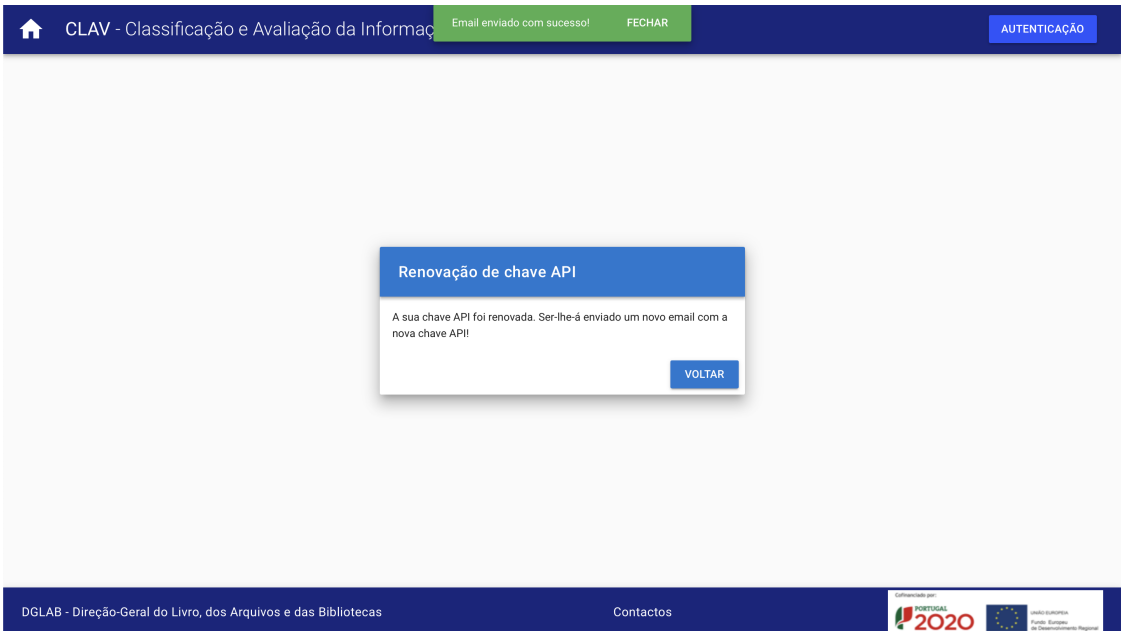


Figura 54: Mensagem de sucesso após clicar no link de renovação de uma chave API na plataforma CLAV.

Figura 55: Email de renovação contendo a nova chave API da plataforma CLAV.

- **Insucesso**

  Ocorre quando o utilizador fornece um email incorreto, ou seja, não associado a uma chave API.

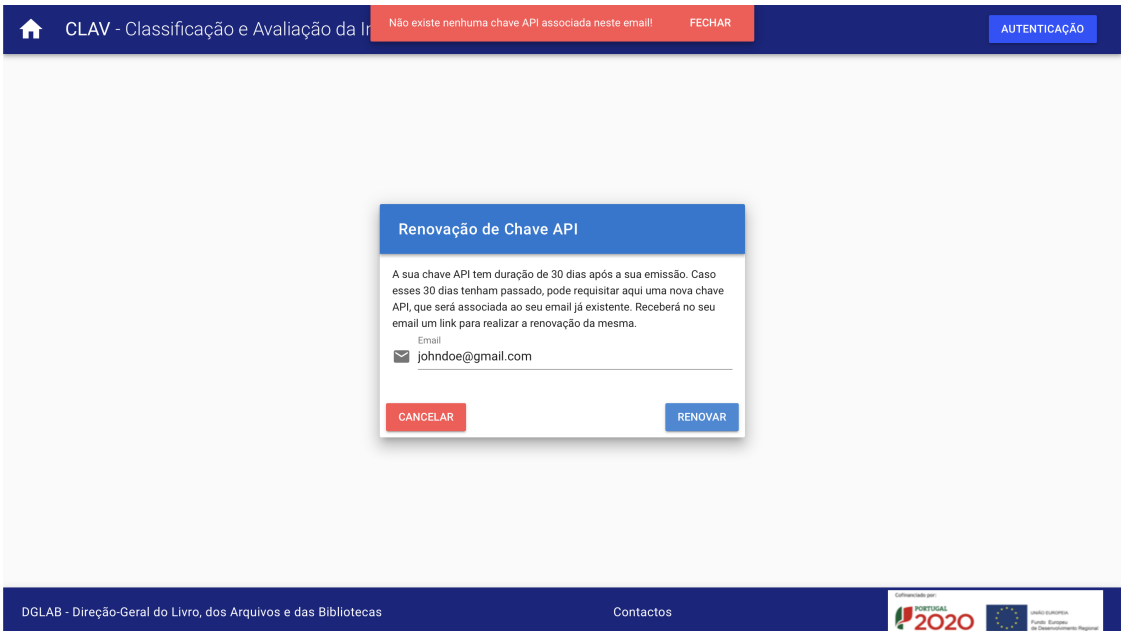


Figura 56: Mensagem de erro após fornecer um email incorreto para renovação de uma chave API na plataforma CLAV.

O comportamento da função de registo da chave API pode ser exemplificado pelo seguinte diagrama de sequência:

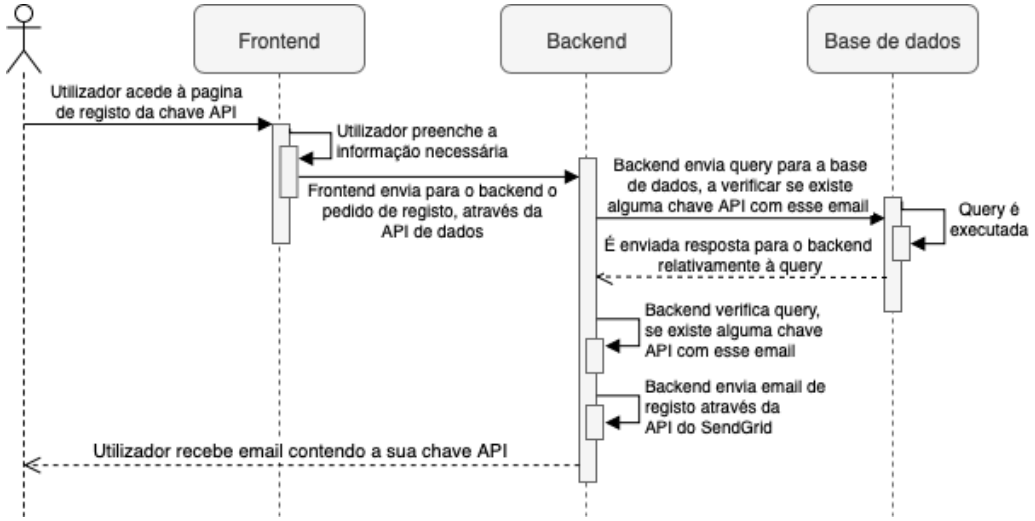


Figura 57: Diagrama de sequência relativo ao processo de registo de uma chave API na plataforma CLAV.

O comportamento da função de renovação da chave API pode ser exemplificado pelo seguinte diagrama de sequência:

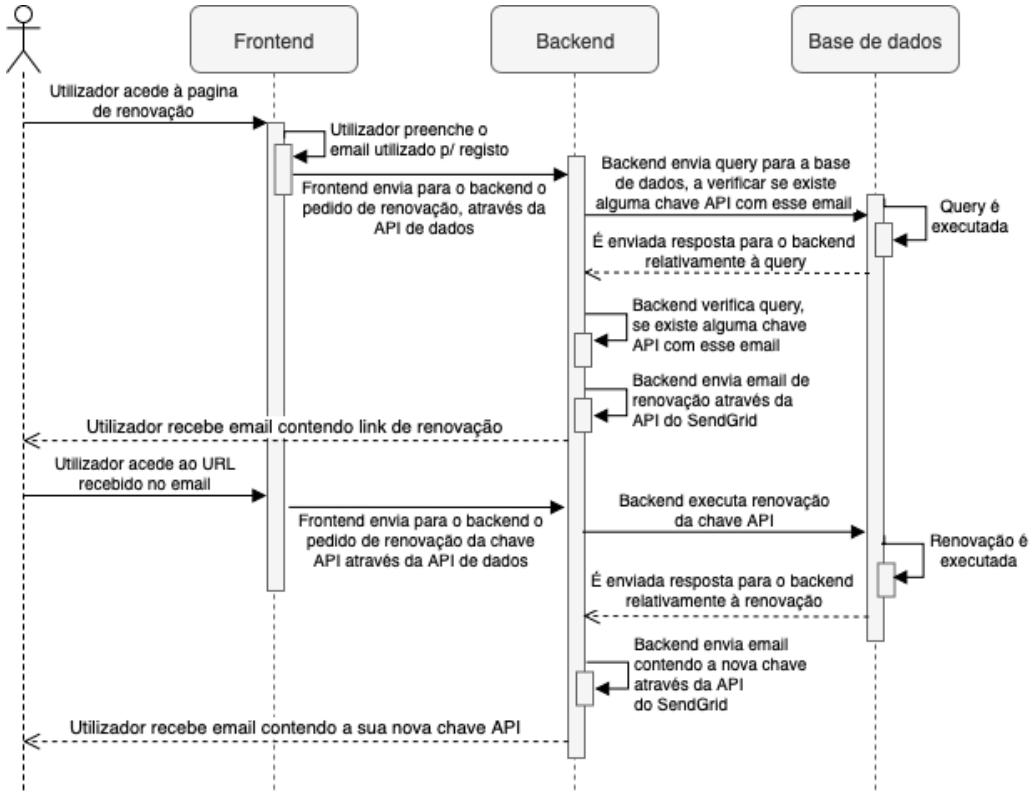


Figura 58: Diagrama de sequência relativo ao processo de renovação de uma chave API na plataforma CLAV.

4.5 GESTÃO DE CHAVES API

Devido ao elevado número de chaves API que irão ser emitidas pela plataforma CLAV (explicado na secção 4.4), foi necessária proceder à criação de uma página capaz de gerir as chaves emitidas.

Para tal foi desenvolvida a seguinte página de gestão, onde é possível ver todas as chaves emitidas, sendo possível ordenar por chave, nome da entidade que requisitou a chave, entidade, contato, bem como o estado da chave e as suas datas de criação e expiração.

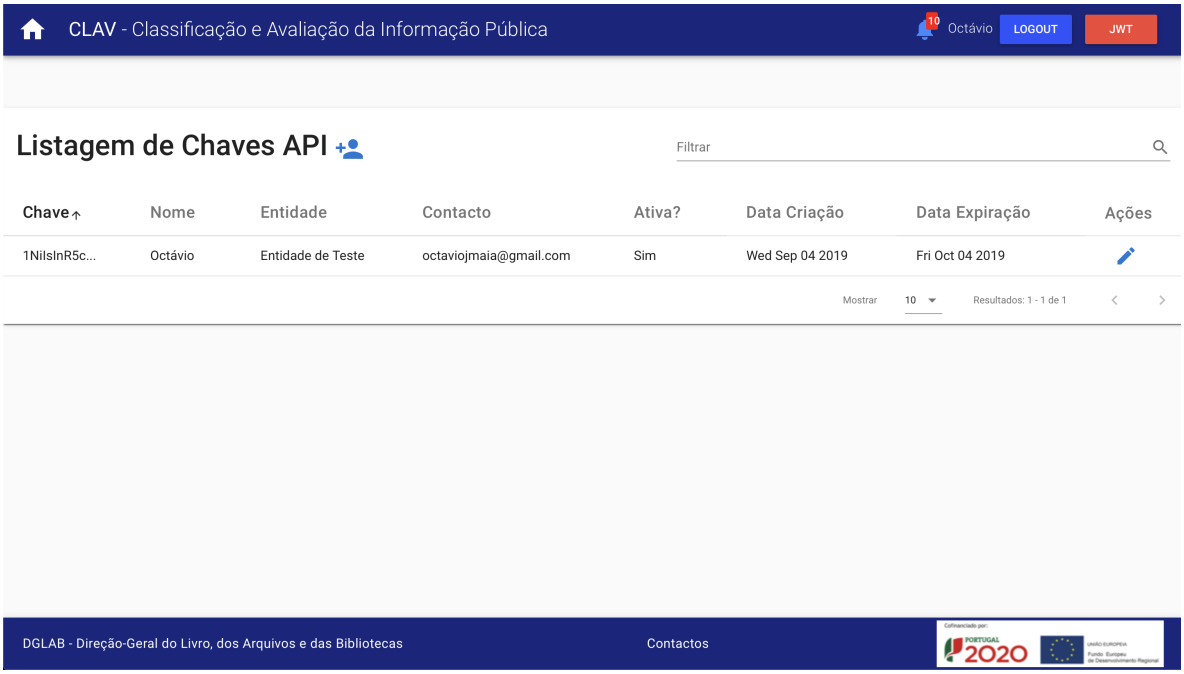


Figura 59: Página de listagem de todas as chaves API emitidas pela plataforma CLAV.

Além de listar as chaves, é necessário fornecer mecanismos capazes de editar a informação nela contida, nomeadamente o nome, entidade e contato de quem a requisitou. Além destes mecanismos de edição é necessário fornecer as seguintes funcionalidades:

- **Desativar chave API** A opção de desativar uma chave API é necessária devido a possíveis usos incorretos da mesma. Caso um utilizador seja suspeito de usar a chave de forma incorreta, esta poderá ser desativada para investigação, não podendo ser utilizada para pedidos à API até ser reativada.
- **Eliminar chave API** Caso se comprove que uma chave está de facto a ser utilizada de forma indevida, esta pode ser permanentemente eliminada da plataforma CLAV.

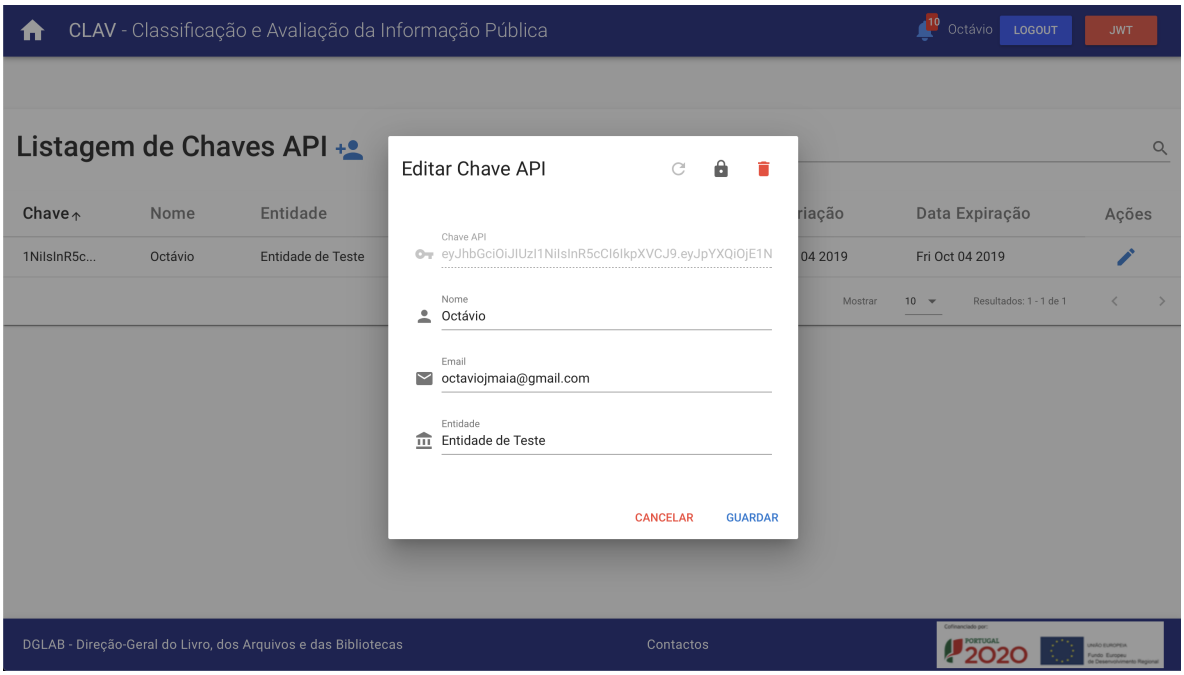


Figura 60: Página de edição de uma chave API emitida pela plataforma CLAV.

O comportamento da função de listagem e edição de chaves API pode ser exemplificado pelo seguinte diagrama de sequência:

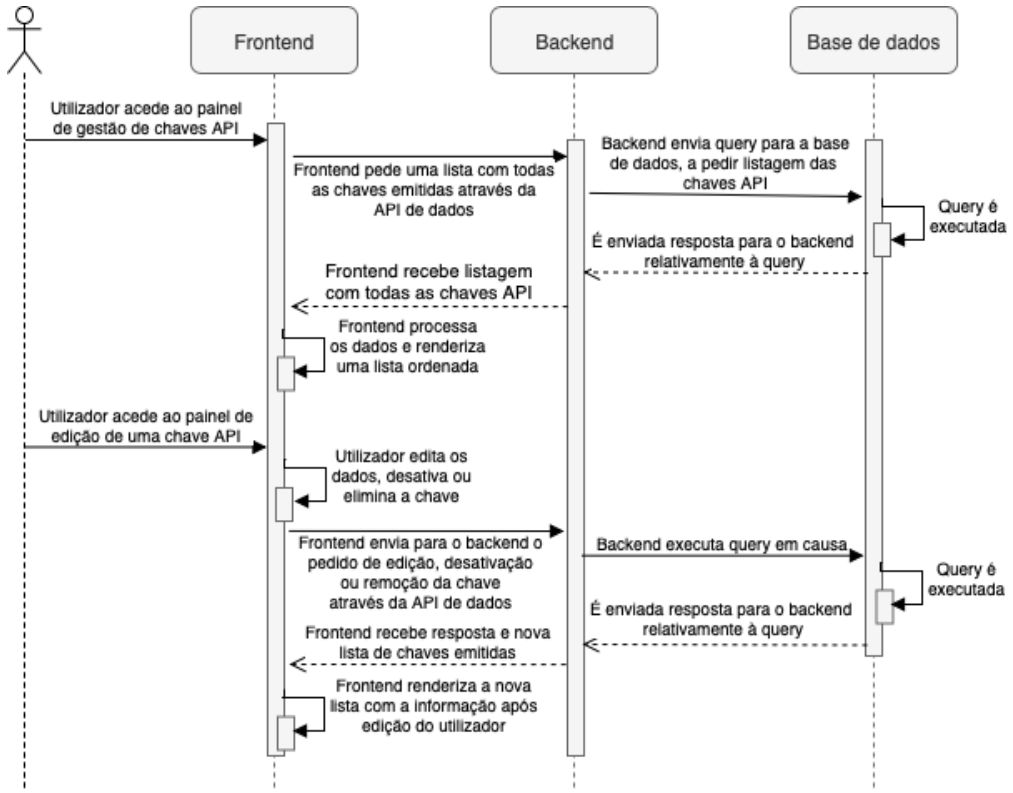


Figura 61: Diagrama de sequência relativo à gestão e edição de chaves API na plataforma CLAV.

4.6 AUTENTICAÇÃO BACKEND

De modo a proporcionar mecanismos capazes de lidar com os diversos tipos de utilizadores presentes na plataforma CLAV, foi implementada uma solução baseada em *middleware*. As funções de *middleware* são responsáveis pela verificação do nível de acesso do utilizador.

Para tal, foram desenvolvidas as seguintes funções, *checkLevel* e *isLevel*, sendo a primeira uma chamada à função auxiliar *isLevel*.

Algorithm 7 Pseudo código da função de middleware *checkLevel*.

```
1: function checkLevel7(user, next)
2:     return isLevel(7, user, next)
```

No pseudo código anterior podemos ver a implementação da função *checkLevel7*, cujo objetivo é verificar se o utilizador possui acesso de nível 7, ou superior.

Esta função recorre a uma chamada à função *isLevel*, sendo passado como parâmetro o nível de acesso a verificar.

Algorithm 8 Pseudo código da função de middleware *isLevel*.

```
1: function isLevel(clearance, user, next)
2:     if user.isAuthenticated then
3:         if user.level ≥ clearance then
4:             return next()
5:         else
6:             return redirect('back')
7:     else
8:         return redirect('login')
```

Com as estratégias de autenticação mencionadas anteriormente podemos assegurar uma correta implementação de níveis de acesso, não sendo possível a utilizadores aceder a conteúdo ao qual não possuem permissão.

O comportamento desta função de middleware pode ser descrita através do diagrama de sequência da figura 21.

4.7 MÉTRICA DA PLATAFORMA CLAV

Como foi especificado na secção 3.5, foi implementado um mecanismo capaz de reconhecer e processar os dados provenientes da utilização da API de dados.

Para tal, foi criada uma coleção em *MongoDB* contendo as diversas rotas API da plataforma CLAV, como por exemplo:

- **/api/users**
- **/api/tipologias**
- **/api/entidades**
- **/api/chaves**
- etc.

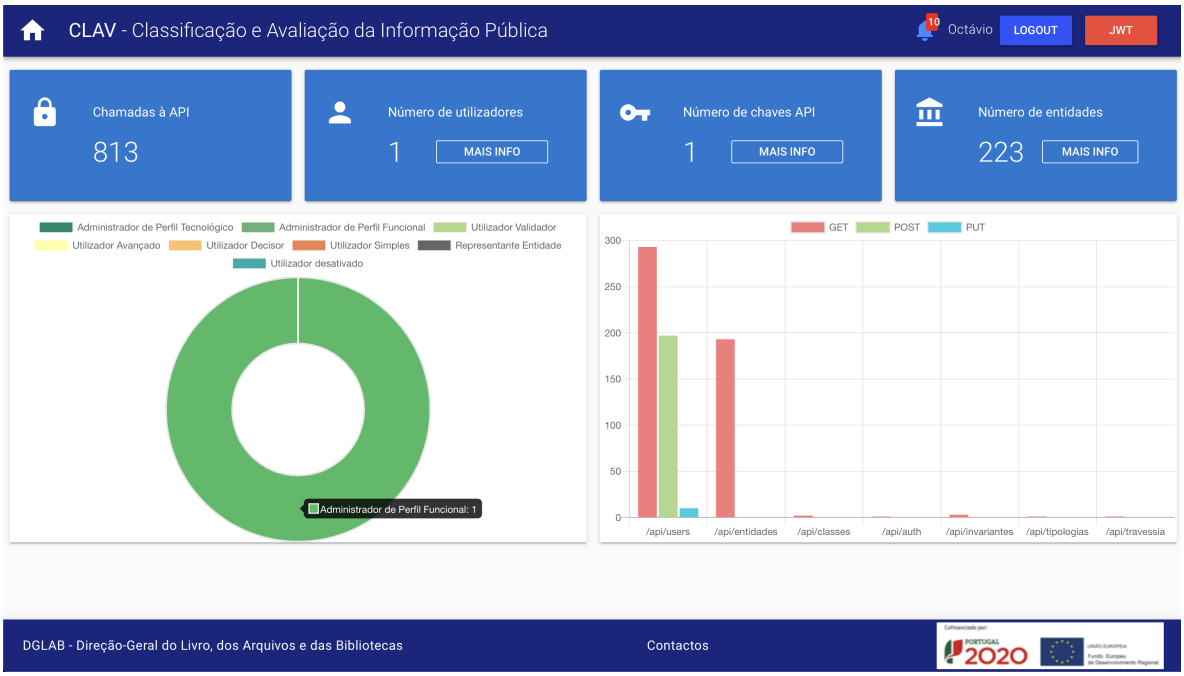


Figura 62: Página responsável pela visualização da métrica sobre a plataforma CLAV.

Cada uma destas rotas tem associado três contadores identificados de seguida:

1. **nCallsGet**

   Contador responsável por guardar o número de pedidos *GET* realizados sobre aquela API.

2. **nCallsPost**

   Contador responsável por guardar o número de pedidos *POST* realizados sobre aquela API.

3. **nCallsPut**

   Contador responsável por guardar o número de pedidos *PUT* realizados sobre aquela API.

O comportamento da função responsável por incrementar o número de chamadas à API pode ser exemplificado pelo seguinte diagrama de sequência:

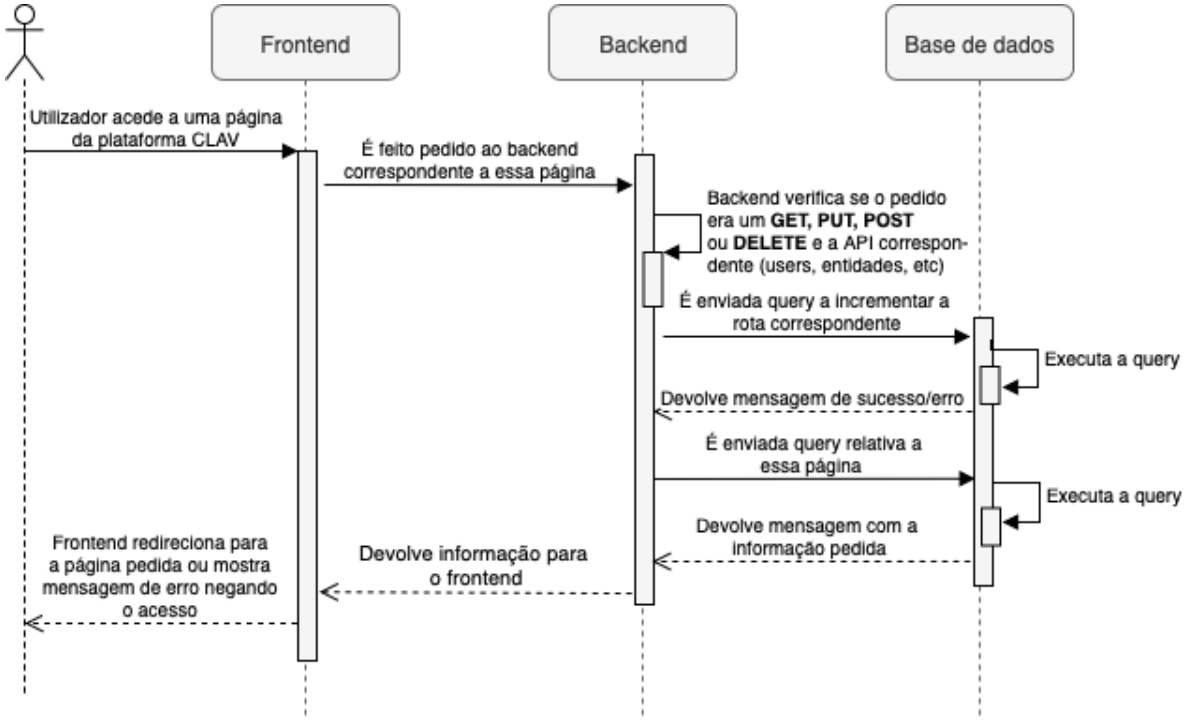


Figura 63: Diagrama de sequência relativo ao processo de recolha de métricas na plataforma CLAV.

4.8 Síntese

Após descrever a solução no capítulo anterior, este foi escrito com a mesma atenção ao detalhe, de modo a proporcionar uma explicação completa de toda a lógica aplicacional.

Este capítulo descreve também as duas vertentes presentes na plataforma CLAV, o Frontend e o Backend/API de dados, utilizando para tal diversos diagramas de sequência que ilustram toda a interação entre estas duas vertentes em grande detalhe. Além dos diagramas previamente descritos, são também utilizadas diversas imagens da plataforma CLAV após implementação das soluções descritas no capítulo anterior.

Espera-se que esta documentação permita que qualquer futura alteração, ou duvida existente sobre a implementação da plataforma CLAV, possa ser facilmente solucionada.

5

CONCLUSÕES E TRABALHO FUTURO

Hoje em dia, com o contínuo crescimento de ataques informáticos, bem como fugas de informação sensível, é necessário cada vez mais reforçar os meios de proteção existentes, de modo a não só prevenir, mas evitar ao máximo que os mesmos se repitam.

Para tal, a escrita desta dissertação baseou-se na documentação dos diversos métodos de segurança e autenticação impostos no desenvolvimento do projeto CLAV.

Foram exploradas diversas noções de encriptação, bem como vulnerabilidades existentes e possíveis soluções às mesmas. Após explorada esta vertente foi realizada uma breve introdução ao *bcrypt* e ao seu funcionamento, bem como uma introdução a possíveis vulnerabilidades utilizando hardware especializado.

Após concluída esta etapa, foi realizada uma leitura sobre a ferramenta *Autenticação.Gov*, desenvolvida pela AMA. Esta ferramenta permite a autenticação em diversos serviços utilizando o Cartão de Cidadão, entre outras alternativas. Foi feita uma introdução ao standard SAML, sendo este necessário para a correta implementação dos serviços previamente mencionados, bem como a especificação de outros mecanismos necessários, como a utilização de *HTTPS* e *SSL*, a criação de chaves *RSA* e a geração de um certificado *X509* com as respetivas cadeias de autenticação.

Posteriormente foi introduzida a noção de *JSON Web Token* e o papel crucial que desempenham para o âmbito do projeto, sendo este a autenticação de chamadas à API de dados. Foram também desenvolvidos diversos métodos de proteção da mesma, sendo estes todos baseados no *JSON Web Token* previamente descrito.

Foram também desenvolvidas diversos mecanismos capazes de realizar a gestão de utilizadores, bem como a sua edição e desativação caso necessário. Este conceito foi expandido para as chaves API emitidas pela plataforma CLAV, sendo possível obter uma listagem das mesmas, bem como a sua edição e remoção da plataforma.

Após esta etapa foi desenvolvido um mecanismo capaz de relatar a métrica da plataforma CLAV, ou seja, o número de utilizadores, chaves API emitidas, quanti-

dade de acessos a cada rota da plataforma, bem como o número de *GET*, *POST*, etc., que cada rota da API disponibilizada é sujeita.

Por fim foi concluída a integração do *Autenticação.Gov*, ou seja, o registo e autenticação através de Cartão de Cidadão na plataforma CLAV, sendo todo este processo descrito nesta dissertação de forma extensa, de modo a servir de base para qualquer futura implementação desta mesma vertente de autenticação.

Todo este processo previamente descrito foi documentado através de diagramas de fluxo e de sequência, estando estes presentes nos capítulos 3 e 4, respetivamente.

Por conseguinte, considera-se que todos os objetivos propostos para esta dissertação foram cumpridos, sendo a implementação de autenticação através de Chave Móvel Digital um desafio interessante para uma futura implementação.

BIBLIOGRAFIA

[1] O cartão de cidadão - autenticação.gov. Consult. 21 Fev. 2019; Disponível em https://www.autenticacao.gov.pt/o-cartao-de-cidadao.

[2] Operações de modernização e capacitação da administração pública. Compete2020 [online] http://www.poci-compete2020.pt/concursos/detalhe/Aviso_02_SAMA2020_2016. Consult. 23 Set. 2018.

[3] "papel zero": Administração pública tem de reduzir pelo menos 20%. O Jornal Económico. [online] https://jornaleconomico.sapo.pt/noticias/papel-zero-administracao-publica-reduzir-pelo-menos-20-11847.

[4] Saml 2.0 integration with identityserver4. Consult. 14 Jan. 2019; Disponível em https://www.identityserver.com/articles/saml-20-integration-with-identityserver4/.

[5] Carlisle Adams and Steve Lloyd. Understanding pki: Concepts, standards, and deployment considerations. Boston, MA, USA, 2002. Addison-Wesley Longman Publishing Co., Inc. ISBN 0672323915.

[6] *Autenticação.Gov: Fornecer de Autenticação da Administração Pública Portuguesa*. Administração Pública Portuguesa, 1.5.1 edition, 2018. Consult. 14 Jan. 2019; Disponível em https://www.autenticacao.gov.pt/cc-documentacao.

[7] Scott Contini. Method to protect passwords in databases for web applications. *IACR Cryptology ePrint Archive*, 2015:387, 2015.

[8] Organisation for Economic Co-operation and Development. Making life easy for citizens and businesses in portugal: Administrative simplification and e-government. Technical report, OECD, 2009. ISBN 978-92-64-04788.

[9] John Hughes and Eve Maler. Security assertion markup language (saml) v2. 0 technical overview. *OASIS SSTC Working Draft sstc-saml-tech-overview-2.0-draft-08*, pages 29–38, 2005.

[10] Michael Jones, John Bradley, and Nat Sakimura. Json web token (jwt). Technical report, 2015.

[11] Alexandra Lourenço, Maria Rita Gago, Pedro Penteado, and José Carlos Ramalho. Plataforma m51-clav: o que há de novo? 2017.

[12] Alexandra Lourenço, José Carlos Ramalho, Maria Rita Gago, and Pedro Penteado. Transformação digital: novas políticas e procedimentos para a classificação e avaliação da informação. 2018.

[13] Vincent Lynch. Re-hashed: The difference between sha-1, sha-2 and sha-256 hash algorithms. Consult. 10 Jan. 2019; Disponível em https://www.thesslstore.com/blog/wp-content/uploads/2018/08/Hashing.png.

[14] Katja Malvoni, Designer Solar, and Josip Knezović. Are your passwords safe: Energy-efficient bcrypt cracking with low-cost parallel hardware. In *WOOT'14 8th Usenix Workshop on Offensive Technologies Proceedings 23rd USENIX Security Symposium*, 2014.

[15] Agência para a Modernização Administrativa. Estatísticas relativas ao uso da plataforma autenticação.gov. Consult. 14 Jan. 2019; Disponível em https://www.autenticacao.gov.pt/stats-autenticacaogov.

[16] E Peyrott. The jwt handbook. *Seattle, WA, United states*, 2016.

[17] Niels Provos and David Mazieres. A future-adaptable password scheme. In *USENIX Annual Technical Conference, FREENIX Track*, pages 81–91, 1999.

[18] P Sriramya and RA Karthika. Providing password security by salted password hashing using bcrypt algorithm. *ARPN Journal of Engineering and Applied Sciences*, 10(13), 2015.

[19] International Telecommunication Union. X.509 : Information technology - open systems interconnection - the directory: Public-key and attribute certificate frameworks. Consult. 18 Mar. 2019; Disponível em https://www.itu.int/rec/T-REC-X.509.

[20] Friedrich Wiemer and Ralf Zimmermann. High-speed implementation of bcrypt password search using special-purpose hardware. In *ReConFigurable Computing and FPGAs (ReConFig), 2014 International Conference on*, pages 1–6. IEEE, 2014.