

**ÉLIO CARVALHO FARIAS**

**GESTÃO DE IDENTIDADE  
PESSOAL COM OPENDID CONNECT  
E CARTÃO DE CIDADÃO**

**Orientador: Prof. Doutor. José Quintino Rogado**

**Universidade Lusófona de Humanidades e Tecnologias  
Escola de Comunicação, Arte e Tecnologias da Informação (ECATI)  
Departamento de Engenharia Informática e Sistemas de Informação (DEISI)**

**Lisboa**

**2018**



**ÉLIO CARVALHO FARIAS**

**GESTÃO DE IDENTIDADE  
PESSOAL COM OPENDID CONNECT  
E CARTÃO DE CIDADÃO**

Tese defendida em provas públicas na Universidade Lusófona de Humanidades e Tecnologias no dia 10 de Janeiro de 2018, perante o júri, nomeado pelo Despacho de Nomeação n.º: 161/2017, de 5 de Maio, com a seguinte composição:

- Presidente: Prof. Doutor. Rui Pedro Nobre Ribeiro
- Argente: Prof. Doutor. Pedro Hugo Queirós Alves
- Orientador: Prof. Doutor José Quintino Rogado

**Universidade Lusófona de Humanidades e Tecnologias  
Escola de Comunicação, Arte e Tecnologias da Informação (ECATI)  
Departamento de Engenharia Informática e Sistemas de Informação (DEISI)**

**Lisboa**

**2018**



## Agradecimentos

Este espaço é dedicado àqueles que de alguma forma, deram a sua contribuição para que esta dissertação fosse possível. A todos eles deixo aqui o meu agradecimento sincero.

Em especial aos meus familiares e companheira, por todo o apoio e confiança necessária para realizar esta dissertação.

Ao meu orientador, Professor Doutor José Quintino Rogado, pelo apoio e orientação disponibilizados na realização deste trabalho, conselhos e sugestões, além das palavras de ânimo que imprimia sempre que achava necessário.



## Resumo

Atualmente é um fato admitido que os sistemas informáticos estão cada vez mais presentes no dia-a-dia dos cidadãos, tendo assumido um papel fundamental nas suas vidas. O seu constante e rápido crescimento tem-se generalizado através do desenvolvimento de múltiplas aplicações disponíveis na Web ou destinadas a dispositivos móveis, que facilitam as suas tarefas diárias e estabelecem inclusivamente novos meios de interação social.

De forma a garantir a identidade e a privacidade dos seus utilizadores, a maioria destas aplicações requerem autenticação, sejam elas aplicações nativas dos sistemas operativos, aplicações móveis ou aplicações orientadas para a Web. Como tem sido amplamente divulgado [1], os métodos tradicionais de autenticação apresentam inúmeras fragilidades, quer nos requisitos de segurança com que são concebidos, quer pelo facto de obrigar os utilizadores a memorizar inúmeras credenciais, distintas para cada aplicação. No sentido de mitigar este problema, atualmente existem várias soluções que permitem aos utilizadores usar um único conjunto de credenciais, armazenadas num único Provedor de Identidade, que podem ser utilizadas para aceder a múltiplas plataformas, sem necessidade de autenticação adicional. Embora constitua uma comodidade indiscutível, esta abordagem tornam os utilizadores extremamente dependentes dos grandes 'players' da Internet, que geralmente fornecem os seus serviços em troca da exploração dos dados dos utilizadores, para fins comerciais.

Este trabalho tem por objetivo demonstrar, com base no conhecimento adquirido e numa prova de conceito original desenvolvida especificamente, que ainda existe evolução possível nesta área, integrando duas tecnologias atuais, o protocolo OpenID Connect e o Cartão de Cidadão Nacional que, conjugadas, permitem criar um método de autenticação pessoal forte e seguro, que pode ser completamente gerido pelo utilizador.

Palavras-chave: OpenID Connect, cartão de cidadão, autenticação.



## Abstract

It is currently a well-known fact that computer systems are increasingly present in citizens' daily life and have played a key role in their lives. Its constant and rapid growth has been generalized through the development of multiple applications available on the Web or intended for mobile devices, which facilitate their daily tasks and even establish new means of social interaction.

In order to ensure the identity and privacy of your users, most of these applications require authentication, whether they are native applications of operating systems, mobile applications or web applications. As has been widely publicized [1] traditional authentication methods present numerous weaknesses both in the security requirements with which they are designed and in requiring users to store countless different credentials for each application. In order to mitigate this problem, several solutions currently exist that allow users to use a single set of credentials, stored in a single Identity Provider, that can be used to access multiple platforms without the need for additional authentication. However, while it is an indisputable convenience, this approach makes users extremely dependent on the major Internet players, who generally provide their services in return for the exploitation of user data for commercial purposes.

This paper aims to demonstrate, based on the knowledge acquired and a specifically developed original proof of concept, that there is still possible evolution in this area, integrating two current technologies, the OpenID Connect protocol and the 'Cartão Cidadão' that, together, allow to create a strong and secure personal authentication method that can be completely managed by the user.

**Keywords:** OpenID Connect, citizen card, authentication.



## Abreviaturas e Símbolos

AC – Autoridade Certificadora  
AES- Advanced Encryption Standard  
API – Application Programming Interface  
BD – Base de Dados  
C# – Linguagem de programação  
CA – Certification Authority  
CC – Cartão de Cidadão  
CSR – Certificate Signing Request  
CAP – Chip Authentication Program  
DAC – Discreminatory Acess Control  
DES- Data Encryption Standard  
DLL – Dynamic Link Library  
JVM – Java Virutal Machine  
EEPROM – Electrically Erasable Programmable Read Only Memory  
FIDO – Fast Identity Online Alliance  
HMAC – Based One-time Passwpord  
HASH – asdasd  
HMAC – Hash-based Message Authentication Code  
HTTP – Hyper Text Transfer Protocol  
HTTPS - Hyper Text Transfer Protocol Secure  
ICAO – International Civil Aviation Organization  
IdP – Identity Provider  
IEC – International Electrotechnical Commission  
ISO – International Organization for Standardization  
JSON – Java Scrit Object Notation  
JWE – Json Web Encryption  
JWT – Json Web Token  
KDF - Key Derivation Function  
LDAP – Lightweigth Directory Access Protocol  
MAC – Mandatory Access Control  
MD5 – Message Digest Algorithm 5  
MFA – Autenticação Multifator  
MRTD – Machine Readable Travel Documents  
NFC – Near Field Commiunication  
OCSP – Online Certificate Status Protocol

OTP – One Time Password

PIN – Personal Identification Number

PKCS – Public Key Cryptography Standard

PKI – Public Key Infrastructure

QR Code – Quick response code

RBAC - Role Based Access Control

REST – Representational State Transfer

RP – Relying Party

SHA – Secure Hashing Algorithm

SHA1 – Secure Hash Algorithmn 1

SOAP – Simple Object Acess Protocol

SP – Service Provider

SSH – Secure Shell

SSL – Secure Socket Layer

SSO – Single Sign-On

TOTP – Time-Based One-time Password Algorithm

U2F – Universal Second Factor

UML – Unified Modeling Language

URL – Uniform Resource Locator

WS Federation – Web Service Federation

XRI – Extensible Resource Identifier

# Índice Geral

1. Introdução .....	15
2. Motivação e Objetivos .....	17
3. Estado da Arte .....	19
3.1 Segurança .....	19
3.1.1 Conceitos de Segurança .....	20
3.1.2 Mecanismos de Segurança .....	20
3.1.2.1. Segurança Física.....	20
3.1.2.2. Segurança Lógica.....	21
3.2 Autenticação.....	21
3.2.1 Fatores de Autenticação.....	21
3.2.1.1. Fatores de conhecimento:.....	21
3.2.1.2. Fatores de propriedade.....	21
3.2.1.3. Fatores de inherência .....	22
3.2.2 Tipos de Autenticação.....	22
3.2.2.1. Autenticação de fator único.....	22
3.2.2.2. Autenticação de dois fatores.....	22
3.2.2.3. Autenticação de vários fatores .....	22
3.2.2.4. Autenticação forte .....	23
3.2.3 Tecnologia de autenticação.....	23
3.2.3.1. oAuth .....	23
3.2.3.2. Auth0 .....	25
3.2.3.3. Google Authenticator .....	30
3.2.3.4. Fido .....	32
3.2.3.5. OpenID Connect .....	35
3.2.3.6. Cartão Cidadão.....	37
3.2.3.7. Criptografia .....	41
3.2.3.8. Certificados Digitais .....	43
3.2.3.9. Tokens de acesso.....	46
4. Trabalhos Relacionados.....	51
4.1 Smart Cards e Open ID .....	51
4.2 CC e acesso a plataformas Cloud .....	51

5. Solução .....	53
5.1 Arquitetura da solução .....	53
5.2 Funcionamento da Solução .....	55
5.3 Protótipo .....	57
6. Implementação.....	69
6.1 Provedor de Identidades.....	69
6.1.1 Autenticação CC .....	69
6.1.2 Pesquisa do certificado .....	71
6.1.3 Assinatura do segredo .....	71
6.1.4 Validação da assinatura .....	72
6.1.5 Envio do identificador .....	73
6.1.6 Registo CC.....	73
6.1.7 Moodle .....	74
7. Resultados Alcançados.....	75
8. Conclusão .....	77
9. Referências .....	79

## Índice Figuras

Figura 1: Fluxo de autenticação OAuth 2.0 (reproduzido a partir de oauth.net).....	24
Figura 2: Empresa e versões do OAuth (reproduzido a partir de oauth.net).....	25
Figura 3: Implementação Auth0 (reproduzido a partir de auth0.com) .....	25
Figura 4: Fluxo de uma implementação Auth0 (reproduzido a partir de auth0.com)26	
Figura 5: Interface de autenticação Auth0 (reproduzido a partir de auth0.com).....	27
Figura 6: Confirmação de login Auth0 (reproduzido a partir de auth0.com) .....	28
Figura 7: Código de verificação Auth0 (reproduzido a partir de auth0.com) .....	29
Figura 8: Configuração passwordless Auth0 (reproduzido a partir de auth0.com) ..	29
Figura 9: App Google Authenticator (reproduzido a partir de play.google.com) .....	31
Figura 10: Pseudocode do Google Authenticator (reproduzido a partir de google.com) .....	32
Figura 11: Formas de autenticação FIDO (reproduzido a partir de fidoalliance.org)	33
Figura 12: Fluxo de autenticação FIDO (reproduzido a partir de fidoalliance.org) ..	34
Figura 13: Fluxo de login FIDO (reproduzido a partir de fidoalliance.org) .....	35
Figura 14: Fluxo OpenID Connect (reproduzido a partir de openid.net).....	37
Figura 15: Cartão de cidadão Português (reproduzido a partir de autenticacao.gov.pt) .....	38
Figura 16: Arquitetura sistemas com Cartão de Cidadão (reproduzido a partir de autenticacao.gov.pt) .....	40
Figura 17: Criptografia com chaves simétricas (reproduzido a partir de en.wikipedia.org).....	42
Figura 18: Criptografia com chave pública e desencriptar com chave privada (reproduzido a partir de en.wikipedia.org) .....	43
Figura 19: Formato certificado (reproduzido a partir de en.wikipedia.org) .....	45
Figura 20: Criação de um certificado digital (reproduzido a partir de en.wikipedia.org) .....	45
Figura 21: Cabeçalho de um JWT (reproduzido a partir de en.wikipedia.org).....	47
Figura 22: Corpo de um JWT (reproduzido a partir de en.wikipedia.org) .....	47
Figura 23: Assinatura de um JWT (reproduzido a partir de en.wikipedia.org).....	47
Figura 24: Exemplo do formato (reproduzido a partir de en.wikipedia.org) .....	48
Figura 25: Formato de um JWT (reproduzido a partir de en.wikipedia.org) .....	48
Figura 26: Fluxo de comunicação JWT (reproduzido a partir de en.wikipedia.org) .	49
Figura 27: Arquitetura da Aplicação .....	52
Figura 28: Diagrama da arquitetura do protótipo .....	54
Figura 29: Diagrama de sequência de autenticação com CC .....	55

Figura 30: Diagrama de sequência do protótipo sem CC .....	56
Figura 31: Página inicial protótipo (IdP).....	57
Figura 32: Gestão de utilizadores (IdP) .....	57
Figura 33: Novo utilizador (IdP) .....	58
Figura 34: Registo de utilizadores com CC (IdP).....	58
Figura 35: Pin do CC (IdP) .....	59
Figura 36: Aviso utilizador (IdP) .....	59
Figura 37: Utilizadores registados (IdP) .....	60
Figura 38: Página inicial (RP).....	61
Figura 39: Página login (IdP).....	62
Figura 40: Pedido pin ao utilizador (IdP) .....	62
Figura 41: Página de dados do utilizador (RP) .....	63
Figura 42: LogOut (IdP).....	63
Figura 43: Login Moodle.....	65
Figura 44: Login com CC .....	66
Figura 45: Solicitação de autorização.....	66
Figura 46: Edição de perfil Moodle .....	67
Figura 47: Função de autenticação (c#) .....	70
Figura 48: Função de autenticação CC (c#) .....	71
Figura 49: Função pesquisa de certificado digital (c#).....	71
Figura 50: Função de assinatura (c#).....	72
Figura 51: Função de validação (c#) .....	72
Figura 52: Função de pesquisa do utilizador na BD (c#) .....	73
Figura 53: Função de criação de um novo utilizador (c#) .....	74

## 1. Introdução

Atualmente, é um facto admitido que os sistemas informáticos estão cada vez mais presentes no dia-a-dia dos cidadãos, tendo assumido um papel fundamental nas suas vidas. Com o seu constante e rápido crescimento, são desenvolvidas aplicações que facilitam, a cada minuto, as suas tarefas diárias.

A maior parte das aplicações requerem autenticação, sejam elas aplicações nativas dos sistemas operativos ou aplicações orientadas para a Web. Assim, colocam-se várias questões, sendo as principais relacionadas com a segurança e a gestão da autenticação. Atualmente, existem várias formas de autenticação possíveis, sendo a mais utilizada, por ser a mais antiga e simples, a autenticação baseada em ‘user’ e ‘password’, como por exemplo o ‘Challenge Handshake Authentication Protocol’ [2].

Mas este método tradicional de autenticação tem grandes fragilidades do ponto de vista da segurança. Apesar de terem sido feitos vários avanços no que respeita à complexidade na criação de ‘passwords’, este continua a ser o método mais vulnerável a ataques. Para além deste facto, a utilização de múltiplas ‘passwords’ pode tornar-se insustentável para o utilizador, uma vez que cada sistema dita as suas próprias regras relativas à sua criação. A dificuldade desta simples ação pode aumentar exponencialmente, tendo em conta o grande número de aplicações que um utilizador geralmente utiliza no seu dia-a-dia.

A vulnerabilidade do método de autenticação baseado em ‘user’ e ‘password’ faz com que seja relativamente fácil aceder a dados privados do utilizador e, até mesmo, furtar a sua identidade. Com a evolução das aplicações do quotidiano, a privacidade está a tornar-se um luxo, que será muito difícil manter com métodos de autenticação frágeis, como o acima referido.

Todas estas questões, que surgem todos os dias, estão a alterar a forma como a tecnologia e os protocolos de comunicação são encarados. Atualmente, a autenticação num sistema evoluiu do simples método baseado no ‘user’ e ‘password’ para esquemas muito mais elaborados, como por exemplo a utilização de fatores biométricos, tais como a leitura da retina, a leitura de impressão digital, a utilização de hardware específico de tipo ‘Smart Cards’ ou, até mesmo, utilizando sistemas de delegação de autenticação. Alguns destes sistemas ou conceitos de autenticação já existiam, mas passaram a ser usados em muitos sistemas da Web para resolver algumas das questões que se colocam no âmbito da autenticação segura.

Com este panorama de instabilidade e fragilidade a nível da segurança, torna-se claro que novos protocolos e ‘standards’ têm de ser desenvolvidos, de forma a permitir a evolução e o aparecimento de novos mecanismos de autenticação. Exemplo disso, são os métodos de autenticação baseados em propagação de confiança, que nos permitem, por

exemplo, utilizar as credenciais de uma conta Google para aceder a um recurso, noutra sistema, com base no protocolo oAuth [3].

Esta dissertação tem por objetivo demonstrar, com base no conhecimento adquirido, que ainda existe evolução possível nesta área, ao integrar duas tecnologias que, conjugadas, permitem criar um método de autenticação pessoal forte e seguro. A primeira destas tecnologias é o protocolo OpenID Connect [5], atualmente já utilizada pelos grandes ‘players’ do mercado da identidade, tais como a Google, o Facebook ou a Microsoft. Esta tecnologia será conjugada com a utilização de um ‘Smart Card’, recorrendo neste caso, ao Cartão de Cidadão Português [4].

A demonstração desta afirmação será feita através da apresentação de um protótipo que permite substituir por completo a tradicional autenticação (‘user’ e ‘password’) por um mecanismo de autenticação mais seguro. Utilizando a infraestrutura do OpenID, foi desenvolvido um módulo cliente e também uma integração com a plataforma Moodle, para aceitar autenticação baseada no estabelecimento de confiança, propagada sob a forma de um ‘token’, fornecido por um outro módulo igualmente desenvolvido, um provedor de identidade (IdP), que implementa o protocolo de autenticação OpenID Connect. Neste IdP, a prova de identidade é realizada com base no ‘Smart Card’, sendo apenas pedido ao utilizador o PIN de proteção da respetiva chave privada.

## 2. Motivação e Objetivos

Apesar de assistirmos a uma evolução muito acentuada do modelo de autenticação dos sistemas da Internet, impulsionada sobretudo pelos grandes *players* de serviços de Gestão de identidade [1](Google, Facebook, Microsoft, etc...), na nossa opinião existem ainda formas de os melhorar. Este modelo, muito focado na tecnologia Open ID Connect [5], permite grandes melhorias na forma como os utilizadores se autenticam. O 'Open ID Connect é um protocolo de autenticação de Single Sign-On baseado no 'OAuth' 2.0 [6], que permite aos utilizadores acederem seguramente a múltiplas aplicações com uma única autenticação garantida por um provedor de identidade da sua escolha. Este protocolo permite às aplicações, que o suportam, abdicar da necessidade de autenticar os seus utilizadores, devendo para tal estabelecer uma relação de confiança com os provedores de identidades.

Este modelo de autenticação garante melhores condições de segurança e privacidade para os utilizadores, visto que as aplicações a que estes acedem podem impor uma autenticação forte, sem a necessidade de conhecer as credenciais dos utilizadores, sendo estas unicamente armazenadas nos gestores de identidade onde os utilizadores têm as suas contas. A contrapartida deste modelo é aumentar a relação de dependência dos utilizadores com os grandes 'players' referidos, sendo os seus serviços geralmente remunerados através de publicidade dirigida com base na informação que recolhem sobre os perfis e hábitos dos utilizadores.

Por outro lado, em Portugal, assim como noutras países da Comunidade Europeia, foi desenvolvido um cartão de identificação para todos os cidadãos, denominado Cartão de Cidadão (CC), que é dotado de tecnologia capaz de garantir uma autenticação forte em aplicações que o suportem. De acordo com o portal do CC, o projeto cresceu no sentido de dinamizar e modernizar a administração pública Portuguesa, através de um documento único de cidadania, de natureza tecnológica, um 'smart card', que permite ao cidadão português realizar duas operações essenciais: (1) identificar-se nos serviços públicos, que estejam aptos para usar esta tecnologia, e (2) assinar documentos eletrónicos, garantido a autenticidade de emissão pelo utilizador correspondente (não repudiável).

Embora estas duas tecnologias não sejam novas em si, a sua utilização permite criar um sistema de autenticação forte, mas completamente controlado pelo utilizador, constituindo, nesse sentido, uma abordagem original.

Com a importância que os temas da Autenticação e da Gestão de Identidade revestem no panorama atual da computação na Web, o objetivo proposto para esta dissertação de mestrado é o de fornecer um novo sistema de autenticação que, utilizando uma tecnologia atualmente considerada como 'standard' (OpenID Connect), de facto liberte

o utilizador da dependência dos grandes provedores de Identidade. Simultaneamente, é introduzida apenas uma componente de autenticação pessoal, unicamente por si controlada (CC), mantendo o modelo genérico da delegação de identidade e com um fator duplo de autenticação baseada num ‘smart card’ e um pin (CC).

Com este objetivo, o primeiro passo foi o de realizar um estudo sobre todo o ecossistema e software disponibilizado pelo estado português, que permite tirar partido de todas as funcionalidades do CC. O seguinte passo, foi estudar o protocolo OpenID Connect de modo a introduzir a componente de autenticação do CC. Em termos tecnológicos, o ponto-chave consistiu em identificar onde e quando colocar a autenticação através do CC, garantindo que o fluxo do protocolo OpenID Connect não é interrompido nem corrompido, mantendo seguro todo o processo de autenticação.

Nesse sentido, foi implementado um protótipo com base no código disponibilizado pela própria instituição OpenID, recorrendo às linguagens ‘C#’ e ‘HTML5’, onde foram realizadas as alterações necessárias para integrar a comunicação com o CC. O sistema desenvolvido garante todo o processo de comunicação, realizado entre um provedor de identidade (Identity Provider) e uma aplicação cliente (Relying Party), usando OpenID Connect e uma autenticação forte sem recurso a ‘user’ e ‘password’, no provedor de identidades.

### 3. Estado da Arte

A investigação realizada no Estado da Arte, que serve de suporte ao protótipo desenvolvido, no âmbito desta tese, faz referência às várias soluções de autenticação existentes no mercado.

Como já abordado, este protótipo não se trata de uma nova solução, mas sim de unir duas soluções que, na ótica da investigação e conhecimento adquirido, foram as duas tecnologias que mais garantias deram para a criação deste protótipo de autenticação segura, sem recurso a um ‘user’ e ‘password’.

A análise centra-se nas soluções que o mercado nos oferece, criadas por grandes empresas, que apostam na segurança e na evolução dos sistemas de autenticação. A facilidade com que uma aplicação pode desenvolver um mecanismo, recorrendo às tecnologias existentes, torna a fiabilidade do protótipo desenvolvido uma resolução a pensar em futuras aplicações e soluções.

#### 3.1 Segurança

Na atualidade, com a evolução tecnológica a que temos assistido, a segurança é um dos grandes problemas no que diz respeito aos sistemas informáticos. O que há cinco anos atrás era impensável, hoje é uma realidade.

Cada vez mais, a informação que possuímos circula livremente na internet, por vários fatores e com um avassalador crescimento. Devido a estes acontecimentos é muito importante ter em conta a gestão da informação e a sua segurança. Nesta linha de pensamento, a segurança da informação define-se como a disciplina responsável pela integridade e proteção de um conjunto de informações guardadas num sistema informático, de um determinado indivíduo ou organização. Informação entende-se como qualquer conteúdo ou dado, com valor para uma determinada organização ou indivíduo.

A segurança de informação tem como principais conceitos, a confidencialidade, a integridade e a disponibilidade. Quando falamos disto referimo-nos à informação digital ou não digital, onde ambas podem aplicar estes conceitos da mesma forma.

Com o ritmo acelerado de crescimento, a que estamos a assistir, todos estes temas sofreram a necessidade de uma padronização. Atualmente, o conceito de segurança de informação está padronizado pelas normas ISSO/IEC[7].

### 3.1.1 Conceitos de Segurança

**Confidencialidade** – conceito que limita o acesso à informação, apenas tem acesso quem é autorizado pelo proprietário da informação.

**Integridade** – conceito que garante todas as características originais da informação, estabelecidas pelo proprietário. Este conceito garante todo o ciclo de vida dessa informação, as suas alterações e que estas não são alteradas por terceiro.

**Disponibilidade** – conceito que assegura que a informação está sempre disponível para todos os utilizadores, que tenham acesso.

**Autenticidade** – conceito que valida a autenticidade da informação, garantido que esta não é manipulada e é proveniente da fonte anunciada.

**Privacidade** – conceito que controla os acessos à informação, quem viu, o que viu, quem realizou determinado processo, controlado por local e horário.

**Não Repúdio** – conceito que impossibilita a negação da autoria de uma transação realizada.

Todos os conceitos descritos são utilizados pelo protótipo apresentado, juntamente com esta tese de mestrado, tirando proveito para conseguir criar uma autenticação segura. São conceitos básicos de segurança da informação, mas que estão presentes em todos os sistemas.

### 3.1.2 Mecanismos de Segurança

De modo, a garantir que os anteriores conceitos possam ser implementados corretamente, foram desenvolvidos mecanismos de segurança, que se dividem em dois grupos: os físicos e os lógicos. Ambos necessitam de ser combinados entre si.

#### 3.1.2.1. Segurança Física

A segurança física baseia-se em controlos físicos, barreiras colocadas para limitar o acesso direto à informação, ou às infraestruturas que guardam a informação. Quando falamos dos controlos físicos, estes também trabalham na área das ameaças físicas, catástrofes naturais, acessos físicos indevidos à informação e falhas nos equipamentos. Todos os temas

que podem causar problemas nos equipamentos ou nas infraestrutura, onde é guardada a informação.

### **3.1.2.2. Segurança Lógica**

A segurança lógica são os controlos realizados a nível da informação guardada, limitando ou obstruído o acesso à mesma. Este tipo de controlos, são mecanismos baseados em softwares, que controlam todo o tipo de ameaças ou acesso indevido à informação, através de uma rede. A segurança lógica é a forma como um sistema protege a informação de forma eletrónica, tendo em conta todo o tipo de ameaças informáticas e erros aplicacionais.

## **3.2 Autenticação**

A autenticação, como definição, baseia-se no ato de assegurar algo ou alguém como autêntico. Baseado nesta definição, a autenticação reclama a autoria ou a veridicidade de alguma coisa, confirmado a origem de um objeto ou pessoa, neste caso, está relacionado com uma verificação de identidade. A verificação de uma identidade pode englobar uma validação através dos documentos de identificação [8].

A autenticação de um utilizador, em um sistema, assenta em três categorias: algo que o utilizador ‘sabe’; algo que o utilizador ‘tem’; e algo que o utilizador ‘é’. Os fatores de autenticação abrangem uma série de elementos utilizados na autenticação ou na validação, antes de ser autorizado o acesso de uma identidade, uma transação ou de uma assinatura de um documento.

A evolução determina que uma autenticação correta é aquela onde pelo menos dois, dos três fatores de autenticação, são verificados.

### **3.2.1 Fatores de Autenticação**

#### **3.2.1.1. Fatores de conhecimento:**

É algo que o utilizador conhece ('sabe'), por exemplo, 'user' e 'password', frase secreta, pin, resposta de desafio ou questão de segurança.

#### **3.2.1.2. Fatores de propriedade**

É algo que o utilizador possui ('tem'), por exemplo, cartão de identificação, 'token' de segurança, 'token' de 'hardware' e 'token' de software.

### **3.2.1.3. Fatores de inherência**

É algo que o utilizador diz ser ('é'), por exemplo, ADN, voz, face, impressão digital, retina e sinais bioelétricos.

## **3.2.2 Tipos de Autenticação**

Os tipos de autenticação são combinações dos fatores de autenticação, de uma ou mais categorias de fatores, para oferecer um elevado nível de segurança ao utilizador.

### **3.2.2.1. Autenticação de fator único**

A autenticação de fator único é o nível mais baixo, no que diz respeito à segurança. Baseado apenas em um único componente, de uma das três categorias de fatores, é aplicado para autenticar ou validar um utilizador. Logicamente o uso de uma autenticação de único fator, não é aconselhável, pois não oferece proteção suficiente aos utilizadores, que ficam facilmente vulneráveis ao uso indevido ou ilegal da sua informação pessoal.

### **3.2.2.2. Autenticação de dois fatores**

A autenticação de dois fatores é um sistema de autenticação mais robusto, que utiliza fatores de autenticação de duas das categorias de fatores. Obrigando que um utilizador conheça algo ('sabe') e possua algo ('tem'). Dois exemplos deste tipo de autenticação, dos quais os portugueses estão bastante familiarizados, são os cartões bancários e os CC. São elementos completamente individuais (algo que o utilizador 'tem') e são combinados com um pin (algo que o utilizador 'sabe'), para que possam ser utilizados nas variadíssimas operações que estes permitem efetuar.

### **3.2.2.3. Autenticação de vários fatores**

A autenticação de vários fatores é a combinação de pelo menos duas categorias. Em uma autenticação deste género, o utilizador é submetido a um mecanismo, que utiliza fatores separados, para cada uma das três categorias de fatores. Podemos considerar a autenticação de vários fatores de nível bastante elevado. A utilização de vários fatores permite utilizar-los de forma aleatória, criando diversas combinações, o que faz que seja bastante difícil que os dados do utilizador sejam usados indevidamente ou expostos.

### **3.2.2.4. Autenticação forte**

A autenticação forte, que muitas vezes é confundida, por uma autenticação de vários fatores ou dois fatores, baseia-se numa autenticação em camadas, com dois ou mais autenticadores para identificar uma identidade ou um recetor de informação. O procedimento de uma autenticação forte é sustentado por dois ou mais fatores utilizados, independentemente, entre si e, pelo menos, um desses fatores não deve ser reutilizado e replicado.

### **3.2.3 Tecnologia de autenticação**

Atualmente existe muita tecnologia e algoritmos que dão suporte a todos os tipos de autenticação, abordados anteriormente. Com o crescimento da informação pessoal e sensível, a circular na internet, a ameaça de riscos que os utilizadores correm é eminente. Devido a isso, ocorre a grande aposta no desenvolvimento destas tecnologias.

#### **3.2.3.1. oAuth**

##### **O que é?**

O oAuth é um protocolo de segurança, ‘open source’[9], que permite uma aplicação aceder aos dados protegidos de um utilizador. O oAuth permite que um utilizador autorize um terceiro a usar os seus dados, sem que este partilhe a sua ‘password’. O utilizador pode criar ou realizar login numa aplicação através de uma conta Google, Facebook, Microsoft, entre outras, sem ter que partilhar o seu ‘user’ e ‘password’.

##### **Como funciona?**

Num modelo de autenticação convencional, cliente-servidor, o cliente solicita um acesso protegido ao servidor de autenticação, sendo este o dono das credenciais dos utilizadores.

No oAuth, o fluxo foi definido com a introdução de uma camada de autorização, que permite separar o cliente dos recursos. O cliente solicita o acesso aos recursos, controlados pelo proprietário, num servidor que emite credenciais, diferentes daquelas que as do recurso proprietário. O cliente, em vez de ter acesso às credenciais do proprietário, obtém um ‘token’ de acesso, com tempo de vida e outros atributos de acesso. Os ‘tokens’ são emitidos para os clientes, através de um servidor de autorizações, sempre com a aprovação do proprietário do recurso. Por fim, o cliente usa o ‘token’ atribuído, para ter acesso aos recursos.

**Proprietário do recurso:** Algo proprietário de um recurso, quando este é uma pessoa chamamos de utilizador final.

**Servidor de recursos:** Servidor que guarda os recursos protegidos, aceita e responde a solicitações para o acesso a recursos, através de ‘tokens’ de acesso.

**Cliente:** Aplicação que solicita acesso a recursos protegidos.

**Servidor de autorização:** Servidor que emite os ‘tokens’ de acesso aos clientes, após uma autenticação com sucesso do proprietário do recurso.

## Fluxo



Figura 1: Fluxo de autenticação oAuth 2.0 (reproduzido a partir de oauth.net)

- (A) O cliente solicita autorização ao proprietário do recurso, o pedido de autorização pode ser feito diretamente ao proprietário ou, indiretamente, através de um servidor de autorizações.
- (B) O cliente recebe uma ‘grant’ autorização, isto é, uma credencial que representa a autorização do proprietário do recurso. O tipo de ‘grant’ depende do método utilizado pelo cliente, para solicitar autorização e que tipos suporta o servidor de autorizações.
- (C) O cliente solicita um ‘token’ de acesso ao servidor de autorizações, apresentando o ‘grant’.
- (D) O servidor de autorizações autentica o cliente, valida o ‘grant’ e emite um ‘token’ de acesso.
- (E) O cliente solicita o acesso aos recursos protegidos e apresenta o ‘token’ de acesso.
- (F) O servidor de recursos valida o ‘token’, e no caso de este ser válido, permite o acesso.

## Evolução

O protocolo oAuth iniciou-se na versão 1.0, mas apenas foi publicado um documento informativo. Em finais de 2007, inícios de 2008, a ‘Internet Engineering Task Force’ (IETF) decidiu adotar a especificação para permitir uma discussão mais ampla e uma normalização. Em 2010, foi publicada a primeira versão do oAuth, já normalizado e como um ‘standard’. Nesse próprio ano, em Maio, inicia-se o desenvolvimento da versão 2.0 do protocolo, uma vez que a integração da primeira versão era complicada. Foi decidido criar uma nova versão para simplificar a integração.

O OAuth 2.0 não é compatível com a versão anterior, de qualquer forma, as duas versões podem coexistir e as implementações podem suportar ambas as versões. Atualmente o foco está na versão 2.0, que tem vindo a ser adotado por muitos serviços na internet, de grandes empresas.

- Google (v2.0)
- Yahoo (v1.0a)
- Twitter (v1.0a e v2.0 <sup>1</sup> ))
- Github (v2.0)
- Microsoft (v2.0)
- Foursquare (v2.0)
- Salesforce (v2.0)
- Facebook (v2.0)

Figura 2: Empresa e versões do oAuth (reproduzido a partir de oauth.net)

### 3.2.3.2. Auth0

#### O que é?

O Auth0 (zero) é um serviço que gere toda a autenticação de uma aplicação, baseado no protocolo OAuth 2.0. O Auth0 foi desenvolvido para suportar qualquer tipo de aplicação web e para facilitar o desenvolvimento dessas aplicações.



Figura 3: Implementação Auth0 (reproduzido a partir de auth0.com)

## Como funciona?

O Auth0 permite integrar qualquer aplicação (desenvolvida em qualquer linguagem de programação), que seja capaz de utilizar ‘http’ [10]. O protocolo padrão utilizado nas integrações entre aplicações e o Auth0 é o OpenID Connect, por ser uma tecnologia leve e ter uma grande facilidade de integração. Como o OAuth 2.0 permite diferentes tipos de fluxos de autorização, dependendo da aplicação.

O fluxo utilizado para aplicações web é baseado num código de autorização, este é redirecionado para o Auth0 (`/authorize`), para que os utilizadores possam introduzir as suas credenciais. Após o utilizador realizar uma autenticação correta, será novamente redirecionado para o ‘URL’ [11] de retorno, passando um parâmetro `authorization_code` na cadeia de consulta do ‘Callback url’. Este parâmetro de autorização pode ser trocado por um `id_token`, realizando o pedido ao `/oauth/token`. De apontar que o `id_token` é um ‘JSON Web token’ (JWT) [12], que contém vários atributos, incluindo os dados do utilizador.

## Fluxo

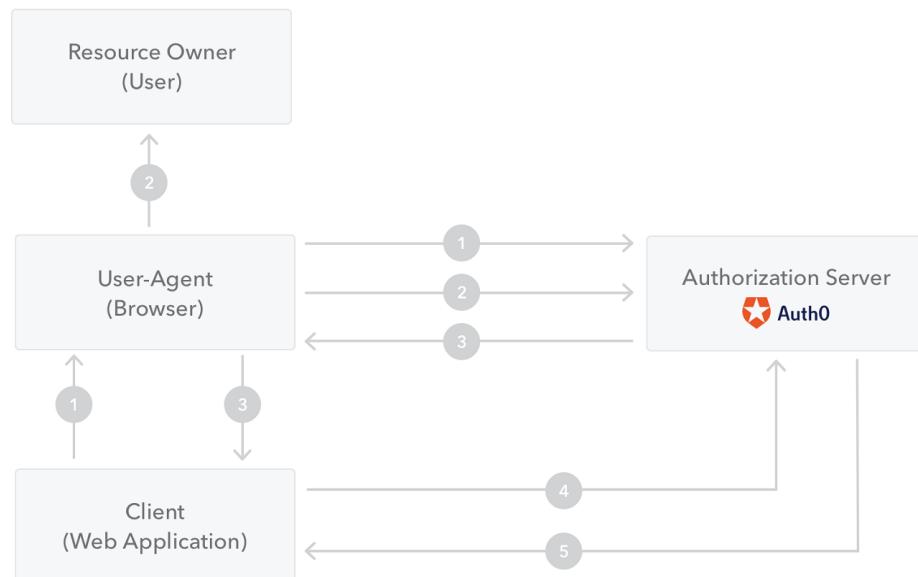


Figura 4: Fluxo de uma implementação Auth0 (reproduzido a partir de auth0.com)

1. O cliente inicia o fluxo e o utilizador é redirecionado para o servidor de autenticação;
2. O utilizador autentica-se;
3. O servidor de autenticação redireciona o utilizador para o `redirect_uri` com um `authorization_code` na cadeia de consulta;
4. O cliente envia o `authorization_code`, o `redirect_uri` e o ‘Client Id/Client Secret’ para o servidor de autenticação;

5. O servidor de autenticação valida esta informação e retorna um `id_token`.

### Possibilidades de autenticação

**Single Sign-On (SSO):** O SSO ocorre quando um utilizador autentica-se a um cliente e, automaticamente, está autenticado em outros clientes aos quais tem acesso, independentemente da plataforma, tecnologia, ou domínio que o utilizador esteja a utilizar. O SSO, normalmente, utiliza um serviço central que gere o SSO entre vários clientes.

No Auth0 o SSO aplica-se da seguinte forma:

1. A aplicação redireciona o utilizador para a Auth0, para efetuar o login;
2. Auth0 irá verificar se existe um ‘cookie’ [13] do SSO existente;
3. Na primeira vez que o utilizador visita a página de autenticação e não existe nenhum cookie SSO, a página é apresentada de acordo com a configuração, por exemplo, um bloco com ‘user’ e ‘password’ e, um outro, com alguns provedores de identidade;

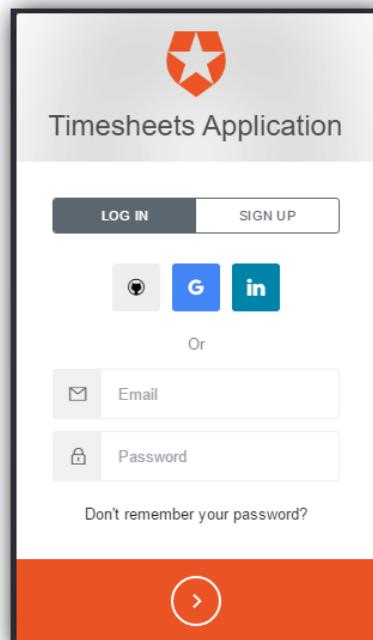


Figura 5: Interface de autenticação Auth0 (reproduzido a partir de auth0.com)

4. Assim que o utilizador se autentica, o Auth0 cria um ‘cookie’ de SSO;
5. Auth0 redireciona o utilizador novamente para a aplicação, enviando um `id_token` com a identidade do utilizador.

Quando os utilizadores voltam às aplicações, o Auth0 comporta-se da seguinte forma:

1. A aplicação redireciona o utilizador para a Auth0, para fazer login;
2. Auth0 verifica se já existe um ‘cookie’ de SSO;
3. A página de login é substituída por uma página que apresenta a identidade do utilizador, que pode, simplesmente, confirmar se quer fazer o login com a mesma conta ou utilizar outra;

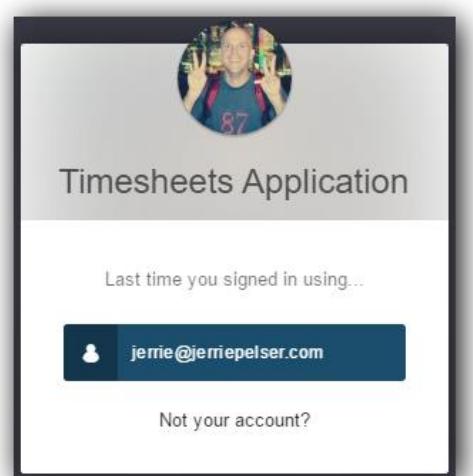


Figura 6: Confirmação de login Auth0 (reproduzido a partir de auth0.com)

4. O Auth0, em caso de necessidade, atualiza o ‘cookie’ de SSO;
5. Auth0 volta a redirecionar o utilizador para aplicação, enviando um `id_token` com a identidade do utilizador.

**Autenticação multi-fator (MFA):** é um método que consiste na verificação da identidade do utilizador, exigindo mais que um fator de autenticação.

O Auth0 permite a autenticação multi-fator, exigindo ao utilizador a seguinte informação:

- Conhecimento : Algo que o utilizador sabe (exemplo: ‘password ’);
- Posse : Algo que o utilizador tem (exemplo: ‘smartphone’);
- Herança : Algo que o utilizador é (exemplo: impressão digital ou a leitura da retina).

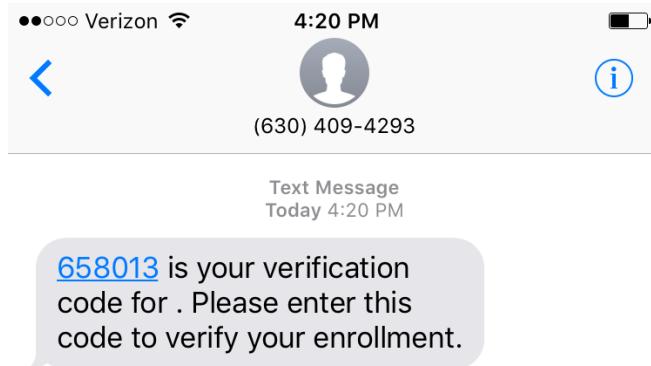


Figura 7: Código de verificação Auth0 (reproduzido a partir de auth0.com)

**Autenticação sem ‘password’:** Com o auth0 os utilizadores podem autenticar-se usando o método ‘passwordless’ [14] que consiste numa autenticação sem ‘passwords’. É possível utilizar este método de três formas distintas: SMS, email ou impressão digital.

Na autenticação por impressão digital, o utilizador é obrigado a usar a sua impressão digital para se autenticar, provando ser quem diz ser. Nas outras duas formas são enviados códigos de acesso, para que o utilizador possa provar a sua identidade.

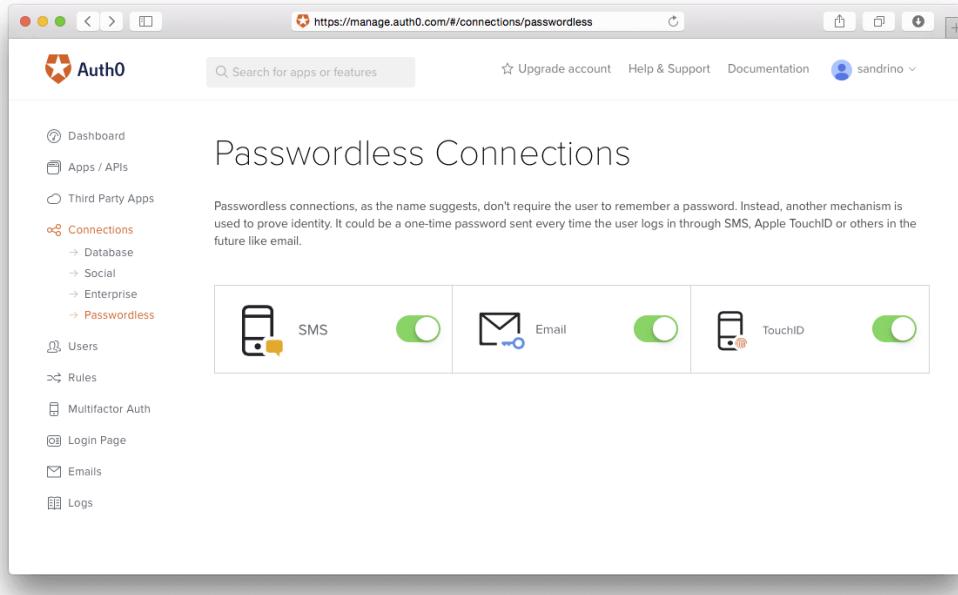


Figura 8: Configuração passwordless Auth0 (reproduzido a partir de auth0.com)

### Protocolos:

- oAuth 2.0 – permite que as aplicações de terceiros obtenham acesso limitado aos recursos do utilizador;

- OpenID Connect – para que as aplicações de terceiros validem a identidade do utilizador e obtenham informação básica do perfil do utilizador;
- SAML [15]– para autenticação e autorização entre duas entidades: um provedor de serviços e um provedor de identidade. O provedor de serviços concorda em confiar no provedor de identidade, para autenticar os utilizadores. O provedor de identidade autentica os utilizadores e fornece ao provedor de serviço a indicação que o utilizador foi autenticado;
- WS-Federation [16] – utilizado por clientes SOAP e serviços web;
- LDAP [17] – para autenticar a partir de uma Active Directory [18].

### **3.2.3.3. Google Authenticator**

#### **O que é?**

O Google Authenticator é uma aplicação propriedade da empresa Google, que permite aos seus utilizadores efetuarem login nas suas aplicações Google, utilizando um método baseado numa verificação a dois passos. São usados dois sistemas na verificação: ‘Time-based One-time Password Algorithm’ (TOTP) [19] e ‘HMAC-based One-time Password Algorithm’ (HOTP) [20].

O Google Authenticator gera uma ‘one-time password’, com seis a oito dígitos, que o utilizador deve fornecer, além do seu ‘user’ e ‘password’, no momento de efetuar o ‘login’ nos serviços Google. O Authenticator também gera códigos para aplicações de terceiros, como ‘password managers’ ou ‘file hosting services’.

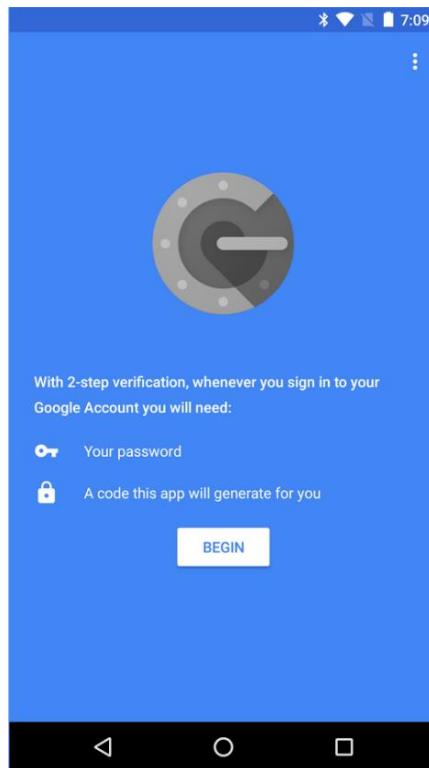


Figura 9: App Google Authenticator (reproduzido a partir de play.google.com)

### Como funciona?

O SP gera uma chave secreta, de 80 bits para cada utilizador. Esta chave pode ser gerada de duas formas: como uma ‘string’ de 16, 26 ou 32 caracteres de base32 ou através de um código QR code [21].

O SP partilha esta chave com o Google Authenticator, para que este possa armazena-la e utiliza-la posteriormente. Assim, que o Google Authenticator armazena a chave, pode servir-se dela para assinar um ‘Hash’, utilizando ‘HMAC’ e ‘SHA1’. A mensagem ‘Hash’ é assinada com o número de períodos de 30 segundos, ocorridos desde o início do processo, contados em ‘Time Unix’ [22] ou por um contador, que é incrementado a cada nova chave criada. Uma parte dessa mensagem é extraída e convertida num código de seis dígitos para complementar o ‘login’ do utilizador.

#### Pseudocode for one-time password (OTP)

```
function GoogleAuthenticatorCode(string secret)
    key := base32decode(secret)
    message := floor(current Unix time / 30)
    hash := HMAC-SHA1(key, message)
    offset := last nibble of hash
    truncatedHash := hash[offset..offset+3] //4 bytes starting at the offset
    Set the first bit of truncatedHash to zero //remove the most significant bit
    code := truncatedHash mod 1000000
    pad code with 0 until length of code is 6
    return code
```

#### Pseudocode for event or counter OTP

```
function GoogleAuthenticatorCode(string secret)
    key := base32decode(secret)
    message := counter encoded on 8 bytes
    hash := HMAC-SHA1(key, message)
    offset := last nibble of hash
    truncatedHash := hash[offset..offset+3] //4 bytes starting at the offset
    Set the first bit of truncatedHash to zero //remove the most significant bit
    code := truncatedHash mod 1000000
    pad code with 0 until length of code is 6
    return code
```

Figura 10: Pseudocode do Google Authenticator (reproduzido a partir de google.com)

#### Protocolos:

- QR Code: usado para configurar a aplicação no smartphone do utilizador; a aplicação lê o QR Code e associa de imediato a aplicação ao serviço Google, que o utilizador pertence.
- HMAC e TOTP: algoritmos utilizados para gerar códigos de validação, na autenticação dos utilizadores.

#### 3.2.3.4. Fido

##### O que é?

O Fast IDentity Online Alliance (FIDO) é um conjunto de especificações técnicas, para suportar mecanismos de autenticação que reduzem a dependência da utilização da ‘password’, na autenticação de utilizadores [23]. Neste sentido, o FIDO tem como base dois conjuntos de especificações: Universal Authentication Framework (UAF) [24], para utilização sem ‘password’; e Universal Second Factor (U2F), para uma autenticação de dois fatores. A utilização do protocolo UAF pelo FIDO, foi desenvolvido de modo a suportar tecnologias de

autenticação como: impressão digital, dados biométricos, leitura da íris, voz, reconhecimento facial, entre outros. O outro protocolo, U2F, foi desenvolvido de forma a suportar padrões de comunicação como Trusted Platform Modules (TPM) [25], UBS tokens de segurança, smart card [26]ou até Near Field Communication (NFC).

A autenticação sem ‘password’, permite ao utilizador registar o seu dispositivo no serviço online que pretende aceder e selecionar um mecanismo de autenticação, como a impressão digital, olhar para a câmara, utilização de voz pelo microfone, inserir um PIN, entre outras opções. Referir ainda, que a utilização do protocolo UAF permite ao serviço selecionar quais os mecanismos de autenticação apresentados aos utilizadores. Uma vez terminado o registo, o utilizador repete a ação de autenticação, utilizando um dos mecanismos, sempre que precisar de se autenticar no serviço. Desta forma, o utilizador não necessita de colocar a sua ‘password’. A autenticação forte, do FIDO, permite que o utilizador adicione um segundo fator no momento do ‘login’. O utilizador faz o ‘logon’ utilizando o seu ‘user’ e a sua ‘password’, de seguida o serviço solicita ao utilizador que apresente um dispositivo de segundo fator, para validar a autenticação.

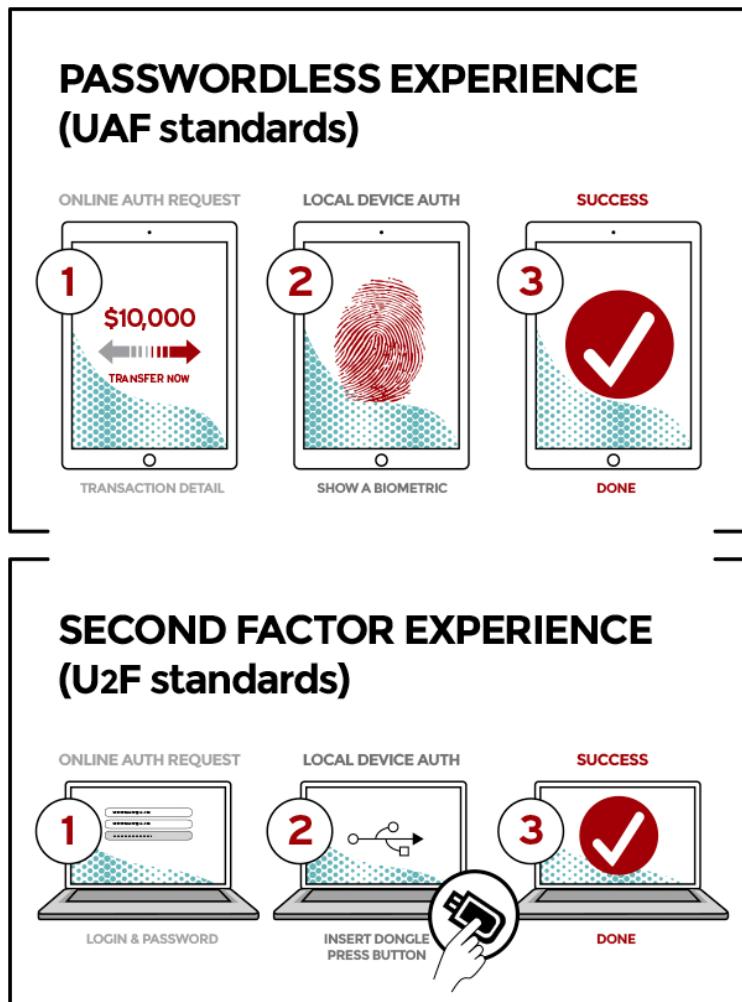


Figura 11: Formas de autenticação FIDO (reproduzido a partir de [fidoalliance.org](http://fidoalliance.org))

### Como funciona?

Os protocolos FIDO utilizam técnicas padrão, de criptografia de chave pública, para fornecer uma autenticação forte. No processo de registo, o dispositivo cliente, do utilizador, cria um novo conjunto de chaves, mantendo a sua chave privada e registando a sua chave pública no serviço on-line. A autenticação feita pelo dispositivo cliente, só é aceite após este provar que possui a chave privada, através da assinatura de um desafio.

As chaves privadas do cliente só podem ser utilizadas após desbloqueadas pelo utilizador. O desbloqueio é realizado através de uma ação amigável e segura, como deslizar um dedo, inserir um PIN, utilizar a voz ou inserir um dispositivo de segundo fator.

O FIDO protege a privacidade do utilizador, desta forma os protocolos utilizados por ele não fornecem qualquer informação sobre o utilizador, que possa ser utilizada pelos serviços online de forma abusiva, por exemplo, guardar a localização do utilizador.

### Fluxo de registo:

1. O utilizador seleciona um autenticador FIDO disponível no serviço online;
2. O utilizador desbloqueia o autenticador FIDO, utilizando um leitor de impressão digital ou um PIN;
3. O dispositivo do utilizador cria um novo conjunto de chaves, públicas e privadas, exclusivas para esse dispositivo, serviço online e para essa conta de utilizador;
4. A chave pública é enviada para o serviço online e associada à conta do utilizador. A chave privada e qualquer outra informação, sobre o método de autenticação local do utilizador, nunca abandona o dispositivo local.

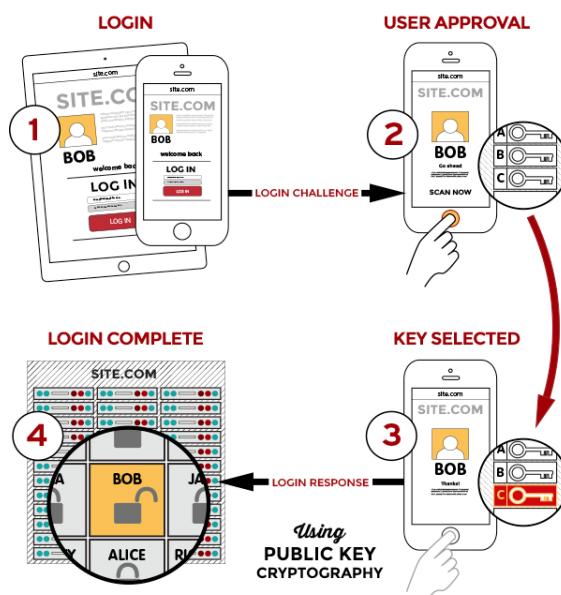


Figura 12: Fluxo de autenticação FIDO (reproduzido a partir de fidoalliance.org)

### Fluxo de Login:

1. O serviço pede ao utilizador que selecione o dispositivo de onde quer iniciar o processo de login;
2. O utilizador desbloqueia o autenticador FIDO, utilizando o mesmo método como se registou no fluxo anterior;
3. O dispositivo usa o identificador do utilizador, fornecido pelo serviço, para selecionar a chave correta e assinar o desafio;
4. O dispositivo envia o desafio assinado de volta ao serviço, para que este comprove com a chave pública armazenada. A autenticação do utilizador é aceite caso o desafio seja comprovado corretamente.

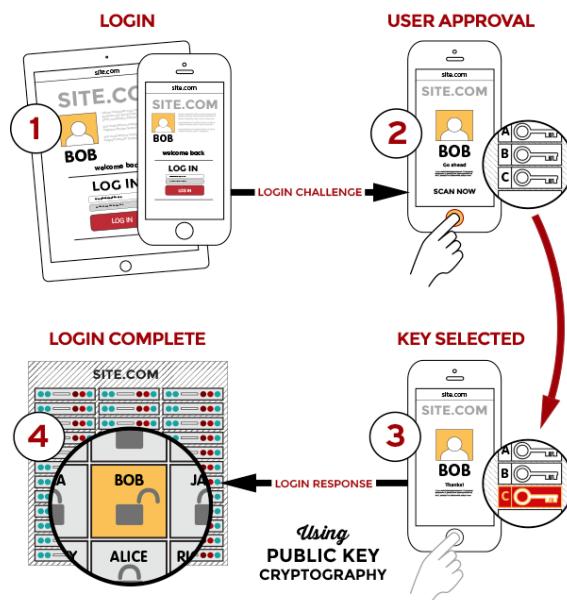


Figura 13: Fluxo de login FIDO (reproduzido a partir de fidoalliance.org)

### Protocolos utilizados:

UAF: em processos de autenticação sem ‘password’;

U2F: em processos de autenticação de dois fatores.

#### 3.2.3.5. OpenID Connect

##### O que é?

O OpenID Connect é uma camada de identidade simples, posicionadas acima do protocolo OAuth 2.0. Esta camada, permite aos utilizadores autenticarem-se nas aplicações sem terem que colocar uma ‘password’. As aplicações validam a identidade dos utilizadores

com base numa autenticação, realizada num serviço de terceiros, conhecidos como servidores de autenticação.

Este protocolo funciona com identidades, levando ao abandono do conceito ‘user’ e ‘password’. As possibilidades deste protocolo são enormes, o que levou a grandes empresas, como Microsoft, Amazon, Google e Facebook, a adotarem-no. As especificações do OpenID Connect foram desenvolvidas no entorno de fluxos de mensagens simples (REST/JSON), para criarem uma fácil integração e suportar qualquer aplicação que assente em JSON ou http [27]. Desta forma, as aplicações cliente não necessitam de controlar todo o processo de autenticação.

Qualquer integração do OpenID Connect, numa aplicação, segundo as suas especificações, está habilitada a pedir e receber informação dos utilizadores, desde que esta seja autorizada por ele.

### **Como Funciona?**

O OpenID Connect é um protocolo de autenticação baseado no OAuth 2.0, que utiliza fluxos de mensagens simples REST / JSON, com o objetivo de tornar o lado complicado em simples e exequível.

Existem três figuras neste protocolo: o utilizador, que passa a ser tratado como uma entidade; o relying Party (RP), aplicação que o utilizador se propõem autenticar; e o provedor de OpenID (OP), serviço especializado em fornecer URLs ou XRIs OpenID, dos utilizadores. Desta forma o OpenID Connect permite aos utilizadores comunicarem a sua identidade a uma aplicação, através de um terceiro de confiança, sem necessite de uma ‘password’. Esta comunicação é feita através da troca de um identificador ou de um OpenID URL/XRI, escolhido pelo utilizador de forma a identifica-lo perante um RP.

O método de autenticação pode variar, geralmente, um OP solicita ao utilizador uma ‘password’ ou um token criptográfico, perguntando-lhe se confia no RP para receber os detalhes da sua identidade. No caso que o utilizador recuse, é redirecionado para o RP com uma mensagem, indicando que a autenticação foi rejeitada. Se o utilizador aceitar, será redirecionado para o RP, onde autoriza ao RP aceder aos seus dados pessoais fornecidos pelo OP.

**Fluxo de autenticação:** O fluxo mais resumido do OpenID Connect, segue as seguintes etapas:

1. O RP envia um pedido de autenticação ao OP;
2. O OP solicita autenticação ao utilizador;

3. O OP responde ao RP, com um token de identificação e, geralmente, um token de acesso;
4. O RP pode enviar o pedido de acesso aos dados do utilizador, através do token de acesso;
5. O OP responde com os dados do utilizador.

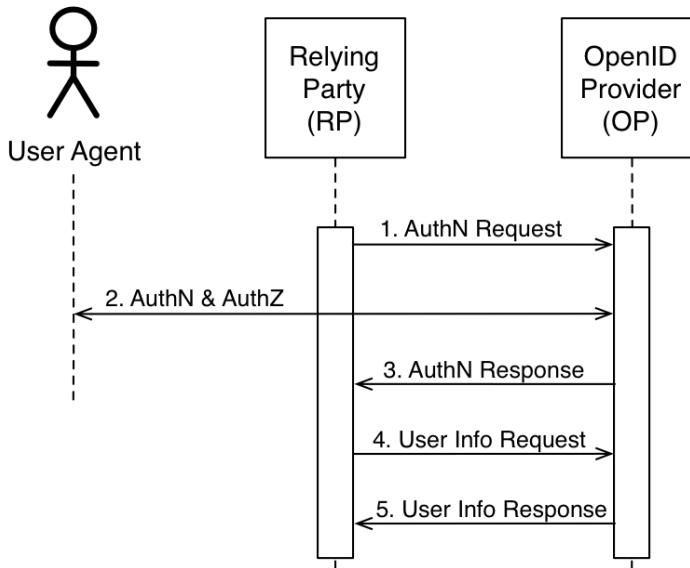


Figura 14: Fluxo OpenID Connect (reproduzido a partir de openid.net)

**Protocolos:** OAuth 2.0: protocolo base do processo de autenticação.

### 3.2.3.6. Cartão Cidadão

#### O que é?

O Cartão de Cidadão (CC) é o documento de identificação dos cidadãos portugueses, desenvolvido durante o antigo governante de José Sócrates e emitido em meados de 2006-2007, em fase experimental na Região Autónoma dos Açores. Este cartão não é apenas uma substituição do bilhete de identidade, ele agrupa outros documentos pessoais do cidadão, nomeadamente: o cartão de beneficiário da Segurança Social; o cartão de utente do Serviço Nacional de Saúde; e o cartão de contribuinte. O CC segue todas as normas internacionais recomendadas, para que este seja um documento de identificação e um documento de viagem reconhecido oficialmente.

O CC é muito mais que um simples documento, foi concebido para ser um documento inteligente, um 'smart card'. Dotado de um chip, o CC permite ao cidadão identificar-se perante terceiros, como sujeito titular de direitos e de deveres. O CC concede aos cidadãos

portugueses fazerem prova da sua titularidade, de forma presencial no seu relacionamento com o mundo físico ou digitalmente. Outro objetivo foi melhorar o nível de segurança dos cartões de identificação, de forma a dificultar eventuais reproduções ou falsificações dos mesmos.

O chip permite guardar dados pessoais do cidadão, garantindo segurança e privacidade. Por exemplo, impede que dados médicos sejam lidos por funcionários das Finanças, cada entidade só tem acesso a dados que dizem respeito ao seu departamento, para evitar eventuais excessos de poder ou lesar a privacidade do cidadão.



Figura 15: Cartão de cidadão Português (reproduzido a partir de autenticacao.gov.pt)

### Como Funciona?

Está equipado com um ‘middleware’, que disponibiliza um conjunto de funcionalidades: permite operações criptográficas, como a assinatura de um documento de Word; e não-criptográficas, como a consulta de dados identificativos, como o nome ou a morada do portador do CC.

O CC é um cartão ID-1, definido pela norma ISO/IEC 7810 (dimensões 85,60 × 53,98 mm) e correspondente ao formato TD-1 da ‘machine readable travel documents’ (MRTD), definido pela ICAO. Esta norma deve ser complementada pela ISO 7816-2, que define a posição e funcionamento do chip do cartão.

De forma a corresponder a todos estes objetivos, o chip do CC tem as seguintes características:

- É um chip ‘Java Card’ e multi-aplicação;
- Suporta a versão mais recente da plataforma ‘Java Card’ e o uso de ‘logical channels’;
- Possui uma memória EEPROM (ou equivalente) mínima de 64 KB;
- Possui capacidades de gestão de memória dinâmica, suportando ‘garbage collection’ pela JVM e de proteção de memória;
- Possui capacidades de gestão de espaço de armazenamento, incluindo desfragmentação e reutilização de espaço libertado;

- Possui capacidades de ‘true random number generation’;
- Suporta múltiplos PIN;
- Os PIN estão em conformidade com a norma ISO/IEC 7816- 4;
- Suporta mecanismos de bloqueio, em caso de erro na introdução do PIN, após repetitivos desbloqueios, por meio de introdução de PUK e chave administrativa de acesso;
- Suporta mecanismos de geração de novo PIN, em caso de esquecimento, mediante a introdução do PUK do cidadão e de PUK aplicacional de geração de PIN;
- Possui um motor criptográfico interno que suporta:
  - i. Assinatura e verificação RSA de 1024 bits;
  - ii. Assinatura eletrónica qualificada, segundo a norma CEN CWA 14169 ‘Secure Signature-creation devices’ EAL 4+;
  - iii. DES e TDES – ‘triple Data Encription Standard’;
  - iv. MD5, SHA-1 e SHA-256, no mínimo;
  - v. MAC – ‘message authentication code’;
  - vi. PKCS#1 RSA - *Cryptography Standard* e PKCS#15 – ‘Cryptographic Token Information Format Standard’;
  - vii. É compatível com leitores de cartões da norma EMV-CAP, para funcionamento de autenticação multicanal, baseada em ‘one-time password’;
- Possui uma chave de proteção da personalização inicial;
- Está preparado para resistir aos ataques, conhecidos como ‘hardware attack’, ‘timing attack’, ‘simple power analysis’, ‘differential power analysis’, entre outros.

## Fluxo

O fluxo seguinte exemplifica, de forma sumária e transversal, a utilização do Autenticação.Gov, com base no caso de autenticação de um cidadão, junto de uma entidade, utilizando o CC:

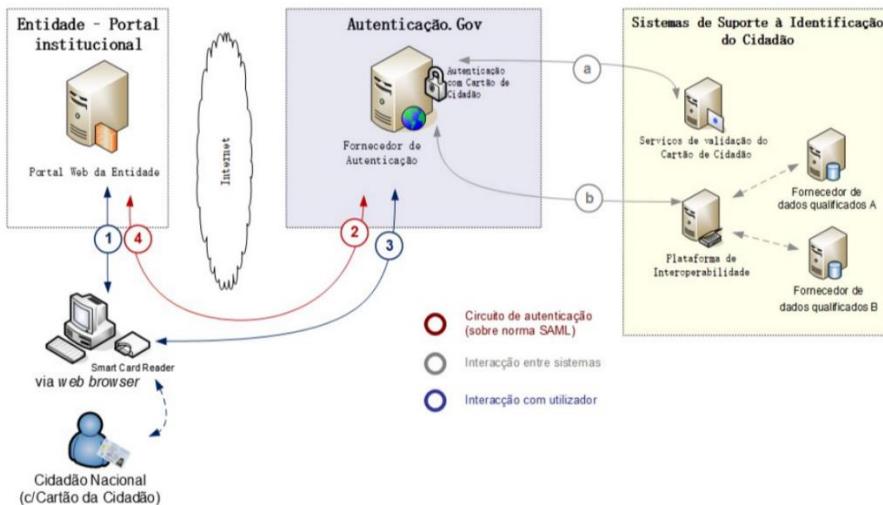


Figura 16: Arquitetura sistemas com Cartão de Cidadão (reproduzido a partir de autenticacao.gov.pt)

1. O utilizador pretende aceder à área privada do portal de uma entidade, na qual necessita provar a sua identidade;
2. O portal da entidade delega a autenticação e redireciona o utilizador para o Autenticação.Gov, juntamente com um pedido de autenticação assinado digitalmente;
3. O Autenticação.Gov valida o pedido de autenticação recebido e solicita a autenticação do cidadão com recurso ao seu CC, pedindo a inserção do seu PIN de autenticação. Durante este processo, o Autenticação.Gov efetua as seguintes operações internas:
  - a. Valida as credenciais do cidadão, com recurso à ‘public key infrastructure’ (PKI) do CC via ‘Online Certificate Status Protocol’ (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 e pode incluir a obtenção de dados da Federação de Identidades ou de outras entidades.
4. A identificação e os atributos do utilizador são autenticados e assinados digitalmente, pelo Autenticação.Gov. Após este processo o utilizador é direciona de volta ao portal da entidade original. Cabe à entidade a validação das credenciais do Autenticação.Gov e utilização dos atributos do cidadão.

### Protocolos:

PKI: utilizado para revogar o certificado digital;

OCSP: utilizado para obter o ‘status’ de revogação de um certificado digital.

### 3.2.3.7. Criptografia

#### O que é?

A criptografia é o processo de codificação de mensagens, para que apenas as partes autorizadas possam ter acesso. Por si só, a criptografia, não impede a interferência, mas nega o conteúdo a um intercetor.

Num esquema deste género, a mensagem em texto simples é criptografada, usando um algoritmo, gerando texto cifrado que só pode ser lido se descriptado. Por razões técnicas, um esquema de criptografia, geralmente, utiliza um ‘pseudo-aleatório’: chave de criptografia, gerada por um algoritmo.

Existem formas de descriptar uma mensagem sem possuir a chave, mas são necessários grandes recursos e habilidades técnicas. Por outro lado, um receptor autorizado para descriptar uma mensagem, facilmente decifra a mensagem com uma chave fornecida pelo autor.

#### Como funciona?

A criptografia reside em quatro objetivos principais:

- Confidencialidade da mensagem: só o destinatário autorizado deve ser capaz de extrair o conteúdo da mensagem;
- Integridade da mensagem: o destinatário deverá ser capaz de determinar se a mensagem foi alterada durante a transmissão;
- Autenticação do remetente: o destinatário deverá ser capaz de identificar o remetente e verificar que foi o próprio a enviar a mensagem;
- Rejeição do emissor: não deverá ser possível ao emissor negar a autoria da mensagem.

Nem todos os sistemas ou algoritmos criptográficos são utilizados para atingir todos os objetivos, habitualmente, existem algoritmos específicos para cada uma destas funções. Inclusive, em sistemas criptográficos concebidos convenientemente e utilizados adequadamente, alguns dos objetivos acima não são práticos. Por exemplo, o remetente de uma mensagem pode querer permanecer em incógnito, ou o sistema pode destinar-se a um ambiente com recursos computacionais limitados.

Com a era moderna a criptografia deixou de se preocupar exclusivamente com a confidencialidade da mensagem, passou a preocupar-se também com facto da mensagem passar a ser ilegível, por um intercetor não autorizado. Nasce, assim, a criptografia moderna.

Neste documento, utilizamos a criptografia moderna, existente em várias áreas, abordando o que conhecemos como chaves simétricas e chaves pública.

A criptografia de chave simétrica refere-se a métodos, nos quais, tanto o remetente como o receptor partilham um segredo, isto é, a mesma chave de encriptação é conhecida pelas duas partes, para que uma possa encriptar e a outra desencriptar.

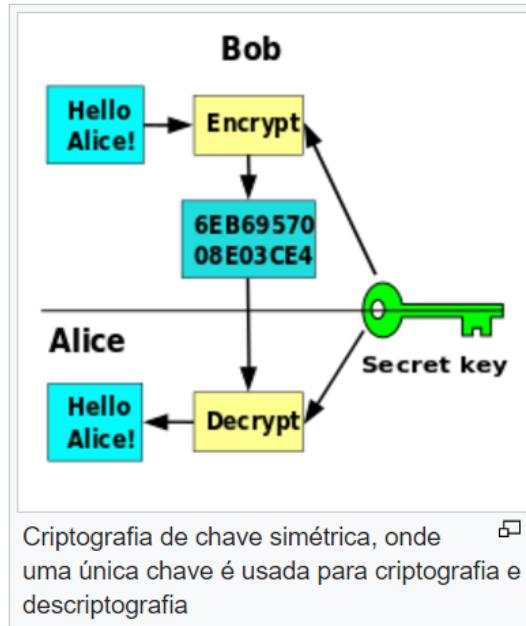


Figura 17: Criptografia com chaves simétricas (reproduzido a partir de [en.wikipedia.org](https://en.wikipedia.org))

A criptografia de chave pública baseia-se em métodos que utilizam duas chaves diferentes, a pública e a privada. Num sistema de chave pública, a chave pode ser distribuída livremente, enquanto a chave privada deve permanecer secreta. Este sistema de criptografia, de chave pública, é utilizada para encriptar, enquanto a chave privada é usada para desencriptar.

O receptor partilha a sua chave pública com o emissor, permitindo que este encripte a mensagem com a chave partilhada; quando o receptor recebe a mensagem, este utiliza a sua chave privada, que ambos têm conhecimento, para desencriptar a mensagem.

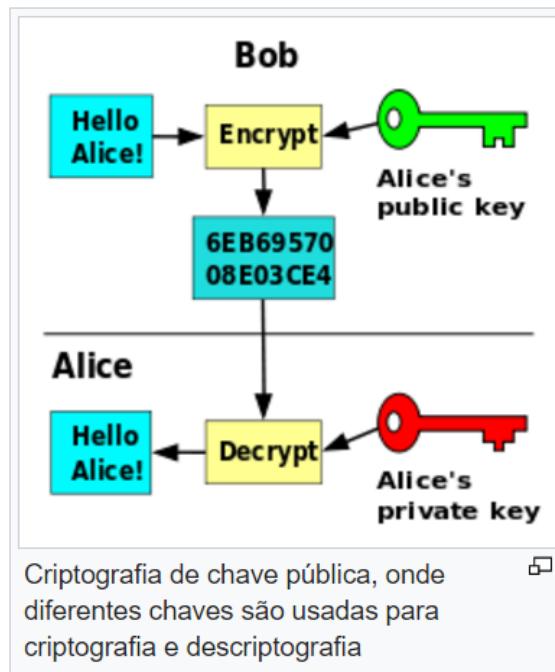


Figura 18: Criptografia com chave pública e desencriptar com chave privada (reproduzido a partir de en.wikipedia.org)

### 3.2.3.8. Certificados Digitais

#### O que é?

O certificado digital é um documento eletrônico, utilizado para provar a propriedade de uma chave pública, por norma, inclui informações sobre a chave, a identidade do seu proprietário e a assinatura digital de uma entidade, que tenha validado o conteúdo do certificado digital.

Um certificado digital é válido se, em uma infraestrutura de chaves públicas, é assinado pela Autoridade Certificadora (AC) que o emitiu. No caso de uma rede de confiança, o certificado é assinado pela própria entidade e também assinado por outros, que dizem confiar nessa entidade. Em ambos os casos as assinaturas contidas em um certificado são validadas por uma entidade, que diz confiar nos dados contidos naquele certificado. O certificado pode ser anulado, no caso que seja descoberta a sua chave privada, se o seu relacionamento entre uma entidade e a chave pública estiver incorreta ou tenha sofrido uma alteração.

#### Como funciona?

Os certificados digitais, para serem emitidos, seguem um conjunto de processos e normas de validação até que possam ser emitido e validados, por uma entidade emissora, de

certificados digitais. Habitualmente, um processo de emissão de um certificado digital segue os seguintes passos:

1. A entidade emissora do certificado cria um conjunto de chaves criptográficas: uma chave pública e uma chave privada;
2. É emitido um ficheiro chamado ‘Certificate Signing Request’ (CSR), que contem a chave pública da entidade emissora e informações que a AC requer. A própria entidade emissora assina digitalmente o certificado;
3. Confirmação dos dados do individuo, contidos no CSR;
4. O ficheiro CSR é transformado num certificado digital, assinado pela AC.

No processo descrito acima, comprehende-se como conteúdo de um certificado digital válido, a seguinte informação:

- Número de série: identificação do certificado;
- Assunto: pessoa ou entidade identificada;
- Algoritmo de assinatura: algoritmo utilizado para criar a assinatura;
- Assinatura: assinatura da entidade emissora;
- Emissora: nome da entidade emissora, que verificou as informações e emitiu o certificado;
- Validade: período válido do certificado;
- Chave pública: a chave pública da pessoa ou da entidade;
- Algoritmo da chave pública: algoritmo usado para criptografar o certificado de chave pública.

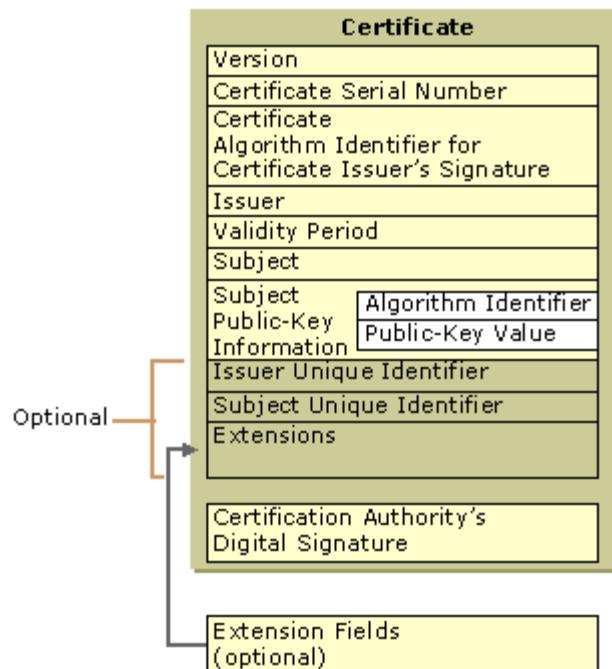


Figura 19: Formato certificado (reproduzido a partir de en.wikipedia.org)

### Fluxo

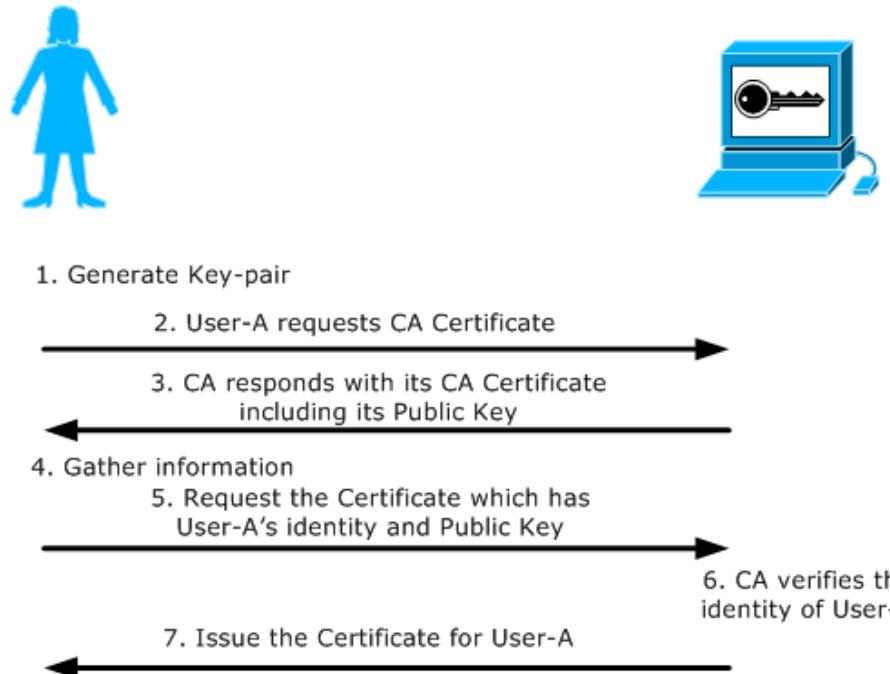


Figura 20: Criação de um certificado digital (reproduzido a partir de en.wikipedia.org)

1. Criação do conjunto de chaves: o utilizador cria um conjunto de chaves (pública e privada);
2. Pedido de certificado: o utilizador solicita o certificado ao servidor da AC;

3. Certificado emitido: a AC responde com o seu certificado, incluindo a chave pública e a assinatura digital, assinada com a chave privada;
4. Reunião de informações: o utilizador reúne todas as informações exigidas pelo servidor AC, para obter o seu certificado. Essa informação pode incluir o email do utilizador, as impressões digitais, entre outros dados. A AC precisa ter a certeza que o utilizador é quem diz ser;
5. Enviar pedido certificado: o utilizador envia o pedido de certificado à AC, constituído pela sua chave pública e informações adicionais. O pedido do certificado é assinado com a chave pública da AC;
6. A AC valida o utilizador: após obter o pedido de certificação, verifica a identidade do utilizador e gera um certificado, vinculando a sua identidade e a sua chave pública. A assinatura da AC valida a autenticidade do certificado;
7. O AC emite o certificado: é emitido o certificado do utilizador.

### **3.2.3.9. Tokens de acesso**

#### **O que é?**

O ‘token’ de acesso é um objeto, que se define de modo compacto e autocontido, para transmitir, com segurança, informações entre duas partes. Geralmente, um ‘token’ de acesso contém os dados de uma sessão de ‘login’ e a identidade de um utilizador, grupos de utilizadores, privilégios do utilizador e, em alguns casos, uma aplicação concreta a que se destina. Enquanto em um ‘token’ convencional são guardadas apenas informações de segurança, em um ‘token’ de acesso é possível guardar dados do utilizador, como informação pessoal [28].

Em condições normais, um ‘token’ de acesso é criado pelo serviço de ‘logon’, no momento em que o utilizador realiza o ‘login’, sempre e quando o processo de autenticação seja válido. O serviço de ‘logon’ possui todas as informações necessárias para criar o ‘token’ de acesso.

O ‘token’ de acesso é um sistema de autenticação de dois fatores de segurança, que podem ser guardados em um computador, ‘PDA’ ou ‘smart phone’, permitindo que este possa ser duplicado pelos vários dispositivos.

De referir que, atualmente, existem inúmeras normas que utilizam e criam os seus próprios ‘tokens’ de acesso. Neste trabalho, abordaremos em concreto o JSON Web Token (JWT), uma vez que fazem parte do protótipo que acompanha este documento.

## Como funciona?

Como referido anteriormente, iremos focar-nos apenas nos tokens JWT, dado a sua utilização concreta neste trabalho.

Um JWT representa um objeto JSON, transportado através de um URL, com um conjunto de reivindicações codificadas, sobre uma estrutura de JSON Web Encryption (JWE). Estes JWT são úteis especialmente no contexto de ‘logon’ único de um ‘browser’ (SSO). As reivindicações JWT podem ser utilizadas para passar a identidade do utilizador autenticado, entre um provedor de identidade e um provedor de serviços, ou qualquer outro tipo de reivindicação. Os ‘tokens’ também podem ser autenticados e criptografados.

Os JWTs são definidos em três partes; cabeçalho, corpo e assinatura. Habitualmente, o cabeçalho identifica o algoritmo utilizado, para produzir a assinatura:

### Cabeçalho:

```
header = '{"alg":"HS256","typ":"JWT"}'
```

Figura 21: Cabeçalho de um JWT (reproduzido a partir de en.wikipedia.org)

HS256: Indica que este ‘token’ é assinado usando HMAC-SHA256.

### Corpo:

```
payload = '{"loggedInAs":"admin","iat":1422779638}'
```

Figura 22: Corpo de um JWT (reproduzido a partir de en.wikipedia.org)

O corpo contém os dados do utilizador, como sugerido na especificação JWT. Incluímos um ‘timestamp’ chamado ‘iat’, abreviação de ‘emitido em’.

### Assinatura:

```
key          = 'secretkey'
unsignedToken = encodeBase64(header) + '.' + encodeBase64(payload)
signature    = HMAC-SHA256(key, unsignedToken)
```

Figura 23: Assinatura de um JWT (reproduzido a partir de en.wikipedia.org)

A assinatura é calculada em base64url, resultado do agrupamento do cabeçalho e do corpo, ambos codificados em base64.

### Token:

```
token = encodeBase64(header) + '.' + encodeBase64(payload) + '.' + encodeBase64(signature) # token is now:  
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJsb2dnZWRJbkFzIjoiYWRtaW4iLCJpYXQiOjE0MjI3Nzk2Mzh9.gzSraSYS8EX8xLN_0wnFSRgCzcmJmMjLiuyu5CSpyHI
```

Figura 24: Exemplo do formato (reproduzido a partir de en.wikipedia.org)

Para entender melhor a informação contida em um JWT, foram definidos os seguintes campos padrão:

- Emissor (`iss`): identifica quem emitiu o JWT;
- Assunto (`sub`): identifica o assunto da JWT;
- Audiência (`aud`): identifica os destinatários para os quais o JWT é destinado;
- Tempo de expiração (`exp`): identifica o tempo de expiração, após esse tempo o JWT não deve ser aceite;
- Not before (`nbf`): identifica o exato momento em que o JWT é aceite para processamento;
- Emitido em (`iat`): identifica o momento em que o JWT foi emitido;
- JWT ID (`jti`): identificador único, sensível a maiúsculas e minúsculas do ‘token’, mesmo entre diferentes emissores.

```
HTTP/1.1 200 OK  
Content-Type: application/json  
Cache-Control: no-store  
Pragma: no-cache  
  
{  
  "access_token": "SlAV32hkKG",  
  "token_type": "Bearer",  
  "refresh_token": "8xLoxBtZp8",  
  "expires_in": 3600,  
  "id_token": "eyJhbGciOiJSUzI1NiIsImtpZCI6IjFlOWdkazcifQ.ewogImlzc  
yI6ICJodHRwOi8vc2VydmVyLmV4YW1wbGUuY29tIiwKICJzdWIiOiAiMjQ4Mjg5  
NzYxMDAxIiwKICJhdWQiOiaiczZCaGRSa3F0MyIsCiAibm9uY2UiOiAibi0wUzz  
fV3pBMk1qIiwKICJleHAIoAxMzExMjgxOTcwLAogImlhdCI6IDEzMTEyODA5Nz  
AKfQ.ggW8hZ1EuVLuxNuuIJKX_V8a_OMXzR0EHR9R6jgdqrOOF4daGU96Sr_P6q  
Jp6IcmD3HP99Obi1PRs-cwh3LO-p146waJ8IhehcwL7F09JdijmBqkvPeB2T9CJ  
NqeGpe-gccMg4vfKjkM6FcGvnzZUN4_KSP0aAp1t0J1zZwgjxqGByKHiOtX7TpD  
QyHE51cMiKPXfEIQILVq0pc_E2DzL7emopWoaoZTF_m0_N0YzFC6g6EJbOEoRoS  
K5hoDalrcvRYLSrQAZZKflyuVCixEoV9GfnQC3_osjzw2PAithfubEEBLuVVk4  
XUVrWOLrLl0nx7RkKU8NXNHq-rvKMzqg"  
}
```

Figura 25: Formato de um JWT (reproduzido a partir de en.wikipedia.org)

## Fluxo

No fluxo a baixo, está representado um fluxo abstrato de um sistema assente na utilização de ‘tokens’ de acesso:

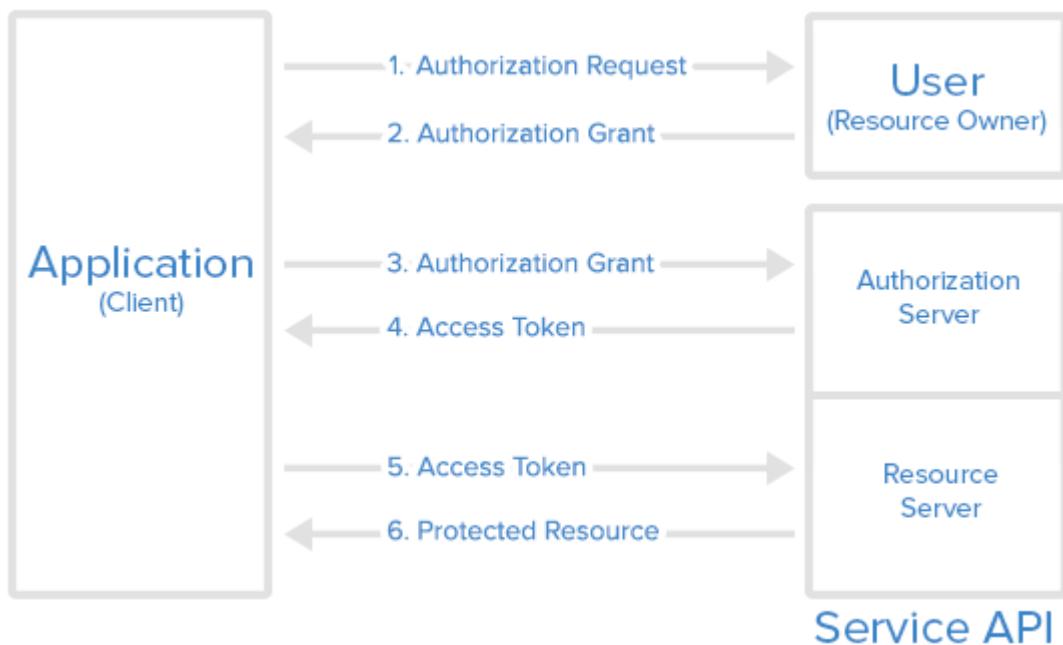


Figura 26: Fluxo de comunicação JWT (reproduzido a partir de en.wikipedia.org)

1. A aplicação pede autorização para aceder aos dados do utilizador;
2. Se o utilizador autorizar o pedido, a aplicação recebe autorização para conectar-se;
3. A aplicação pede um ‘token’ de acesso ao servidor de autorização (API), apresentando a sua própria identidade e a autorização para se interligar.
4. Se a identidade da aplicação for autenticada e a autorização de concessão são válidas, o API emite um ‘token’ de acesso;
5. A aplicação pede o recurso que deseja aceder ao API e apresenta o ‘token’ de acesso para a autenticação;
6. Se o ‘token’ de acesso for válido, o API autoriza o acesso ao recurso.



## 4. Trabalhos Relacionados

Neste capítulo, analisaremos alguns trabalhos relacionados com o Open ID Connect e, alguns deles, estão diretamente relacionados com o tema desta dissertação, por fazerem referência aos ‘Smart Cards’. Os restantes trabalhos também são importantes de examinar, para termos conhecimento do que está a ser feito neste momento sobre estes temas.

### 4.1 Smart Cards e Open ID

Uma das implementações que investigamos está relacionada com um operador de telecomunicações móveis [29]. Este projeto foi desenvolvido para controlar e melhorar as suas infraestruturas de segurança.

A implementação foi mais além do que é o conceito do protocolo Open ID, criando um novo conceito chamado ‘Local OP’ para controlar, principalmente, a autenticação dos utilizadores e emitir afirmações OpenID. Todo este sistema funciona com a utilização de ‘Smart Cards’ que possuem capacidades ‘Universal Integrated Circuit Card’ (UICC) para permitir guardar e criptografar dados.

Neste projeto, foram desenvolvidos outros conceitos mas, dada a sua complexidade e questões técnicas, não iremos abordar em detalhe, mas ficamos com uma ideia geral. Na globalidade o fluxo é o seguinte:

1. O RP procura e associa-se a um OP Support Function (OPSF);
2. OPSF procura um segredo (S), associa um identificador:  $asc-hdl$  e cria uma  $Sa = KDF(S, asc-hdl)$  para a sessão OpenID. Por fim retorna a  $asc-hdl$  e  $Sa$  ao RP.
3. O RP encripta  $Sa$  com segredo Diffie-Hellman (DH) estabelecido entre RP e o OPSF, de seguida redireciona o utilizador para o seu ‘Local OP’;
4. O ‘Local OP’ autentica o utilizador solicitando o seu PIN e redireciona o utilizador para RP com uma mensagem OpenID;
5. O RP valida a mensagem.

### 4.2 CC e acesso a plataformas Cloud

Este caso de estudo refere-se de um projeto Português [30], baseado no CC, que refere alguns pontos que já abordamos e iremos abordar mais à frente, como por exemplo, problemas relacionados com o número de identidades a que um utilizador é obrigado a gerir hoje em dia.

Aqui, são abordados os serviços Cloud, que suportem o protocolo OAuth, tais como DropBox, Skydrive e Cloudpt. A arquitetura baseia-se no conceito de identidade e de

autenticação, com um E-ID. Para colmatar os problemas das inúmeras entidades que cada utilizador é obrigado a gerir e a sua própria segurança, é necessário o uso do E-ID que oferece uma autenticação forte. O projeto está desenvolvido sobre uma API REST, para interagir com os três provedores de serviços, e uma aplicação web, para integrar a autenticação do CC.

A API REST desenvolvida, serve para uniformizar as API's REST dos fornecedores de serviços 'Cloud', para que o protocolo OAuth possa ser implementado em versões diferentes. A API também utiliza o formato JSON para a transferência de 'tokens' de acesso e provas de autenticação. Para esclarecer como este projeto funciona, apresentamos o respectivo fluxo de funcionamento:

1. O utilizador efetua a sua autenticação com o CC na aplicação web;
2. Após se ter autenticado corretamente, a aplicação web acede aos dados públicos do CC e guarda-os;
3. O utilizador seleciona o serviço que a pretende aceder e é iniciado o respectivo mecanismo de comunicação. Caso esta comunicação termine com sucesso, é criado um 'token' de acesso;
4. O 'token' é guardado onde está o certificado público do CC;
5. Sempre que o 'token' seja válido, o utilizador evita uma nova autenticação;
6. O utilizador tem acesso a todos os dados associados ao 'token' de acesso.

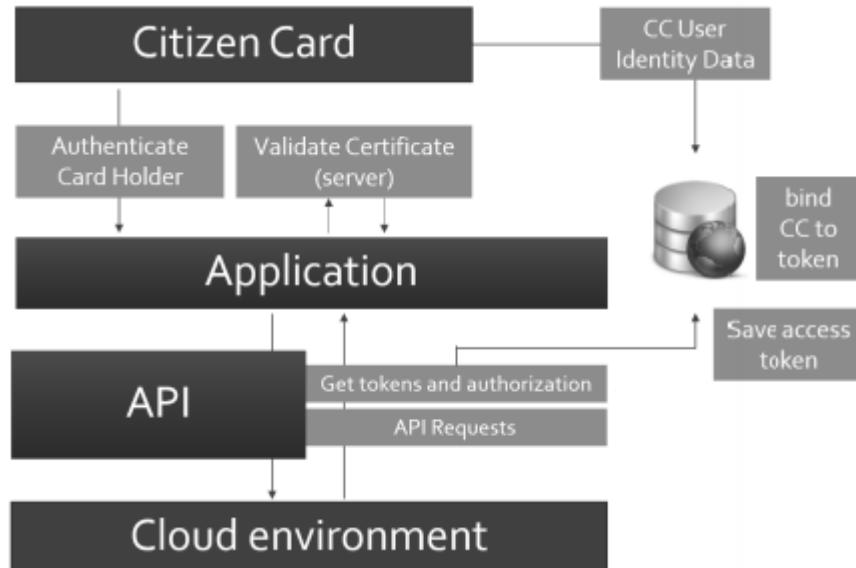


Figura 27: Arquitetura da Aplicação

## 5. Solução

Com a chegada do projeto Cartão de Cidadão Português (CC), os cidadãos portugueses passaram a possuir uma identidade digital dotada de um alto nível de segurança. É com base nesta ferramenta que o protótipo desenvolvido no decorrer deste trabalho, foi realizado.

Tirando partido da alta segurança que possui o CC, aliado com o notável ‘OpenID Connect’, propuséssemos-nos à criação de uma aplicação protótipo, que tire partido dos dois conceitos. Partindo da base das noções do ‘OpenID Connect’ e implementado uma autenticação baseada no CC, foi desenvolvida uma aplicação onde qualquer utilizador, portador de um CC, pode autenticar-se sem necessitar de fornecer credenciais de tipo ‘user’ e ‘password’, permitindo a eliminação desse conceito.

### 5.1 Arquitetura da solução

Uma vez que o ‘Open ID Connect’ é um protocolo dirigido para a web e um standard, é por este motivo a escolha. O protótipo não podia ser desenvolvido em nenhuma outra plataforma que não essa. Assim, qualquer cidadão português pode utilizar o seu CC para ter acesso a esta aplicação, desde que, o sistema informático suporte leitores de ‘smart cards’ e possua um acesso à internet.

Com base no código fonte da comunidade ‘Open ID Connect’, a chave do protótipo está no módulo desenvolvido para integrar a autenticação do CC. Permitindo a um utilizador autenticar-se num provedor de identidades, utilizando o método de autenticação do CC e descartando por completo a utilização de um ‘user’ e ‘password’.

O protótipo é composto por duas aplicações: o provedor de identidades, onde se encontra o módulo desenvolvido para suportar o CC; e uma aplicação cliente, que permite a autenticação através de um servidor de autenticações, desenvolvida de acordo com o protocolo ‘Open ID Connect’.

A aplicação cliente trabalha sobre todas as regras do protocolo e não requereu qualquer alteração. O que esta aplicação faz é solicitar um ‘login’ aos utilizadores, através de um provedor de identidades, e, em função da resposta, apresenta uma mensagem de erro, no caso de haver algum problema com o ‘login’, ou uma página com os dados do utilizador, provando que efetuou o ‘login’ com sucesso.

Neste ponto o Moodle funciona exatamente da mesma forma, suportado por um ‘plugin’ de configuração, foi possível chamar o IdP desenvolvido para efetuar o ‘login’ na plataforma.

Em relação ao provedor de identidades, para que este suporta-se o CC, foi necessário alterar o processo de ‘login’, incluindo uma chamada ao módulo desenvolvido para comunicar com o CC. Recorrendo às bibliotecas do CC, fornecidas pelo próprio, foram desenvolvidos métodos com todos os passos necessários para efetuar um ‘login’ ou ler dados do CC. As bibliotecas disponibilizam métodos que permite manipular toda a informação contida dentro do cartão, desde a sua leitura até à sua autenticação. Neste caso prático, foram utilizados vários métodos: os de leitura, para os utilizadores realizarem o registro na aplicação; e outros de autenticação, para se autenticarem.

O módulo desenvolvido suporta duas páginas web: a de registo, onde os utilizadores são convidados a registar-se com o próprio CC, recorrendo ao processo de ‘login’ e precedido pela leitura dos dados do utilizador, como o nome, o número de identificação fiscal (NIF), a data nascimento, entre outros; a outra página é a de ‘login’ do servidor, que permite ao utilizador ter acesso a toda a sua informação.

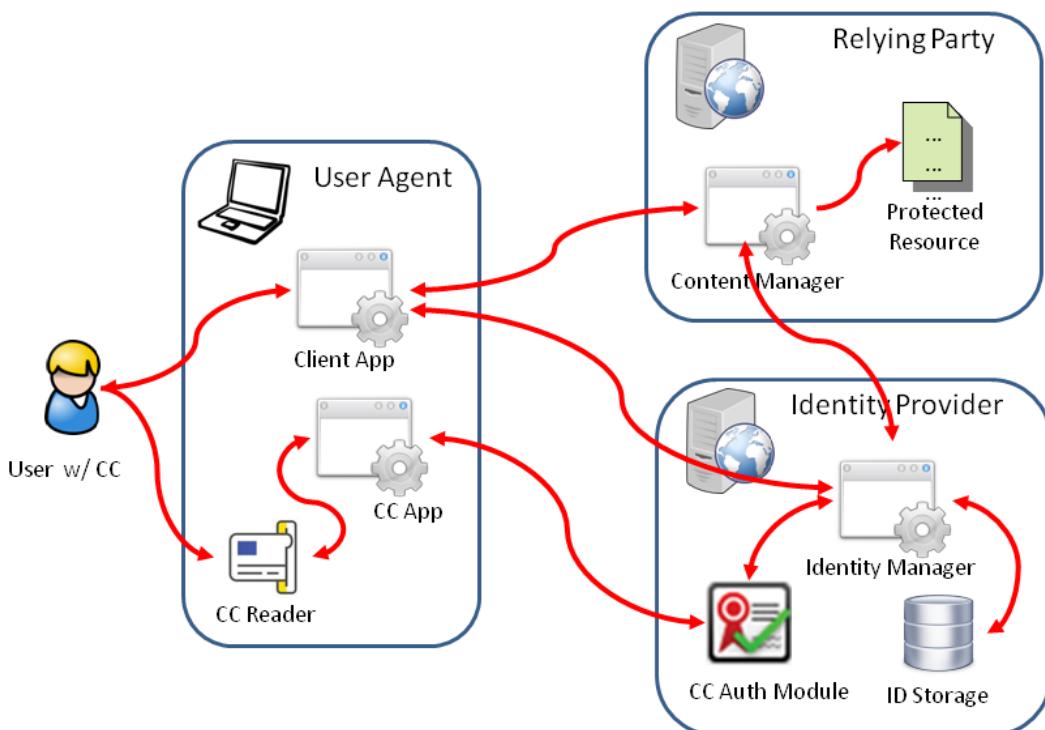


Figura 28: Diagrama da arquitetura do protótipo

## 5.2 Funcionamento da Solução

Diagrama da solução, com recurso à autenticação do CC.

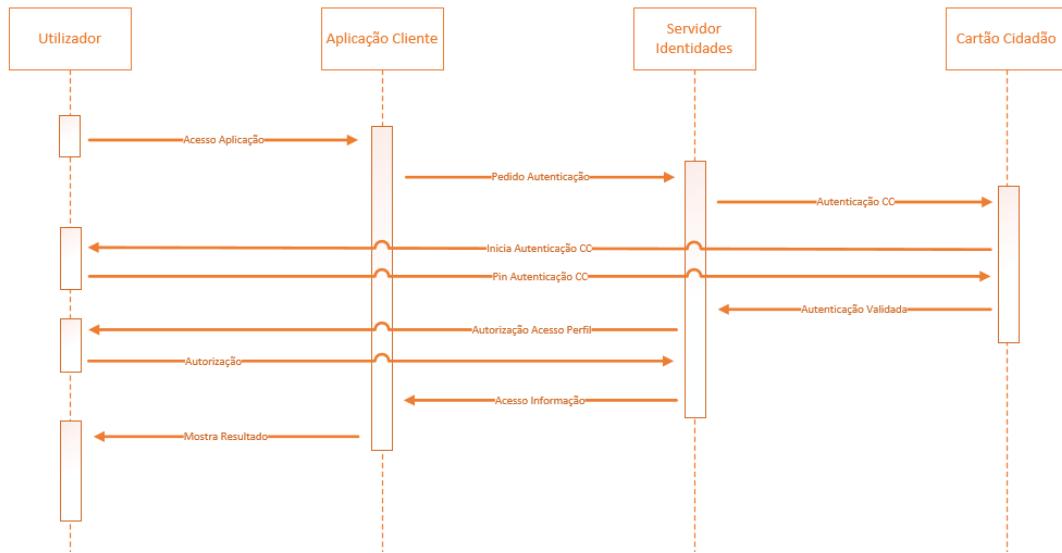


Figura 29: Diagrama de sequência de autenticação com CC

- Acesso à aplicação: o utilizador acede à aplicação cliente do protótipo;
- Pedido de autenticação: o utilizador inicia o processo de autenticação, neste caso, com utilização do CC, para efetuar o ‘login’;
- Início da autenticação do CC: a comunicação com o CC é iniciada e solicita ao utilizador o seu pin de autenticação;
- Pin de autenticação do CC: o utilizador introduz o seu pin;
- Autenticação validada: o servidor recebe a confirmação que o utilizador introduziu corretamente o pin;
- Autorização de acesso ao perfil: o servidor pergunta ao utilizador se autoriza a aplicação cliente aceder aos seus dados;
- Autorização: o utilizador concede a autorização;
- Acesso à informação: através de um ‘token’ de acesso o provedor de identidades disponibiliza a informação do utilizador à aplicação cliente;
- Resultado: a aplicação cliente mostra ao utilizador o resultado final, obtido em todo este processo.

Diagrama da solução, sem recurso à autenticação do CC:

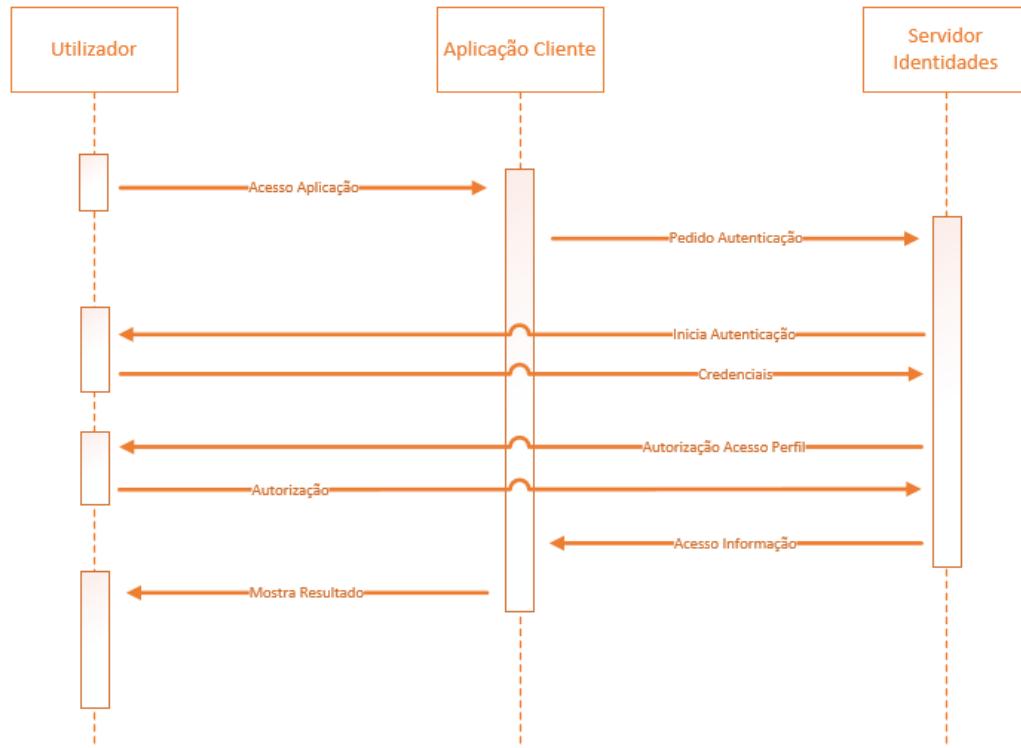


Figura 30: Diagrama de sequência do protótipo sem CC

- Acesso à aplicação: o utilizador acede à aplicação cliente do protótipo;
- Início da autenticação: é solicitado ao utilizador o seu ‘user’ e ‘password’ ;
- Credenciais: o utilizador introduz as suas credenciais;
- Autorização ao acesso do perfil: o servidor pergunta ao utilizador se autoriza a aplicação cliente aceder aos seus dados;
- Autorização: o utilizador concede a autorização;
- Acesso à informação: através de um ‘token’ de acesso, o provedor de identidades disponibiliza a informação do utilizador à aplicação cliente;
- Resultado: a aplicação cliente mostra ao utilizador o resultado final, obtido em todo este processo.

### 5.3 Protótipo

#### Provedor de Identidade

1 – Página Inicial: Apresenta uma mensagem de boas vindas aos utilizadores, e disponibiliza várias opções, desde o registo no servidor até à edição de dados.

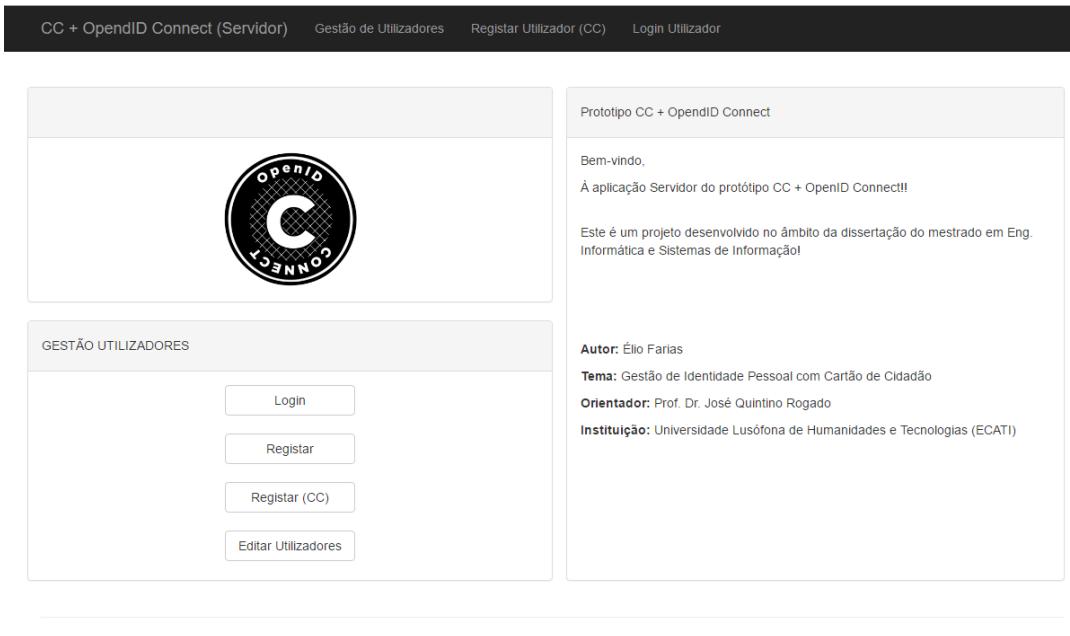


Figura 31: Página inicial protótipo (IdP)

2 – Página de Registo do Utilizador: O registo dos utilizadores pode ser efetuado através de dois processos: o mais simples, através da criação de um ‘user’ e ‘password’; e um outro, mais complexo, através da utilização do CC.

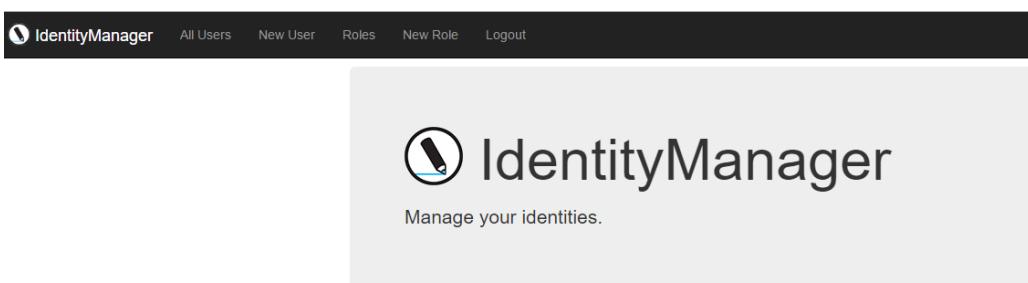


Figura 32: Gestão de utilizadores (IdP)

2.1 – Registo Normal: O utilizador necessita de preencher três dados, o ‘user’, a ‘password’ e confirmar essa mesma ‘password’.



## Create New User

Figura 33: Novo utilizador (IdP)

2.2 – Registo CC: No registo através do CC, o utilizador é reencaminhado para uma página com toda a informação necessária para efetuar o registo. Em caso de sucesso este é informado.



Figura 34: Registo de utilizadores com CC (IdP)

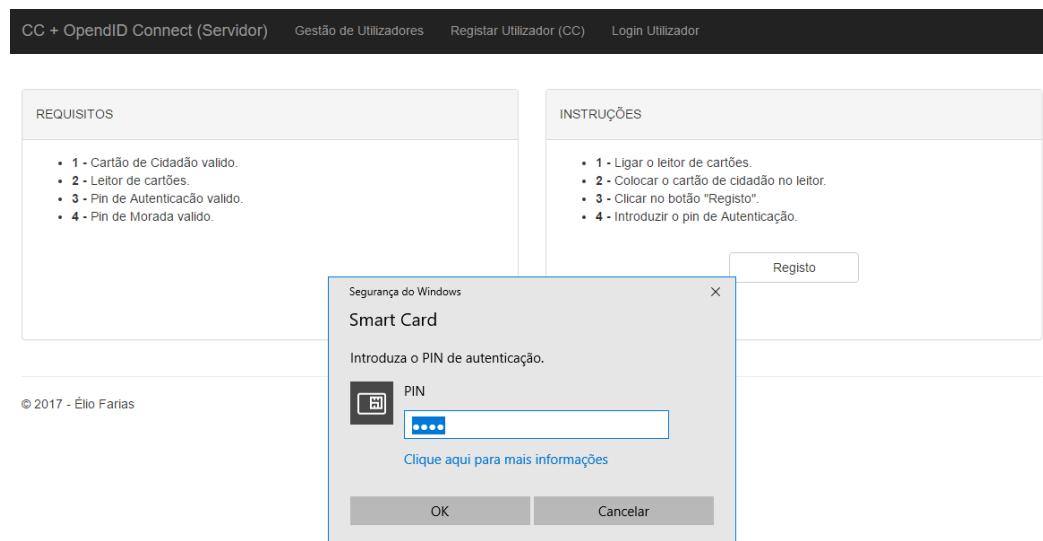


Figura 35: Pin do CC (IdP)

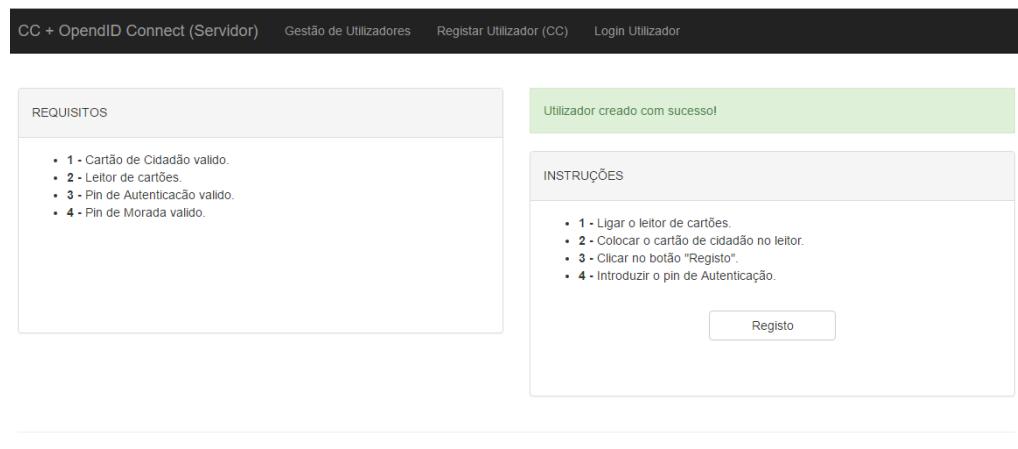


Figura 36: Aviso utilizador (IdP)

3 – Página de edição: é possível fazer toda a gestão dos utilizadores registados; editar os dados do utilizador; ou eliminar utilizadores, obrigando que estes se voltem a registar no servidor.

The screenshot shows the 'IdentityManager' application interface. At the top, there is a navigation bar with links: 'All Users', 'New User', 'Roles', 'New Role', and 'Logout'. Below the navigation bar, the title 'Users' is displayed with a subtitle '(3 found)'. There is a search bar with a 'Search' button. A table lists three users: CAROLINA, ÉLIO, and teste. Each user row has an 'edit' link. The table columns are 'Name', 'Username', and 'Subject'. At the bottom, there is a pagination control with buttons for '<<', '<', '1', '>', and '>>'. To the right of the pagination, it says 'Page 1 of 1'.

	Name	Username	Subject
<a href="#">edit</a>	CAROLINA	127787658	18951-CC-7108
<a href="#">edit</a>	ÉLIO	130092215	13802-CC-5541
<a href="#">edit</a>	teste		684b92f3-cdea-4d14-bb6b-dca2ef9dc6c0

Figura 37: Utilizadores registados (IdP)

## Aplicação Cliente

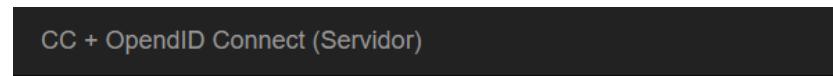
1 – Página Inicial: Apresenta uma mensagem de boas vindas aos utilizadores, e disponibiliza, através do menu SIGNIN, o ‘login’ aos utilizadores.



© 2017 - Élio Farias

Figura 38: Página inicial (RP)

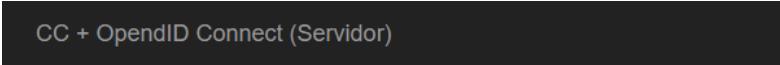
2 – Página Login (Provedor de Identidade): O utilizador ao clicar no menu SIGNIN, é reencaminhado para o provedor de identidades e, através dessa página de ‘login’ do servidor, pode efetuar o ‘logon’. Neste momento, o utilizador deve selecionar o método de ‘login’, de acordo com o seu registo no servidor. Se realizou um registo simples, deve utilizar o seu ‘user’ e ‘password’, caso tenha optado pelo registro através do CC, deve utilizar a opção de ‘login’ com o Cartão Cidadão.



## Login

A screenshot of a "Local Login" page. A modal window titled "Segurança do Windows" is displayed, asking for a PIN. It shows a placeholder "PIN" and a redacted input field. Buttons for "OK" and "Cancelar" are at the bottom.

Figura 39: Página login (IdP)



## Login

A screenshot of a "Local Login" page. A modal window titled "Segurança do Windows" is displayed, asking for a PIN. It shows a placeholder "PIN" and a redacted input field. Buttons for "OK" and "Cancelar" are at the bottom.

Figura 40: Pedido pin ao utilizador (IdP)

3 – Página Dados: Com o ‘login’ realizado, o utilizador é novamente reencaminhado para a aplicação cliente. Neste caso, para uma página com os dados recebidos do servidor. A página

DADOS representa a informação obtida, separada por dois tipos: pessoal e técnica. Todos os dados apresentados nesta página são recebidos segundo as normas do protocolo ‘OpenID Connect’.

The screenshot shows a user interface for managing personal data. At the top, there are tabs: 'CC + OPENID' (selected), 'DADOS', and 'TESTE FARIAS'. The main area is divided into two sections: 'DADOS PESSOAIS' and 'DADOS APLICAÇÃO'.

**DADOS PESSOAIS:**

CLAIMS	DADOS
Utilizador	teste
Nome	Teste
Apelido	Farias
Nº Cartão de Cidadão	0
Autenticação com Cartão	Não

**DADOS APLICAÇÃO:**

CLAIMS	DADOS
Id. Utilizador	684b92f3-cdea-4d14-bb6b-dca2ef9dc8c0
Id. Token	eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJpc3MiOiJsb2dpbi5jbGllbnQub3BlcmF0aW9ucyIsImF1ZGUiOiJsb2dpbi5jbGllbnQub3BlcmF0aW9ucy5pZCIsImV4cCI6MTQwMjIwMDIwMjowMDAwMCwibmJmIjoxNDQwNjY2NjA2fQ.vB411f62a53aa18ef31edcacd6b1ca6a
Acesso Token	bc411f62a53aa18ef31edcacd6b1ca6a
Validade	04/03/2017 20:20:29
Refresh	c1a670567f0e0c1eb5ff8fc810dfd48d

© 2017 - Élio Farias

Figura 41: Página de dados do utilizador (RP)

4 – Página ‘Signout’ (Provedor de Identidade): O utilizador, quando deseja terminar a secção, deve utilizar o ‘SIGNOUT’. Neste ponto, o utilizador é novamente reencaminhado para o servidor e é lhe apresentada a página de ‘Logout’, com a indicação que não existe nenhum ‘login’ efetuado. Para terminar o processo, o utilizador é reencaminhado para a página principal.

The screenshot shows a user interface for logging out. At the top, there is a header: 'CC + OpenID Connect (Servidor)'. Below it, there is a 'LOGOUT' button. A yellow box contains the message: 'De momento não existe nenhum login realizado!!'. Below the message, there is a link: 'Click [aqui](#) para voltar a página principal!!!'

Figura 42: LogOut (IdP)



## Moodle

1 – Página Login: Apresenta ao utilizador todas as possibilidades de login. Notar que o utilizador pode utilizar o Provedor de Identidades criado para este protótipo.

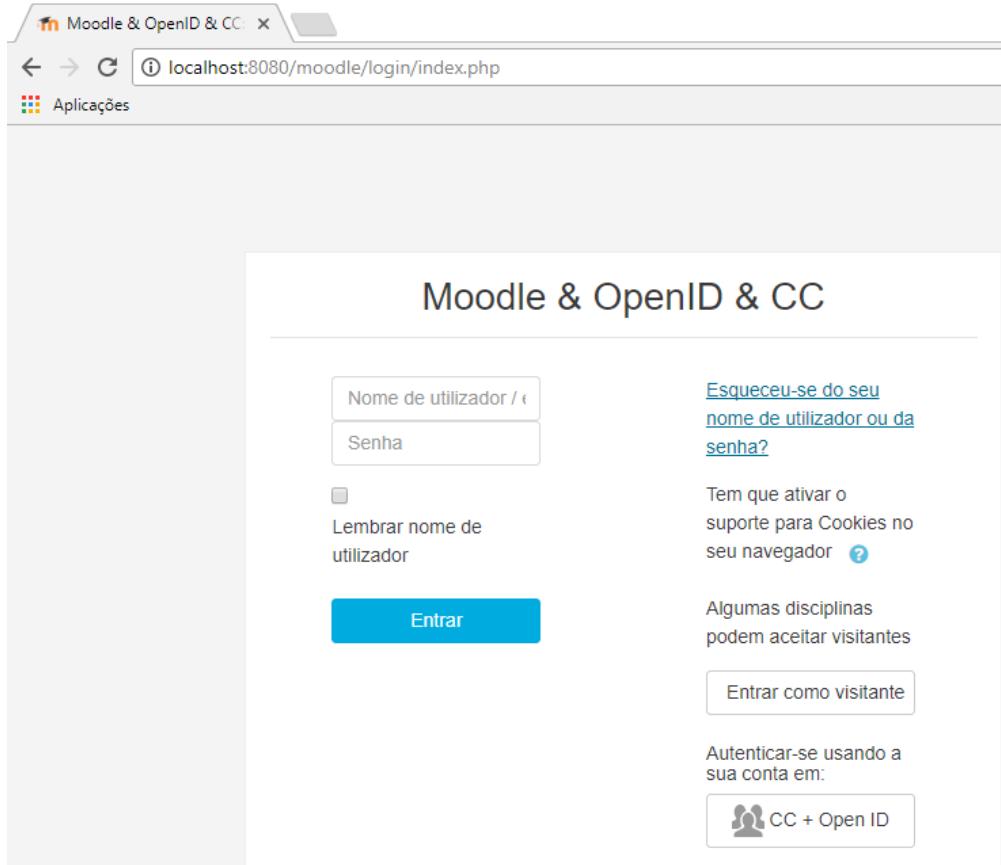
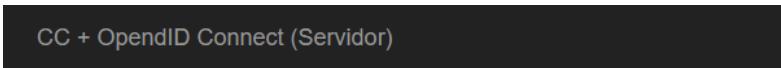


Figura 43: Login Moodle

2 - Página Login (Provedor de Identidade): O utilizador quando decide autenticar-se com “CC + OpenID” é reencaminhado para a página de login do provedor de identidades e, através deste deve efetuar o seu login. Conforme descrito anteriormente, é solicitado ao utilizador que se identifique perante o provedor de identidades. Caso este realize a sua identificação corretamente é-lhe permitido avançar.



## Login

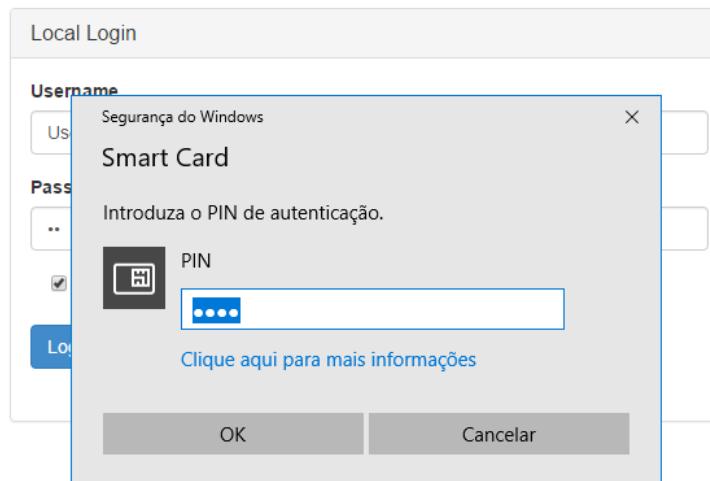


Figura 44: Login com CC

3 – Página de Autorização: Após o login, é solicitado ao utilizador que autorize o Moodle a receber os seus dados.

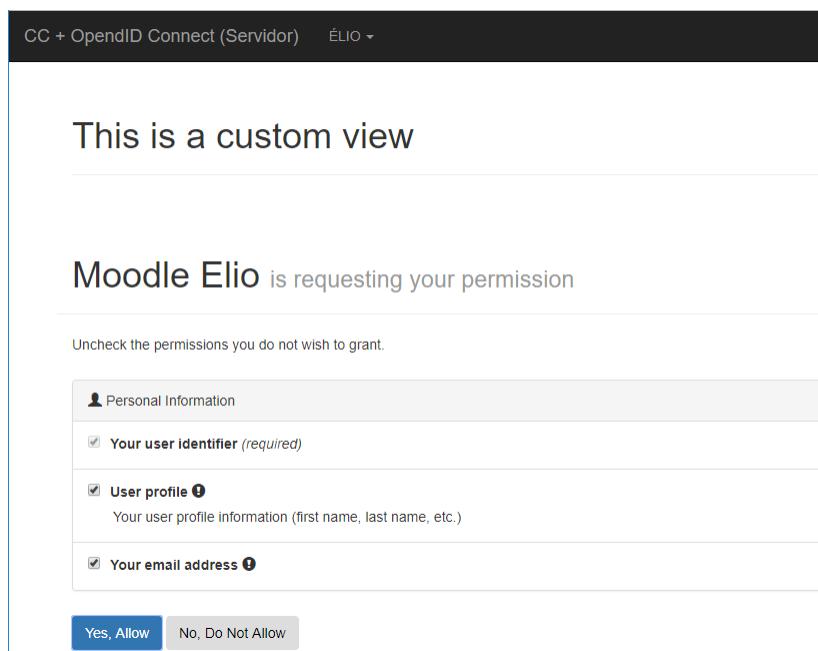


Figura 45: Solicitação de autorização

4 – Página de Registo: Com o login terminado, o Moodle reencaminha o novo utilizador para a sua página de perfil, para o completar. A partir deste momento o utilizador está habilitado para utilizar o Moodle de acordo com o seu perfil.

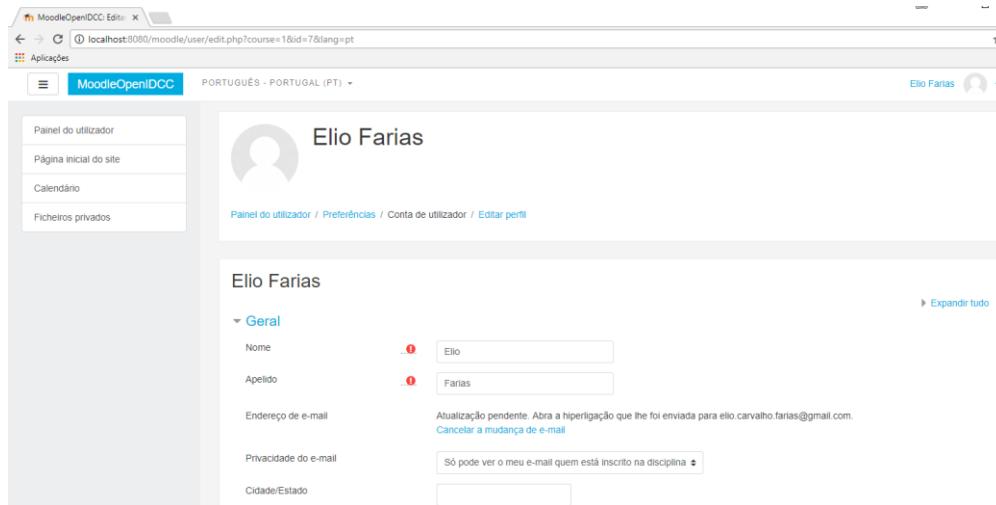


Figura 46: Edição de perfil Moodle



## 6. Implementação

### 6.1 Provedor de Identidades

#### 6.1.1 Autenticação CC

Na aplicação servidor, foi necessário recorrer ao código que controla todo a parte de autenticação, para implementar o módulo de comunicação do CC. Para manter o fluxo do ‘Open ID Connect’ as alterações foram realizadas na função que efetua o ‘login’ local, ‘AuthenticateLocalAsnc’. Neste ponto foram realizadas duas alterações: validar o tipo de ‘login’ e criar a chamada ao módulo de autenticação. Esta função recebe toda a informação que necessita a partir do ‘interface’.

Foi com esta fórmula que soluciona-mos o problema de como ‘Open ID Connect’ vai reconhecer o ‘login’ através do CC. Era necessário que o módulo de autenticação tivesse a capacidade de autenticar a pessoa e, no final, devolver um identificador para que fosse possível identificar o utilizador na base de dados.

```
public override async Task AuthenticateLocalAsync(LocalAuthenticationContext ctx)
{
    var username = ctx.UserName;
    var password = ctx.Password;
    var message = ctx.SignInMessage;

    ctx.AuthenticateResult = null;

    if(isLoginCC)
    {
        WebHost.App_Code.AutenticacaoCartaoCidadao a = new WebHost.App_Code.AutenticacaoCartaoCidadao();
        username = a.Autentica();

        var user = await FindUserAsync(username);
        if (user != null)
        {

            if (userManager.SupportsUserLockout && await userManager.IsLockedOutAsync(user.Id))
            {
                return;
            }

            if (userManager.SupportsUserLockout)
            {
                await userManager.ResetAccessFailedCountAsync(user.Id);
            }

            var result = await PostAuthenticateLocalAsync(user, message);
            if (result == null)
            {
                var claims = await GetClaimsForAuthenticateResult(user);
                result = new AuthenticateResult(user.Id.ToString(), await GetDisplayNameForAccountAsync(user.Id), claims);
            }

            ctx.AuthenticateResult = result;
        }
    }
    else
    if (userManager.SupportsUserPassword)
    {
        var user2 = await FindUserAsync(username);
        if (user2 != null)
```

Figura 47: Função de autenticação (c#)

A função ‘Autentica’ é a responsável por processar todo o fluxo da autenticação, através do CC. O processo está dividido em quatro passos: pesquisa do certificado de autenticação do CC; assinatura do segredo; validação da assinatura; e, por fim, o envio do identificador do utilizador.

```

public string Autentica() {
    X509Certificate2 _certificadoAutenticacaoCartao;
    PteidCertif _Cert;

    try
    {
        //Pteid.Init("");
        //Pteid.SetSODChecking(0);

        //Procura certificado no leitor de cartões
        _Cert = encontraCertAutentCartaoCidadao(Pteid.GetCertificates());

        _certificadoAutenticacaoCartao = new X509Certificate2(_Cert.certif);

        //Assina
        byte[] signature = Sign(secret, _certificadoAutenticacaoCartao.SerialNumber);

        if (Verify(secret, signature, new X509Certificate2(_certificadoAutenticacaoCartao)))
        {

            Pteid.Init("");
            Pteid.SetSODChecking(0);

            return retornaUserName(Pteid.GetID().numBI);
        }
    }
    catch (Exception ex)
    {
        errorCC = ex.Message;
    }
    return string.Empty;
}

```

Figura 48: Função de autenticação CC (c#)

### 6.1.2 Pesquisa do certificado

Na pesquisa do certificado, foi desenhada uma função capaz de identificar o certificado a utilizar, sobre todos os existentes, no equipamento em uso. Após ser encontrado é devolvido para poder ser utilizado nas funções seguintes.

```

2 references
private static PteidCertif encontraCertAutentCartaoCidadao(PteidCertif[] _certificados)
{
    //Percorre toda a lista de certificado até encontrar o certificado
    foreach (PteidCertif a in _certificados)
    {
        if (a.certifLabel == "CITIZEN AUTHENTICATION CERTIFICATE")
        {
            return a;
        }
    }
    return null;
}

```

Figura 49: Função pesquisa de certificado digital (c#)

### 6.1.3 Assinatura do segredo

A assinatura é feita sobre um segredo, enviado pela aplicação (uma 'string' com um texto aleatório), através da chave privada do certificado encontrado na função anteriormente

descrita. É neste momento, que é solicitado ao utilizador a introdução do seu ‘pin’ de autenticação, que é enviado ao CC e este tem a capacidade de confirmar se está correto. A partir daqui, é possível efetuar a assinatura, uma vez que conseguimos obter a chave privada. No fim, é devolvido um ‘array’ de ‘bytes’, resultado da assinatura do segredo.

```

    < references
private static byte[] Sign(string text, string numeroSerieCertificado)
{
    //Access Personal Certificate
    SHA1Managed sha1 = new SHA1Managed();
    UnicodeEncoding encoding = new UnicodeEncoding();
    RSACryptoServiceProvider csp = null;
    X509Store my = new X509Store(StoreName.My, StoreLocation.CurrentUser);

    my.Open(OpenFlags.ReadOnly);

    foreach (X509Certificate2 cert in my.Certificates)
    {
        if (cert.SerialNumber == numeroSerieCertificado)
        {
            //Get chave privada associado ao certificado
            csp = (RSACryptoServiceProvider)cert.PrivateKey;
        }
    }

    //Erro
    if (csp == null) throw new Exception("Certificado invalido!!");

    //Cria Hash do segredo
    byte[] data = encoding.GetBytes(text);
    byte[] hash = sha1.ComputeHash(data);

    //Assina o segredo
    return csp.SignHash(hash, CryptoConfig.MapNameToOID("SHA1"));
}

```

Figura 50: Função de assinatura (c#)

#### 6.1.4 Validação da assinatura

A validação da assinatura é realizada com a chave pública do certificado. Após se obter a chave pública, é comprovado se a encriptação do segredo é igual ao que foi encriptado anteriormente com a chave privada.

```

private static bool Verify(string text, byte[] signature, X509Certificate2 _cert)
{
    // Load the certificate!! Verify the signature from a file
    SHA1Managed sha1 = new SHA1Managed();
    UnicodeEncoding encoding = new UnicodeEncoding();
    X509Certificate2 cert = _cert;
    RSACryptoServiceProvider csp = (RSACryptoServiceProvider)cert.PublicKey.Key; //Get chave publica associado ao certificado

    //Cria Hash do segredo
    byte[] data = encoding.GetBytes(text);
    byte[] hash = sha1.ComputeHash(data);

    //Verifica se a assinatura esta correcta
    return csp.VerifyHash(hash, CryptoConfig.MapNameToOID("SHA1"), signature);
}

```

Figura 51: Função de validação (c#)

### 6.1.5 Envio do identificador

De forma a ser possível enviar para a aplicação o identificador do utilizador, é realizada uma consulta à base de dados para obter esse mesmo identificador.

```
private string retornaUserName(string _numCC)
{
    SqlConnection connection = new SqlConnection(ConfigurationManager.ConnectionStrings["AspId"].ConnectionString);
    string result = string.Empty;

    using (connection)
    {
        SqlCommand command = new SqlCommand(string.Concat("SELECT UserName FROM AspNetUsers WHERE NumCC='", _numCC, "'"), connection);
        connection.Open();

        SqlDataReader dr = command.ExecuteReader();

        while (dr.Read())
        {
            result = dr["UserName"].ToString();
        }

        dr.Close();
    }
    return result;
}
```

Figura 52: Função de pesquisa do utilizador na BD (c#)

No final destes quatro passos, já com o identificador do utilizador, todo o processo do provedor de identidades decorre normalmente, como se fosse uma autenticação através de um ‘user’ e uma ‘password’: carrega toda a informação do utilizador e disponibiliza-a de acordo com o ‘Open ID Connect’.

### 6.1.6 Registo CC

Como foi possível ver na secção anterior, o registo de um utilizador através do CC é realizado de forma muito simples: é solicitado o ‘pin’ ao utilizador e, em caso de ser correto, o registo é criado automaticamente. Para que seja possível, reutilizamos varias funcionalidades da Autenticação, abordada anteriormente. Os três primeiros pontos são exatamente iguais, já o último ponto, o comportamento, é totalmente distinto. Construímos um objeto com os dados do utilizador, obtidos através do CC, após passar pelos três passos anteriores, e é enviado para a base de dados, para que seja criado um registo de utilizador novo.

```

public static AspNetUsers criarUtilizador()
{
    AspNetUsers novoUtilizador = new AspNetUsers();
    X509Certificate2 _certificadoAutenticacaoCartao;
    PteidCertif _Cert;
    Random rnd = new Random();

    errorCC = string.Empty;

    try
    {

        //Procura certificado no leitor de cartões
        _Cert = encontraCertAutentCartaoCidadao(Pteid.GetCertificates());

        _certificadoAutenticacaoCartao = new X509Certificate2(_Cert.certif);

        // Sign text
        byte[] signature = Sign("Test", _certificadoAutenticacaoCartao.SerialNumber);

        if (Verify("Test", signature, new X509Certificate2(_certificadoAutenticacaoCartao)))
        {

            Pteid.Init("");
            Pteid.SetSODChecking(0);

            //saca dados
            byte[] bytes = Encoding.Default.GetBytes(Pteid.GetID().firstname);
            var nome = Encoding.UTF8.GetString(bytes);

            novoUtilizador.Id = string.Concat(rnd.Next(10000, 20000), "-CC-", rnd.Next(999, 10000));
            novoUtilizador.FirstName = nome;
            novoUtilizador.LastName = Pteid.GetID().name;
            novoUtilizador.NumCC = Pteid.GetID().numBI;
            novoUtilizador.UserName = Pteid.GetID().numBI;

        }
    }
    catch (Exception ex)
    {
        errorCC = ex.Message;
    }
    return novoUtilizador;
}

```

Figura 53: Função de criação de um novo utilizador (c#)

### 6.1.7 Moodle

A integração com a plataforma partiu de uma configuração de raiz do Moodle e da instalação do ‘plugin’ “OpenID Connect”. Com este ‘plugin’ e apenas com a sua configuração foi possível estabelecer a realização do ‘login’ e registo na plataforma, através do IdP desenvolvido.

## 7. Resultados Alcançados

A aplicação protótipo descrita nesta dissertação é, como o próprio nome indica, uma prova de conceito. Estamos conscientes que a aplicação apresenta as funcionalidades mínimas, mas que cumpre dois aspectos muito importantes: prova que é possível desenhar e implementar um sistema de ‘login’ seguro com o CC, sem recorrer à utilização de ‘user’ e ‘password’; e que é possível criar uma base de trabalho para novas implementações, baseadas neste modelo.

Inicialmente, o protótipo foi pensado para ser capaz de interagir com plataformas como o Gmail, ou dar acesso a qualquer outra aplicação capaz de aceitar o protocolo OpenID Connect. Nem todos estes objetivos foram possíveis de alcançar, devido ao tempo necessário para o seu desenvolvimento. Para validar um cenário equivalente, a solução encontrada foi criar um ambiente capaz de realizar uma integração com uma aplicação existente, neste caso o Moddle, garantindo que a prova de conceito implementada se pode integrar com qualquer aplicação que suporte OpenId Connect.

Observando este protótipo, com o objetivo de criar uma aplicação real, sabemos que existe muito trabalho pela frente, mas as possibilidades de evolução são muito variadas. Sendo a opção que mais salta à vista ser a integração com plataformas como o Gmail.

Face ao número de problemas encontrados, sobretudo em termos de segurança e de manipulação da própria arquitetura do protocolo com terceiros, foi possível criar duas aplicações para simular o ecossistema que nos propusemos. É um ambiente controlado, mas contém todos os processos básicos. Recorrendo à utilização da plataforma Moodle, podemos comprovar que o conceito do protótipo estava correto. O Moodle é uma plataforma bastante conhecida e utilizada, o que nos permite garantias na integração realizada.

A plataforma Moodle foi escolhida, pois existe um ‘plugin’ que permite a autenticação através do protocolo OpenID Connect. Este ‘plugin’, com o nome ‘OpenID Connect Authentication Plugin’, permite configurar o Moodle como um cliente do IdP utilizado. Com o conhecimento adquirido no desenvolvimento da aplicação cliente, que acompanha este protótipo, consegui-mos rapidamente chegar a uma configuração funcional entre o Moodle e o IdP. Ultrapassado este ponto, efetuamos provas de autenticação, todas elas positivas, o que acabou por validar todo o protótipo. De salientar ainda, que esta solução foi bastante surpreendente e benéfica, porque em questão de minutos e apenas por configuração temos uma plataforma Moodle a autenticar com o IdP desenvolvido.

Um dos grandes problemas foi encontrar a localização correta para incluir o módulo de ‘login’ com o CC. Por um lado, não se podia quebrar a segurança do protocolo e, por outro

lado, havia que mantê-la no próprio módulo. Esta situação foi, sem dúvida, um dos grandes desafios que este protótipo permitiu concretizar.

O objetivo de termos um sistema robusto, onde um utilizador possa efetuar o ‘login’, sem recurso a um ‘user’ e ‘password’, foi alcançado. É possível através do CC, efetuar o ‘login’ no protótipo. Com alguns dos objetivos iniciais descartados, foi determinante provar que um utilizador pode-se registar no provedor de identidades, sem necessitar de criar um ‘user’ e ‘password’ ou utilizar um formulário de registo. O utilizador pode registar-se no servidor unicamente com o seu CC, fornecendo somente os dados contidos no seu cartão.

Depois de todos os problemas iniciais ultrapassados, ainda foi necessário resolver o problema da comunicação com o CC, devido a várias dificuldades encontradas: na instalação da ‘dll’; na informação que a documentação nos fornecia, que era escassa ou nula; e vários ‘bugs’ nas ‘dlls’. Para resolver estas dificuldades foi necessário contatar a empresa que criou o ‘software’ de gestão do CC, de modo a podermos instalar corretamente as ‘dlls’. Apenas depois desta resolução foi possível ter acesso às funções de autenticação e assinatura do cartão.

## 8. Conclusão

O desenvolvimento desde protótipo é de grande relevância para provar que o mecanismo de autenticação dos utilizadores deve e pode ser modificado rapidamente, reduzindo, ou mesmo anulando, a utilização de credenciais de tipo ‘user’ e ‘password’ devido, principalmente, aos problemas de segurança subjacentes. As novas propostas existentes nesta área, aliadas às tecnologias de segurança em grande evolução, podem gerar transformações importantes na área da Gestão de Identidade.

Com este projeto tivemos a intenção de mostrar uma nova realidade, facilmente alcançável e baseada apenas em tecnologias existentes, que permite aos utilizadores alcançar um maior controlo sobre a gestão da sua identidade e garantindo, simultaneamente, fortes propriedades de segurança.

Num futuro próximo, é provável que esta abordagem, ou outras semelhantes, passem a constituir os ‘standards’ da Gestão de Identidade, centrada no utilizador. O fator que mais influenciará a sua aceitação será, possibilidade, o dos utilizadores acederem às aplicações de forma simples, sem necessidade de memorizar inúmeras credenciais distintas. Por outro lado, sendo o receio de perca da privacidade um dos grandes fatores de resistência dos utilizadores à utilização das TICs, devido à fácil perca de controlo sobre a sua informação pessoal, será possível a utilização de um sistema, por eles gerido, que limita o acesso abusivo às informações pessoais, que irá constituir um fator de adoção inquestionável.

Após toda a investigação realizada e o trabalho desenvolvido, chegámos à conclusão que é possível adaptar, com grande facilidade, todo este ecossistema à escala real. O CC cumpre, perfeitamente, todos os requisitos para a gestão e criação de uma identidade digital pessoal e segura.

Mesmo que esta plataforma seja apenas uma prova de conceito, é possível refletir sobre a sua evolução, sendo um das propostas mais imediatas a de integrar outros tipos de dispositivos, como por exemplo, os que suportam a norma FIDO.

É necessário salientar outro fator importante, a facilidade de interação com o utilizador. O protótipo mostra que o registo de um utilizador pode evoluir do atual processo moroso, que necessita o preenchimento de vários formulários, para uma interação única, constituída apenas pela introdução de um pin que liberta as credenciais contidas no CC. São estes pequenos detalhes que fazem com que um projeto tenha as condições para ser adotado em massa pelos utilizadores.

Este protótipo ainda tem um longo caminho pela frente, uma vez que foram desenvolvidas apenas as funcionalidades mínimas, os testes realizados foram somente unitários e, como já foi referido, apenas suporta o CC. Uma vez que a solução seja validada em ambiente de produção real, e certificada relativamente às normas de segurança

associadas aos protocolos utilizados, poderá ser facilmente integrada em qualquer sistema ou tecnologia.

## 9. Referências

- [1] Laurent, M, Bouzefrane, S., Digital Identity Management , 2015, ISTE Press - Elsevier, ISBN-10: 1785480049
- [2] CHAP - Challenge Handshake Authentication Protocol, <https://tools.ietf.org/html/rfc1994>.
- [3] Pareccki, A., "OAuth2 Simplified", on line document, July 2012, <https://aaronparecki.com/2012/07/29/2/oauth2-simplified>.
- [4] Documentação on-line do Cartão do Cidadão, [https://www.cartaoecidadao.pt](https://www.cartaodecidadao.pt).
- [5] OpenID Foundation, <https://openid.net/connect/>
- [6] OAuth 2.0 Authorization Framework, <https://tools.ietf.org/html/rfc6749>
- [7] Podio, Fernando L. (2012-09-05). "Published International Biometric Standards Developed by ISO/IEC JTC 1/SC 37 – Biometrics"
- [8] Windley, P.: Digital Identity. O'Reilly Media, Inc. (2005)
- [9] M.Tariq Banday: "Ensuring Authentication and Integrity of Open Source Software using Digital Signature" (2011)
- [10] Fielding, Roy T.; Gettys, James; Mogul, Jeffrey C.; Nielsen, Henrik Frystyk; Masinter, Larry; Leach, Paul J.; Berners-Lee, Tim (June 1999). *Hypertext Transfer Protocol -- HTTP/1.1*. IETF. RFC 2616
- [11] Mealling, M.; Denenberg, R., eds. (August 2002). "Report from the Joint W3C/IETF URI Planning Interest Group: Uniform Resource Identifiers (URIs), URLs, and Uniform Resource Names (URNs): Clarifications and Recommendations"
- [12] JSON ORG, <http://json.org/json-pt.html>
- [13] Cookies in JavaScript, <http://www.yourhtmlsource.com/javascript/cookies.html>
- [14] Auth0, Passwordless, <https://auth0.com/passwordless>
- [15] P. Mishra; et al. (May 2003), Differences between OASIS Security Assertion Markup Language (SAML) V1.1 and V1.0, OASIS, sstc-saml-diff-1.1-draft-01, retrieved 7 April 2011
- [16] WS Federation, <https://msdn.microsoft.com/en-us/library/bb498017.aspx>
- [17] Network working Group RFC 4511, <https://tools.ietf.org/rfc/rfc4511.txt>
- [18] MSDN Library. Microsoft, [https://technet.microsoft.com/en-us/library/cc780036\(WS.10\).aspx#w2k3tr\\_ad\\_over\\_qbjd](https://technet.microsoft.com/en-us/library/cc780036(WS.10).aspx#w2k3tr_ad_over_qbjd)
- [19] RFC 6238 – TOTP, Time-Based One-Time Password Algorithm, <https://tools.ietf.org/html/rfc6238>
- [20] RFC4226: HOTP: An HMAC-Based One-Time Password Algorithm, <http://www.ietf.org/rfc/rfc4226.txt>
- [21] QR Code, [https://pt.wikipedia.org/wiki/C%C3%B3digo\\_QR](https://pt.wikipedia.org/wiki/C%C3%B3digo_QR)

- [22] "The Open Group Base Specifications Issue 7, section 4.16 Seconds Since the Epoch",  
[http://pubs.opengroup.org/onlinepubs/9699919799/basedefs/V1\\_chap04.html#tag\\_04\\_16](http://pubs.opengroup.org/onlinepubs/9699919799/basedefs/V1_chap04.html#tag_04_16)
- [23] About the FIDO Alliance, <https://fidoalliance.org/about/overview/>
- [24] Specifications Overview, FIDO Alliance,  
<https://fidoalliance.org/specifications/overview/>
- [25] Trusted Computing Group, <https://trustedcomputinggroup.org/tpm-library-specification/>
- [26] About Smart Cards: Introduction: Primer". Secure Technology Alliance. Retrieved 7 August 2017.
- [27] Hyperterxt Transfer Protocol, <http://www.ietf.org/rfc/rfc2616.txt>
- [28] Access Tokens, <https://msdn.microsoft.com/en-us/library/Aa374909.aspx>
- [29] Leicher A., Schmidt A.U., Shah Y. (2012) Smart OpenID: A Smart Card Based OpenID Protocol. In: Gritzalis D., Furnell S., Theoharidou M. (eds) Information Security and Privacy Research. SEC 2012. IFIP Advances in Information and Communication Technology, vol 376. Springer, Berlin, Heidelberg, DOI: [https://doi.org/10.1007/978-3-642-30436-1\\_7](https://doi.org/10.1007/978-3-642-30436-1_7)
- [30] E-Id Authentication and Uniform Access to Cloud Storage Service Providers, Proc IEEE International Conf. on Cloud Computing Technology and Science - CloudCom, Bristol, United Kingdom, Vol. 5, pp. 1 - 10, December, 2013.