UNIVERSIDADE DE LISBOA
FACULDADE DE CIÊNCIAS
DEPARTAMENTO DE INFORMÁTICA

Single Sign-On 2.0: Autenticação Centralizada e Integração com a Chave Móvel Digital

Francisco Loureiro Castel-Branco

Mestrado em Segurança Informática

Trabalho de Projeto orientado por:
Prof. Doutor Hugo Alexandre Tavares MIranda

2021

UNIVERSIDADE DE LISBOA
FACULDADE DE CIÊNCIAS
DEPARTAMENTO DE INFORMÁTICA

Single Sign-On 2.0: Autenticação Centralizada e Integração com a Chave Móvel Digital

Francisco Loureiro Castel-Branco

Mestrado em Segurança Informática

Trabalho de Projeto orientado por:
Prof. Doutor Hugo Alexandre Tavares MIranda

2021

Resumo

Este projeto tem como principal objetivo a integração da autenticação de dois fatores através da Chave Móvel Digital no sistema de *Single Sign-On* em utilização na Faculade de Ciências da Universidade de Lisboa.

A utilização de autenticação de dois fatores proporciona uma segunda camada de segurança tanto ao utilizador final, como à instituição responsável pelos serviços. Assegura-se que, para o utilizador obter autorização de acesso aos diversos conteúdos disponíveis, necessita de duas componentes: algo que sabe, por exemplo, o email e a respetiva palavra-chave, e algo que tem, como o seu telemóvel.

O Autenticação.gov é um serviço fornecido pela Agência para a Modernização Administrativa (AMA), que é um órgão oficial da República Portuguesa, que permite instituições autorizadas a verificar a autenticidade de um cidadão.

A Chave Móvel Digital (CMD) proporciona autenticação de dois fatores. O utilizador apenas tem que conhecer o seu número de telemóvel associado ao Número Único de Cidadão (NUC) e que, ao indicá-lo na plataforma da Autenticação.gov, irá receber e autenticar-se com uma senha descartável.

Palavras-chave: single sign-on, oauth, autenticação, cartão de cidadão, aplicações web

Abstract

This project was carried in the scope of the Information Security Project, integrating the Master's degree program in Information Security on Faculdade de Ciências da Universidade de Lisboa.

It's primary objective was to integrate Two Factor Authentication (2FA) using *Chave Móvel Digital* (Digital Mobile Key) with the previously Single Sign-On system in use by this institution.

Multi-factor authentication in general, and 2FA in particular, provides a new layer of security to the end-user and to the institution which hosts services. It ensures that, for a user to be granted with permissions to access the content, he needs two components: something he knows, for instance an email and password, and something he has, as a received one-time password via SMS in his cellphone.

Autenticação.gov is a service provided by the Agency for the Administrative Modernization (AMA), an official Portuguese Government entity. It allows authorized institutions to delegate the authentication mechanism to AMA, passing through the official government registries.

The Autenticação.gov allows two-factor authentication, simplifying it by using the user's phone number and a password which will be sent via SMS by Autenticação.gov to the associated phone number, ensuring its veracity and legitimacy.

Keywords: single sign-on, oauth, authentication, portuguese citizen card, web applications

Índice

Lista de Figuras

Capítulo 1

Introdução

O método de autenticação mais conhecido é utilizando a combinação de um identificador, por exemplo o *email*, e de uma senha que apenas o próprio utilizador conhece. A cifra não reversível das senhas dos utilizadores é um procedimento habitual que tem como objetivo dificultar a descoberta da senha por terceiros. No entanto, conhecendo o algoritmo que é utilizado na cifra, é possível encontrar o conteúdo de uma senha por tentativa e erro, testando as várias combinações possíveis de senhas, comparando-as com os valores cifrados. À medida que a capacidade de processamento informático se torna cada vez mais eficiente, torna-se naturalmente mais rápido de se executar este processo de tentativa e erro.

Como alternativa à alteração do algoritmo de cifra em ambiente de pro-dução, pode-se considerar a utilização de algoritmos mais complexos. Infelizmente, tornam o processo de autenticação de um utilizador num serviço mais demorado e mais custoso para o fornecedor de serviços. Mesmo assim não é garantido que, na eventualidade de um ataque direto, por exemplo, iludindo o utilizador com um portal comprometido, não seja possível obter as suas credenciais.

Para garantir que, mesmo com acesso às credenciais comuns do utilizador, um atacante não consiga aceder às áreas protegidas, acrescentam-se métodos de autenticação. Com esta adição, a autenticação passa a ser de dois ou mais fatores.

A prática da autenticação de dois fatores, *Two Factor Authentication (2FA)*, é cada vez mais comum. É caracterizada pela verificação de duas características diferentes do utilizador, que podem ser de três tipos: algo que se sabe, como a palavra-chave, algo que se tem, como uma senha descartável, e algo que se é, como a impressão digital.

Esta dissertação tem por objetivo demonstrar a implementação de 2FA num sistema te autenticação existente, reforçando, assim, a segurança da organização e dos seus utilizadores.

1.1 Motivação

Um sistema de *Single Sign-On* (SSO), permite ao utilizador final executar a sua autenticação apenas uma vez para todos os serviços que são disponibilizados por uma organização. Simplifica e melhora a experiência de utilização por, cada vez que o utilizador pretender autenticar-se em diferentes serviços, a sua identidade ser automaticamente detetada e validada, podendo prosseguir sem haver necessidade de introduzir novamente as suas credenciais, evitando, assim, trocas de mensagens pela rede desnecessárias com este conteúdo sensível.

Uma autenticação de dois fatores garante que um utilizador demonstre, a uma organização, ter duas características necessárias para aceder aos seus serviços. A forma mais comum de concretização, é uma senha descartável enviada por mensagem de texto para o telemóvel do utilizador. Isto aumenta o grau de confiança que as organizações têm relativamente à autenticação. A organização tem a garantia que o utilizador não só sabe as suas credenciais, como também garante que possui algo que o identifica como tal. Assim, não só as

organizações têm uma camada extra de segurança, como também os utilizadores têm a garantia que as suas credenciais não são extraviadas e utilizadas sem o seu consentimento.

A implementação do sistema de autenticação Chave Móvel Digital (CMD) fornecida pela Agência para a Modernização Administrativa (AMA), através da plataforma Autenticação.gov, é uma exemplo de autenticação de dois fatores (2FA) que associa o Número Único de Identificação (NIC), como o número do Cartão de Cidadão, ao número de telemóvel dos utilizadores e a um código (PIN). Para realizar autenticação com a CMD, um utilizador sabe o seu número de telemóvel e o PIN, e, após indicar estas credenciais, recebe por SMS, uma senha de utilização única, tornando-se em algo que o utilizador possui.

Para além da camada extra de autenticação, a Autenticação.gov garante que o SSO está, de facto, a lidar com um cidadão válido. Desta forma, passa a deixar de ser necessário que utilizador tenha que saber mais do que o seu número de telemóvel e um pequeno código PIN.

1.2 Objetivos

O principal objetivo deste projeto é aumentar o nível de segurança das informações protegidas pela Faculdade de Ciências da Universidade de Lisboa, motivando o utilizador a recorrer ao uso da Chave Móvel Digital para se autenticar.

Pretende-se, inicialmente, que seja realizada uma atualização do sistema de autenticação atual, garantido que as novas funcionalidades e correções de segurança estejam devidamente implementadas.

De seguida, é necessário compreender as possibilidades de delegação do Sistema de Autenticação Central, *Central Authentication System* (CAS), a outra plataforma de autenticação, como a Autenticação.gov, e definir qual o protocolo a ser utilizado.

Para situações em que não é possível realizar autenticação utilizando CMD ou CC, por exemplo quando as contas não representam pessoas, é pretendido criar-se um método de verificação da validade da palavra-chave do utilizador que, ao estar expirada, deve disponibilizar uma interface de redefinição da mesma. Também se pretende implementar um sistema de Notificações.

Por fim, é necessário normalizar os registos (*logs*) que o CAS produz e implementar testes de validação do software criado nesta dissertação, em conjunto com a documentação necessária para dar continuidade à utilização e suporte do projeto.

1.3 Instituição de acolhimento

A Faculdade de Ciências da Universidade de Lisboa, (Ciências), é uma das escolas da Universidade de Lisboa, (ULisboa) foi fundada a 11 de Abril de 1911 e conta com mais de 5000 alunos. Ciências assume a herança histórico-cultural e científica das suas antecessoras: o Noviciado da Cotovia, o Real Colégio dos Nobres e a Escola Politécnica.

A Direção dos Serviços Informáticos, DSI[1], é uma unidade de serviço de Ciências que opera sobre os serviços de informação disponibilizados aos alunos e colaboradores por meios informáticos, como a gestão de infraestruturas técnológicas e a disponibilização de serviços *e-learning*, promovendo a sua utilização. Tem como principais competências o suporte aos laboratórios e utilizadores, a gestão dos sistemas de informação, administração de redes e outros.

1.4 Estrutura do documento

Este documento está organizado no seguinte formato:

No Capítulo 2: Estado da Arte e Conceitos, são abordadas as tecnologias e conceitos necessários para a plena interpretação dos conteúdos desta dissertação.

No Capítulo 3: Autenticação em Ciências, é analisada a estrutura do sistema de autenticação implementada previamente ao desenvolvimento deste projeto.

No Capítulo 4: Alterações ao processo de autenticação em Ciências, são analisadas as diversas alternativas de implementação do sistema de autenticação pretendido, assim como os protocolos utilizados nas mesmas e a sua implementação.

No Capítulo 5: Avaliação, é analisada a adesão ao novo método de autenticação por parte dos utilizadores, assim como o desempenho do novo sistema de autenticação.

No Capítulo 6: Trabalho Relacionado, é demonstrado o desenvolvimento e aplicabilidade de serviços e conceitos utilizados durante a execução deste projeto.

No Capítulo 7: Conclusão, são apresentadas as conclusões, dificuldades e trabalho futuro relativamente à elaboração deste tema.

Capítulo 2

Estado da Arte e Conceitos

Neste capítulo irão ser analisados diferentes conceitos e implementações fundamentais para a compreensão e desenvolvimento deste projeto.

A tecnologia sobre a qual o sistema de autenticação e respetivos componentes são desenvolvidos ditam a forma como os diferentes protocolos podem ser aplicados. Assim, torna-se necessário conhecer a arquitectura do sistema de autenticação e os diferentes protocolos que foram utilizados durante o desenvolvimento deste projeto.

2.1 Spring Boot

O *Spring Boot*[2] é uma ferramenta de desenvolvimento construída sobre a linguagem de programação *Java*. Propõe diminuir a complexidade de desenvolvimento que os programadores encontram, permitindo a criação de aplicações prontas para produção de forma rápida e eficaz.

É estruturado sobre o conceito de *Inversion of Control* (IoC), que define fluxos de processos, fazendo com que os programadores possam ter um controlo reduzido sobre uma aplicação ao utilizar dependências que já definam um determinado fluxo.

Apesar do IoC significar falta de controlo em componentes específicos, claramente não se reflete numa desvantagem. Esta ideologia permite uma maior modularização das aplicações, permitindo uma preocupação no desenvolvimento muito mais focada em determinados comportamentos dos diferentes componentes, como o desenho de testes unitários. Permite, também, uma agilização do processo de desenvolvimento por utilizar a importação de dependências como forma de definir uma aplicação, substituindo, em alguns casos, o tradicional método de importar bibliotecas e utilizá-las diretamente no código-fonte.

Spring Boot consegue esta inversão de controlo através da injeção de dependências, ou *Dependency Injection*[3]. Este conceito agiliza a criação e utilização de métodos e funções em comparação com as formas tradicionais. O fluxo de trabalho dos programadores é direcionado para um género de declaração global dos objetos e classes, fazendo com que, algures numa outra parte do código, estes componentes possam ser populados sem ser necessário a escrita de código específico para este efeito.

2.2 Hypertext Transfer Protocol

Uma rede de computadores permite a comunicação entre diferentes computadores. É definida por um conjunto de protocolos que garantem formas dos computadores saberem como devem comunicar uns com os outros nessa mesma rede.

O *Hypertext Transfer Protocol* (HTTP), é um dos protocolos mais utilizados da atualidade. Pertence à camada de Aplicação de uma rede de computadores[4] e funciona sobre a camada de transporte *Transmission*

Control Protocol (TCP). Foi inicialmente introduzido em 1991.

Funciona num modelo cliente-servidor e, por norma, é utilizado através de um *user-agent*, como um navegador web ou software de indexação, e é um protocolo que não tem estado.

2.2.1 Pedidos e respostas

A primeira mensagem enviada através deste protocolo chama-se *request*, ou pedido. Um pedido HTTP básico requer que se indique qual o método que se pretende utilizar, a versão do protocolo, o servidor onde se pretende aceder e um identificador do recurso a que se pretende aceder. A este conjunto de dados, chamamos *Unified Resource Identifier* (URI)[5].

Os diferentes métodos disponíveis são OPTIONS, GET, HEAD, POST, PUT, DELETE, TRACE e CONNECT e representam uma forma unificada de indicar que tipo de operação se pretende realizar sobre um determinado recurso. Desta forma, os servidores podem realizar diferentes operações sobre o mesmo recurso indicado através do URI.

```
1    GET /conteudo/pagina.html HTTP/1.1
2    Host: ciencias.ulisboa.pt
```

Listagem 2.1: Exemplo de pedido HTTP

No exemplo de Listagem 2.1, podemos verificar a estrutura do conteúdo do pedido enviado do cliente ao servidor.

Para se obter o ficheiro `pagina.html`, define-se que o método é o GET, a localização completa em `/conteudo/pagina.html` e o protocolo, neste caso a versão 1.1 do protocolo HTTP. Na segunda linha, indicamos o servidor onde pretendemos aceder ao recurso.

De forma a estruturar um pedido de uma forma mais simples, é utilizado um URI, onde se indica os três componentes chave para uma ligação HTTP, assumindo que o método GET é o predefinido.

```
1 http://ciencias.ulisboa.pt/conteudo/pagina.html
```

Listagem 2.2: URI que representa o pedido do Listagem 2.1.

Assim, podemos afirmar que o protocolo HTTP segue a uniformização de localização de recursos como representado na Listagem 2.3.

```
1 URI = [protocolo]://[servidor]/[caminho para o recurso]
```

Listagem 2.3: Representação abstrata de um URI básico

Assumindo que o ficheiro está disponível, o servidor irá responder com uma pacote semelhante ao da Listagem 2.4.

```
1  HTTP/1.1 200 OK
2  Date: Mon, 27 Jul 2020 13:18:43 GMT
3  Last-Modified: Wed, 22 Jul 2020 20:00:00 GMT
4  Content-Length: 88
5  Content-Type: text/html
6  Set-Cookie: fcul=abc123
7  Connection: Closed
8
9  <html>
10 <body>
11 <h1>Hello, World!</h1>
12 </body>
13 </html>
```

Listagem 2.4: Exemplo de resposta a um pedido HTTP

Para contornar a ausência de estado do protocolo, o método utilizado para se identificar um utilizador é, por exemplo, através de *cookies* guardados no user-agent.

2.2.2 Cookies

Como o protocolo HTTP não mantém o estado das ligações, uma das formas do servidor manter informações sobre cada um dos *user-agents* é sob a forma de *HTTP Cookies*, vulgarmente designados de *Cookies*.

Como pode ser verificado na linha 6 do Listagem 2.4, a resposta do servidor inclui um parâmetro que define o *cookie* `fcul` com o valor `abc123`. Este conjunto de chave-valor permite, ao ser utilizado num pedido posterior, que o servidor identifique inequivocamente o *user-agent* nos pedidos seguintes, através do cabeçalho `Cookie` do pedido HTTP. A utilização de identificadores como *cookies*, é o que permite a um servidor saber quem está a realizar os diferentes pedidos para proceder à utilização dos seus mecanismos de autorização.

2.2.3 Segurança

A solução mais comum para garantir confidencialidade e integridade da troca de mensagens, é utilizar *Transport Layer Security* (TLS), que dificulta o processo de interpretação e modificação das mensagens. Esta medida previne ataques dos seguintes tipos:

- *Eavesdropping* é a ação de intercetar mensagens privadas alheias sem o consentimento dos intervenientes.
- *Theft of User Authentication Information* é a ação de extraviar informação sobre a autenticação de um utilizador, nomeadamente as suas credenciais
- *Man-in-the-Middle* descreve a interceção e adulteração das mensagens durante o processo de troca de mensagens sem que nenhum dos principais intervenientes tenha conhecimento desta ação.
- *Theft of the Bearer Token* consiste na obtenção de *access tokens*, ou *cookies*, que identificam um utilizador quando acede a recursos na *web*. Este *token* pode ser utilizado maliciosamente para personificar o utilizador que representa.

2.3 Single Sign-on / Single Log-out

Um sistema de *Single Sign-on*, SSO, propõe unificar e centralizar a autenticação dos utilizadores de uma organização, substituindo a tradicional autenticação em cada serviço. Desta forma, e apenas no primeiro momento necessário, os utilizadores são encaminhados para uma página, correspondente ao SSO, onde se podem autenticar.

A palavra *Single* do SSO promete que a autenticação apenas tenha que ser realizada uma vez. Esta característica é conseguida porque todos os serviços encaminham o utilizador para o mesmo sistema. Ao já ter sido executada, a autenticação deixa de ser necessária, redirecionando o utilizador de volta ao serviço de forma transparente.

Assim, podemos afirmar que os SSO são um sistema independente dos outros serviços visto terem uma aplicação muito específica e focada na segurança da entidade que os utilizam. Este sistema permite, principalmente, que os serviços nunca tenham qualquer tipo de interação com a palavra-chave do utilizador e também centraliza as políticas de autenticação pretendidas.

Nesta secção, são abordados os conceitos fundamentais deste tipo de aplicação.

2.3.1 *Identity Provider*

Um *Identity Provider* (IdP) pode, ou não, ser centralizado e é-lhe atribuída a funcionalidade de identificar o utilizador perante um determinado domínio ou organização. Um sistema de SSO é um IdP que se certifica que o utilizador existe e devolve estas características, chamadas atributos, aos diferentes serviços autorizados.

Existe, também, o conceito de Identidade Federada, onde, para aceder a um determinado recurso, ou serviço, o utilizador tem que se autenticar perante uma determinada Federação. Este conceito permite que utilizadores de diferentes organizações, com sistemas de SSO diferentes, que pertençam à mesma federação, consigam partilhar os mesmos recursos ou serviços.

Quando um utilizador se autentica num sistema de SSO, o IdP fornece os atributos do utilizador aos diferentes serviços autorizados a utilizar este mecanismo, chamados de *Service Providers*.

Os sistemas de SSO podem, também, atuar como *Service Provider* (SP) se a ação pretendida for delegar a autenticação a outro SSO. Ainda assim, após uma autenticação delegada bem sucedida, o SSO pode utilizar as informações do utilizador obtidas através desta delegação para encontrar informações adicionais na própria organização.

2.3.2 *Service Provider*

Um *Service Provider* (SP), é um serviço responsável por proteger recursos e disponibiliza-los aos utilizadores.

Para disponibilizar recursos protegidos, um SP utiliza um SSO para obter a informações sobre um utilizador e autoriza-lo mediante os dados obtidos.

2.3.3 SAML

Security Assertion Markup Language[6], ou SAML, é um protocolo aberto para troca de autenticação e autorização usando a sintaxe XML.

No início de 2001, a *OASIS Security Services Technical Committee* acreditou ser necessária a criação de uma arquitetura XML para troca de informação de autorização e autenticação. Em Novembro de 2002, surgiu o protocolo SAML, na versão 1.0. O protocolo SAML 2.0, lançado em 2005, conta com extensões e implementações de segurança mais recentes e atualizadas. A partir de 2008, a utilização deste protocolo começou a ser bastante comum em órgãos governamentais, empresas e ensino superior[7].

A arquitectura do protocolo SAML agrupa os diferentes componentes de autenticação, atributos e autorização em quatro categorias hierárquicas para criar uma relação de confiança. Definem-se *profiles* como combinações de declarações e projeções de atributos e protocolos de pedido/resposta. Dentro de *profiles*, *bindings* é o subconjunto de definições que indica quais os protocolos que devem ser utilizados na troca de mensagens, por exemplo definir o método HTTP GET ou HTTP POST. Em *protocols*, define-se quais as características da troca de mensagens, por exemplo o tipo de síntese, ou *digest*, que a mensagem irá conter. Por fim, *assertions* define o conjunto de atributos resultantes de uma autenticação, por exemplo o endereço de email do utilizador.

Protocolos

O SAML disponibiliza diversos protocolos relacionados com as diferentes etapas dos processos de autenticação[8]. Entre estes, os mais utilizados são o *Authentication Request Protocol* e *Single Logout Protocol*, sendo, este último, a abordagem necessária para disponibilizar autenticação federada.

O *Authentication Request Protocol* define a forma como um agente delega a autenticação a um IdP, enviando os vários parâmetros durante a troca de mensagens. É na fase final desta delegação de autenticação que verificamos se o utilizador existe num determinado domínio.

Single Logout Protocol é um mecanismo que permite que um utilizador termine a sessão em diversos serviços de forma quase simultânea. Esta ação pode ser executada pelo próprio utilizador, IdP ou SP, ou pode resultar de operações administrativas ou temporizadores.

É possível utilizar estes protocolos de diferentes formas. Os *Bindings* definem a forma como as mensagens irão ser trocadas ao nível da camada de transporte no protocolo HTTP. O SAML permite a utilização de redirecionamentos, de submissão de informação através do método POST e execução de procedimentos remotos por *Simple Object Access Protocol*, SOAP.

Destas, os componentes mais utilizados são *HTTP POST Binding*, *HTTP Redirect Binding* e *HTTP Artifact bindings*, visto serem de mais simples implementação e, em *browsers*, o suporte adicional é irrelevante por serem utilizarem métodos e cabeçalhos comuns do protocolo HTTP.

***HTTP POST Binding* e *HTTP Redirect Binding* iniciado pelo SP**

O método POST consiste numa submissão, no corpo do pedido HTTP, com a chave `SAMLRequest` preenchida com o corpo da mensagem de pedido de autenticação, `<AuthnRequest>`, juntamente com o estado da aplicação na chave `RelayState`, codificada em *base64*. A resposta irá ter, também, este formato.

```
1 POST /SAML2/SSO/POST HTTP/1.1
2 Host: idp.example.org
3 Content-Type: application/x-www-form-urlencoded
4 Content-Length: nnn
5 SAMLRequest=request&RelayState=token
```

Listagem 2.5: Cabeçalho do pedido HTTP POST

Ao ser inicialmente acedido, o SSO determina se o utilizador já tem alguma sessão iniciada no domínio do IdP, recorrendo aos *cookies* apresentados pelo *user-agent*. Se tal não se verificar, o utilizador irá receber um formulário para se autenticar com as suas credenciais. Após este passo, o IdP cria uma mensagem de resposta SAML, que é digitalmente assinada e contém um identificador `source ID`. Este último identifica um artefato no IdP que pode ser acedido pelo SP para obter informações sobre esta autenticação.

Todo este procedimento é semelhante ao fluxo encontrado na Figura 2.1, que representa o *HTTP Redirect Binding*.

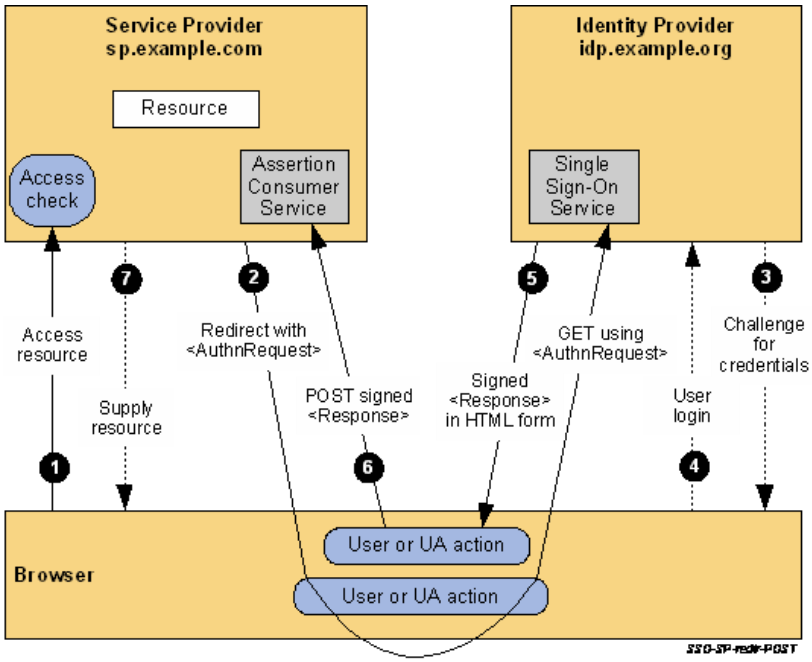

Figura 2.1: Processo de autenticação utilizando *HTTP Redirect Binding*

Assertions

Assertions é a componente do protocolo SAML que procura cruzar informações de um determinado agente do processo. Existem três tipos diferentes:

- *Authentication statements* são criados pelo sistema que autenticou um utilizador e, no mínimo, contém um identificador do mesmo e uma marcação que identifica esta ação no tempo;
- *Attribute statements* contém informação específica sobre o utilizador identificado, normalmente chamados atributos.
- *Authorization decision statements* procuram saber se o utilizador tem permissões para executar uma determinada ação.

Metadados

A flexibilidade do protocolo SAML em definir diversos tipos de síntese, cifra, certificados e outras configurações, implica uma maior diversidade na combinação de configurações que cada IdP pode disponibilizar. De forma a agilizar o processo de configuração de SP e IdP, é possível definir-se os requisitos através de uma forma estruturada designada de metadados[9].

```
1     <saml2:Assertion xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion
         "
2     ID="ID_ab0392ef-b557-4453-95a8-a7e168da8ac5" IssueInstant="
         2010-09-30T19:13:37.869Z"
3     Version="2.0">
4     <saml2:Issuer>http://idp.example.com </saml2:Issuer>
5     <saml2:Subject>
6         <saml2:NameID NameQualifier="urn:picketlink:identity-federation
             ">jduke</saml2:NameID>
7         <saml2:SubjectConfirmation Method="urn:oasis:names:tc:SAML:2.0
             :cm:bearer" />
8     </saml2:Subject>
9     <saml2:Conditions NotBefore="2010-09-30T19:13:37.869Z"
10         NotOnOrAfter="2010-09-30T21:13:37.869Z" />
11    <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
12        <ds:SignedInfo>
13            <ds:CanonicalizationMethod Algorithm="http://www.w3.org
                 /2001/10/xml-exc-c14n#WithComments" />
14            <ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/
                 xmlds#rsa-sha1" />
15            <ds:Reference URI="#ID_ab0392ef-b557-4453-95a8-a7e168da8ac5
                 ">
16                <ds:Transforms>
17                    <ds:Transform Algorithm="http://www.w3.org/2000/09/
                         xmlds#enveloped-signature" />
18                    <ds:Transform Algorithm="http://www.w3.org/2001/10/
                         xml-exc-c14n#" />
19                </ds:Transforms>
20                <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/
                     xmlds#sha1" />
21                <ds:DigestValue>0Y9QM5c5qCShz5UWmbFzBmbuTus=</
                     ds:DigestValue>
22            </ds:Reference>
23        </ds:SignedInfo>
24        <ds:SignatureValue>se/(...))Mtt7A=</ds:SignatureValue>
25        <ds:KeyInfo>
26            <ds:KeyValue>
27                <ds:RSAKeyValue>
28                    <ds:Modulus>
29                        suGI(...)0dfNPbs=
30                    </ds:Modulus>
31                    <ds:Exponent>AQAB</ds:Exponent>
32                </ds:RSAKeyValue>
33            </ds:KeyValue>
34        </ds:KeyInfo>
35    </ds:Signature>
36 </saml2:Assertion>
```

Listagem 2.6: Metadados de um IdP SAML

A definição dos metadados exemplificada na Listagem 2.6 permite saber que, para utilizar este IdP, é necessário utilizar um algoritmo de síntese SHA1 e que a assinatura é feita através de chaves RSA em conjunto com essa mesma cifra. A partilha da especificação permite a utilização de bibliotecas que implementem as diferentes configurações de forma dinâmica, simplificando o processo de configuração.

2.3.4 Autenticação Federada

De forma a permitir autenticação entre diferentes domínios, os sistemas de autenticação federada estabelecem relações de confiança entre diferentes IdP sem ser necessário existir duplicações de credenciais. A autenticação federada permite partilhar recursos de diferentes organizações utilizando sistemas de SSO diferentes que pertençam a um círculo de confiança.

Conforme referido, as diferentes interações do protocolo SAML incluem uma assinatura digital. É através da validação das assinaturas das mensagens com os certificados confiados pela federação que se determina se o SSO é, ou não, confiável.

Numa autenticação federada, os atributos dos utilizadores são projetados num formato unificado e determinado pelo protocolo SAML.

Exemplificando com um utilizador de Ciências que pretenda aceder a um recurso da FCCN. O IdP da FCCN não irá reconhecer o utilizador e, por isso, reencaminha-o para um sistema de SSO em que confie para ser realizada autenticação. Sabendo que a FCCN inclui a Universidade de Lisboa no seu círculo de confiança, e esta última inclui Ciências, o utilizador é redireccionado para o sistema de autenticação a que lhe corresponde. Assim que a autenticação for bem sucedida, o utilizador poderá, então, aceder ao recurso protegido.

2.3.5 OAuth

OAuth é um protocolo de autorização lançado em 2006. A sua arquitetura permite a aplicações terceiras aceder a recursos cujas permissões são definidas por uma primeira. Propõe-se a fornecer um método de delegação de autenticação de um SP através do protocolo HTTP. É cada vez mais utilizado na indústria pela sua simplicidade.

Quase diariamente podemos verificar que determinadas aplicações web nos permitem iniciar sessões com outros serviços, como *Google* ou *Facebook*, que funcionam como *Identity Provider*. Desta forma, torna-se possível iniciar sessão ou registar num serviço sem ser necessário realizar o procedimento habitual de autenticação. Passsa, apenas, a ser necessário estar autenticado num IdP e autorizar o serviço a utilizar as informações pretendidas.

Neste protocolo, um utilizador, ao autenticar-se corretamente num IdP OAuth, é redirecionado para o serviço com um identificador, *token*. Em baixo, são abordados os dois métodos de autenticação deste protocolo, que mostram duas funcionalidades diferentes para este identificador.

Segundo a Secção 3 do RFC 3986[10], este protocolo implementa três *endpoints*, ou URI, diferentes: um de autorização, que é utilizado para redirecionar o cliente para o portal de autenticação, um para obtenção do *token*, normalmente utilizado durante a autenticação do utilizador, e também um *endpoint* de redirecionamento, que é por onde irão ser devolvidas as credenciais.

Vamos abordar dois dos métodos que a especificação define e que são relevantes para este projeto.

Authorization Code Grant

O método de autenticação *Authorization Code Grant* do protocolo *OAuth2* é o método recomendado na maioria dos cenários [11].

Inicia-se o processo de autenticação redirecionando o utilizador para a página de autenticação do IdP, enviando também, no URI, ou *endpoint* de redirecionamento, um identificador do serviço que requer autenticação e o URI para onde o utilizador irá ser redirecionado no final deste processo.

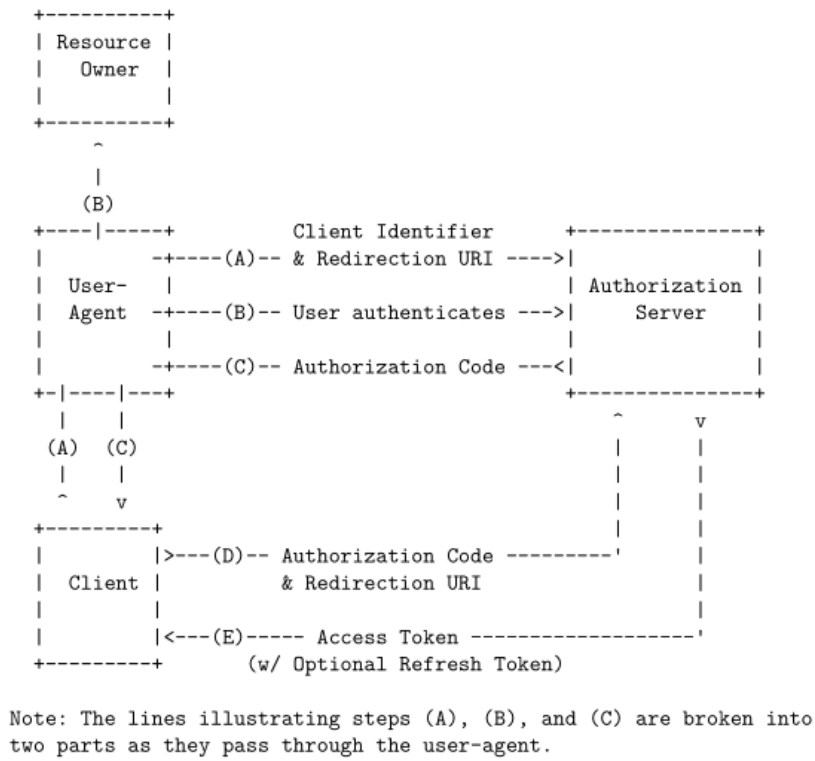

Figura 2.2: Fluxo do método *Authorization Code* do protocolo OAuth2, conforme o RFC 3986[10]

A Figura 2.2 mostra que este método separa os pedidos de autorização dos pedido do *access token*, o que significa que este identificador pode ser atualizado, através de um `refresh token`, *token* de atualização, para que a sessão do utilizador possa ser renovada caso o identificador expire.

Desta forma, antes de obtermos os recursos que pretendemos, são necessárias quatro interações, mesmo que transparentes ao utilizador, com o servidor de autenticação:

1. A aplicação a que se pretende aceder redireciona o utilizador para o servidor de autenticação com o seu próprio identificador único, que valida a mesma sobre o servidor de autenticação, e com o endereço onde se pretende que a autenticação se finalize.

2. O utilizador autentica-se sobre o servidor de autenticação

3. O servidor de autorização valida, ou não, o utilizador e redireciona-o para o serviço com um código de autorização.

4. O serviço utiliza o código de autorização e obtém um código de acesso junto do servidor de autorização.

O código de autorização que permite obter o código de acesso só pode ser utilizado uma única vez. Desta forma, garantimos que o código de acesso que permite obter os atributos do utilizador esteja apenas disponível para o SP.

Após estes passos, o utilizador encontra-se no serviço a que pretende aceder e indica-lhe o seu código de autorização. Aqui, o serviço utiliza este código para obter o *token* de acesso que lhe permitirá gerar algum

tipo de sessão única, abstraindo-se do código de acesso. Atribuímos, assim, um certo grau de confiança ao serviço que utilizará o código de acesso como garantia que o utilizador se autenticou correctamente no servidor de autorização.

Implicit grant

Apresenta-se como uma forma simplificada do método *Authorization Code Grant* por não ser necessário existir um servidor que obtenha os atributos do utilizador.

Ao contrário do anterior, fornece o *access token* diretamente na resposta ao pedido de autorização. Foi desenvolvido com aplicações móveis e serviços de apenas uma página (*Single-Page Application*) em mente por não requerer que o serviço tenha interação com o servidor de autorização.

Para obter um código de acesso aos recursos, procede-se da seguinte forma:

1. O utilizador acede ao servidor de autorização apresentando um identificador de cliente e o URI a que o fluxo de autenticação nos irá reencaminhar.
2. O utilizador autentica-se.
3. O servidor de autorização redireciona o utilizador para o URI indicado no primeiro passo, acrescentando um fragmento com os parâmetros correspondentes à validade da autenticação.

Este código dá ao utilizador acesso direto aos recursos.

Segundo o RFC3986, a resposta do servidor de autorização manifesta-se através do redirecionamento composto por um fragmento identificado pelo símbolo #, que inclui o código de acesso.

```
1 HTTP/1.1 302 Found
2 Location: http://example.com/cb#access_token=2AotnAAAAjr1zCsicAApAA&
     state=xyz&token_type=example&expires_in=3600
```

Listagem 2.7: Resposta HTTP a uma autenticação *OAuth Implicit Grant* bem sucedida

Este método tem como grande desvantagem encontrar-se em claro no fragmento do URI, como se pode verifica na Listagem 2.7. Como resultado, cria a possibilidade de ser conhecido quando se tem acesso ao histórico do utilizador.

2.4 CAS - Central Authentication System

O CAS, *Central Authentication System*[12], é um protocolo de *Single Sign-on* para a web. Pretende unificar a autenticação do utilizador num só ponto de entrada e apenas uma vez para as várias aplicações por este protegidas. Foi desenvolvido sobre *Spring Boot* (abordado na secção 2.1), o que permite uma modularização das diferentes componentes e funcionalidades disponíveis através da definição de diferentes dependências que alteram o seu comportamento podendo requerer, apenas, a atualização do ficheiro de configuração.

É um software extremamente flexivel que suporta diversos protocolos de bases de dados, como bases de dados relacionais (SQL) e LDAP, e de autenticação como OAuth2 e SAML. Anuncia, também, suporte para integrações com Office365, Moodle e Google Apps, monitorização e estatísticas em tempo real, gestão de palavas-chave e 2FA.

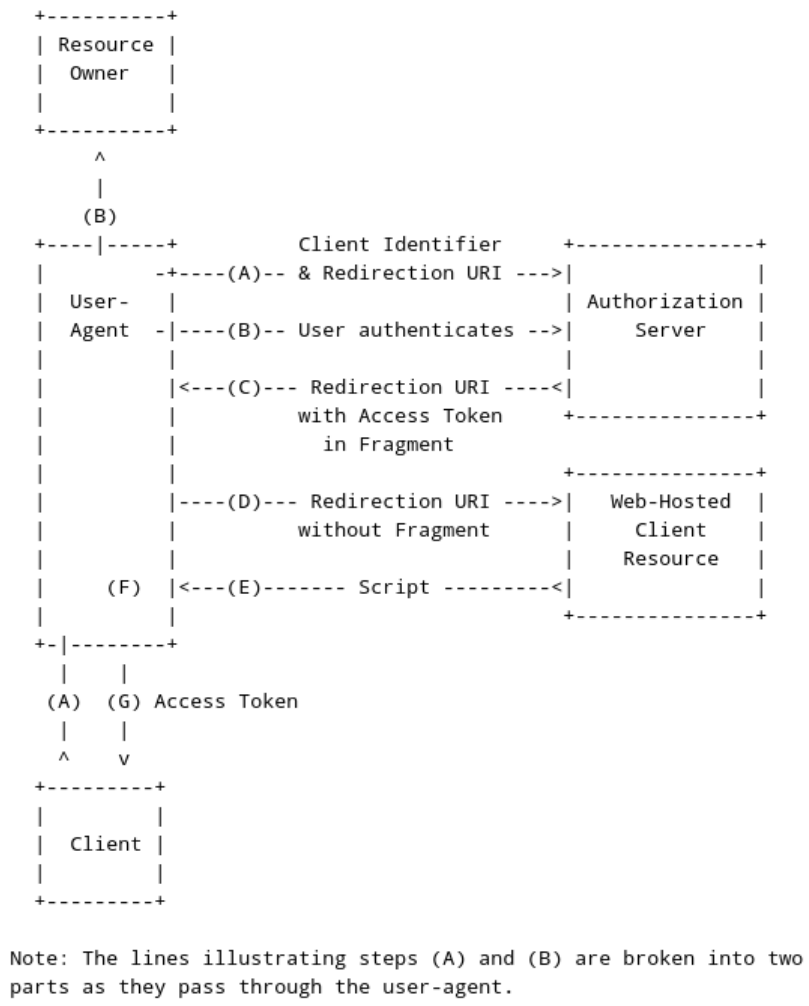

Figura 2.3: Fluxo do método *Implicit Grant* do protocolo OAuth2, conforme o RFC3986 [10]

2.4.1 Arquitetura

O CAS suporta três protocolos diferentes para comunicar com as aplicações que por este são protegidas: (*CAS Protocol*, *SAML* e *OAuth*) para responder aos serviços que o utilizam como *Identity Provider* (IdP).

O principal objetivo do CAS é ser um *Identity Provider* (IdP). Contudo, também permite delegar essa funcionalidade, desempenhando um papel de *Service Provider* (SP) que pede a identidade de um utilizador a outro IdP.

Ao receber um pedido de autenticação com as credenciais de um utilizador, o CAS verifica a sua validade nos diversos locais disponíveis. Na Figura 2.4, a autenticação é feita em Bases de dados relacionais, em LDAP e SPNEGO, mas também é possível definir outros protocolos ou até autenticação básica de utilizador/palavra-chave. Aqui, podemos verificar que, todos os clientes que os utilizadores pretendem aceder, realizam uma autenticação centralizada com o servidor CAS, fazendo com que este seja um intermediário com os diferentes bancos de dados onde é possível existir uma correspondência das credenciais para realizar a autenticação. Os utilizadores não precisam de indicar em que servidor estão estas credenciais, visto que o CAS irá verificar em todas até encontrar informações de autenticação. Caso não existam credenciais válidas, o utilizador não terá uma autenticação bem sucedida.

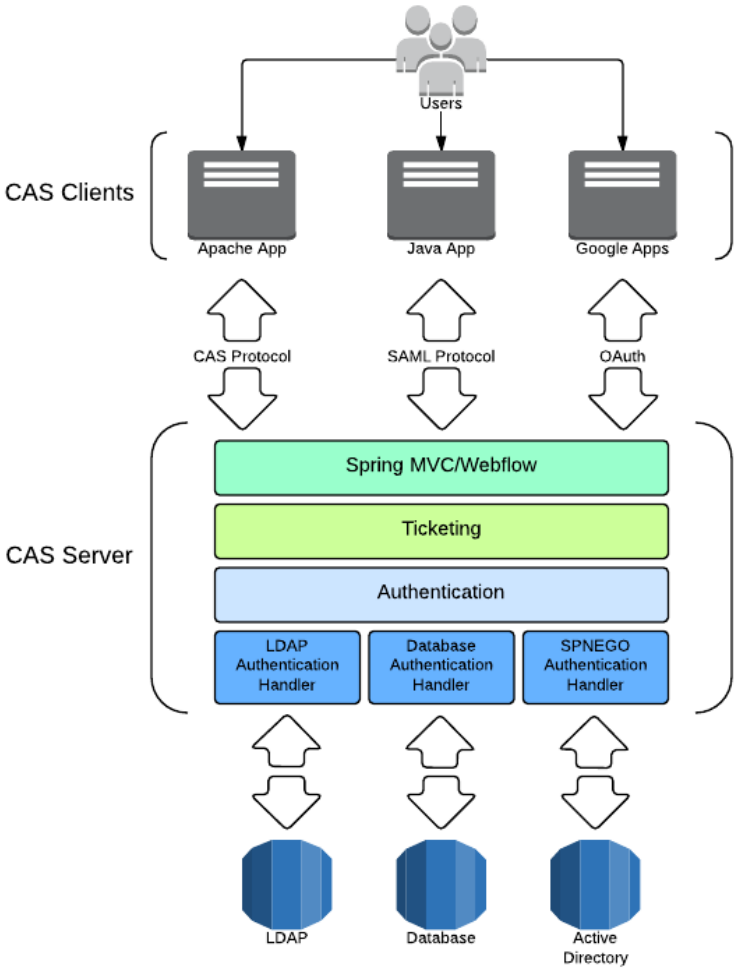

Figura 2.4: Arquitetura do SSO CAS.

2.4.2 *Tickets*

O CAS utiliza um sistema de *tickets* para identificar o utilizador ao longo do fluxo de autenticação e propõem mitigar um conjunto de ataques, primariamente de *Cross-Site Request Forgery* (CSRF). Os dois tipos de identificadores mais importantes neste processo de autenticação são o *Ticket Granting Ticket* (TGT), e o *Service Ticket* (ST).

De forma a identificar o utilizador, o TGT é atribuido ao utilizador, após uma autenticação bem sucedida, sob a forma de *cookie* e é o que torna possível a funcionalidade de *Single Sign-on* e *Single Log-out*. Desta forma, se um utilizador se tentar autenticar noutro serviço protegido pelo SSO, o TGT é validado e o CAS redireciona automáticamente o utilizador para o serviço em que se pretendia autenticar.

Os *Service Tickets* são gerados aleatóriamente e acompanham o redirecionamento de uma autenticação bem sucedida até ao serviço a que se pretende aceder através de parâmetros nos pedidos `GET` do protocolo HTTP. Assim, o serviço consegue interpretá-lo e utilizá-lo como o seu próprio método de validação do utilizador com o sistema de SSO para obter os atributos necessários, e previamente acordados com o administrador do CAS, para dar continuidade à sessão do utilizador.

2.5 Tecnologias relacionadas

Nesta secção são analisados conceitos e tecnologias que permitem uma melhor compreensão dos mecanismos utilizados no desenvolvimento do projeto.

2.5.1 LDAP

O *Lightweight Directory Access Protocol*, ou LDAP, é um protocolo *open-source* de comunicação de serviços de diretoria, como o Active Directory, OpenLDAP e Apache Directory Service. Estes serviços armazenam utilizadores, contas de utilizadores e palavras chave.

Este protocolo disponibiliza vários tipos de operações. Além dos métodos tradicionais de gestão, (pesquisar, adicionar, remover e modificar), existe também a operação *bind* que permite realizar autenticações através de credenciais em texto simples ou por certificados. Este processo é baseado em *Simple Authentication and Security Layer* (SASL).

Diretorias podem ser compostas por entidades de vários tipos, entre os quais unidades organizacionais, utilizadores e equipamentos.

Os utilizadores são identificados por um identificador único denominado *Distinguished Name* (DN), que agrega vários componentes:

1. *Common Name* (CN), é um nome comum dado ao utilizador. Normalmente será preenchida com o seu nome completo. Em Ciências, o seu CN é o seu código identificativo (`fcXXXXX`).

2. *Organizational Unit* (OU), é o nome de uma organização dentro do domínio. Também pode ser utilizado para representar grupos de utilizadores. É aqui que o administrador de sistemas determina onde se situa o utilizador na hierarquia do domínio.

3. *Domain Component* (DC), representa cada parte do domínio. Por exemplo, em Ciências, o domínio é composto por `fc.ul.pt`.

Esta classificação permite estruturar os diferentes elementos em árvore, permitindo que existam Unidades Organizacionais dentro de outras, permitindo, entre outras coisas, que sejam definidas políticas de autorização em aplicações que aproveitem este esquema hierárquico.

A Figura 2.5 representa como dois professores, que exercem funções no Departamento de Informática de Ciências, se encontrariam representados no esquema de árvore. Teriam os DN `CN=Professor Exemplo1, OU=Docentes,OU=DI,DC=fc,DC=ul,DC=pt` e `CN=Professor Exemplo2,OU=Docentes,OU=DI,DC=fc ,DC=ul,DC=pt`.

É, portanto, uma forma estruturada de armazenar utilizadores.

2.5.2 Active Directory

A *Active Directory*, ou AD, é um serviço de diretoria de utilizadores, grupos de utilizadores e serviços. É uma ferramenta proprietária da Microsoft que surgiu com o *Windows 2000*.

A AD tem várias funcionalidades, entre as quais autenticar utilizadores e armazenar os respetivos atributos, permitindo-lhes utilizar computadores de um domínio, serviços de impressão e outros.

Para além de utilizar o protocolo LDAP, também permite a utilização de Kerberos como método de autenticação.

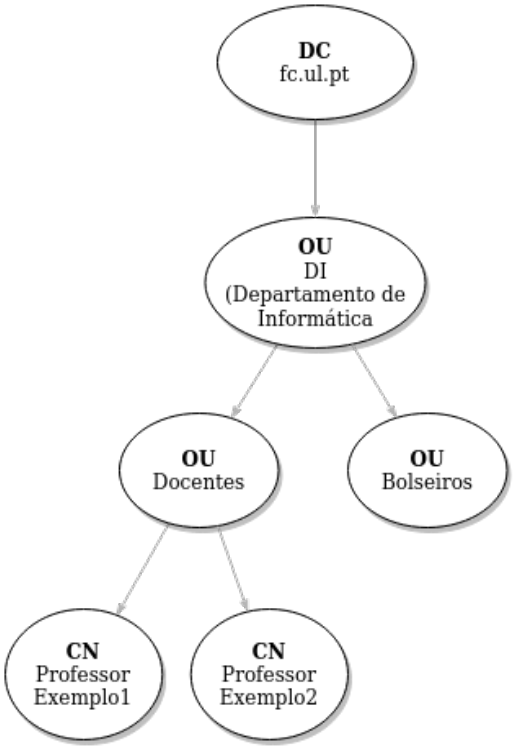

Figura 2.5: Representação visual da arquitetura de utilizadores e unidades organizacionais dentro de um domínio

2.5.3 simpleSAMLphp

O *simpleSAMLphp*[13] é uma ferramenta escrita em PHP que visa facilitar o processo de utilização do protocolo SAML como IdP e SP.

A Faculdade de Ciências da Universidade de Lisboa utiliza esta aplicação de modo a obter autorização para aceder a serviços federados, delegando a sua autenticação ao sistema de SSO.

Capítulo 3

Autenticação em Ciências

Neste capítulo são apresentados os diferentes componentes do sistema de autenticação de Ciências relevantes para o projeto desenvolvido.

3.1 CAS

O CAS funciona como um guarda para a maioria das aplicações de Ciências. Ao aceder a um dos serviços, o utilizador envia uma prova de autorização, conhecido como *token*, que foi obtido previamente através do sistema de autenticação. O serviço confirma com o CAS se o *token* é válido, através de um canal de rede seguro.

Ao confirmar a veracidade deste token, o serviço está autorizado a disponibilizar-se ao utilizador, limitando as permissões de acesso ao conteúdo através dos atributos disponibilizados pelo CAS.

No caso de existir uma falha nesta autorização, o utilizador é redirecionado para o CAS, que disponibiliza uma interface Web com um formulário. Recebendo a sua submissão, comunica com o Active Directory através do protocolo LDAP para verificar as credenciais inseridas pelo utilizador.

Na configuração do CAS, é necessário indicar quais os serviços autorizados a utilizar este sistema de modo a obter os atributos do utilizador, que estão definidos numa base de dados MySQL.

O CAS é, então, considerado uma plataforma de ***autenticação*** e de ***autorização***, embora apenas seja utilizada a primeira. Autentica o utilizador contra uma base de dados, diretório ou outro, autorizando o acesso de um utilizador ao ponto de entrada de um serviço, tendo, agora, a possibilidade de limitar o conteúdo disponibilizado mediante os seus atributos.

3.2 Active Directory e OpenLDAP

Ciências utiliza diretórios de utilizadores para armazenar alguns dados dos mesmos. São utilizadas duas *Active Directory*, tecnologia *Microsoft*, uma para alunos e outra para os restantes utilizadores, e um *openLDAP* para armazenar alunos candidatos à instituição e outras contas temporárias exclusivas do sistema académico.

O *Active Directory* é utilizado para realizar autenticação no CAS, em computadores, disponíveis por todo o *Campus*, na rede virtual privada (VPN) e nos vários ponto de acesso de rede de computadores sem-fios.

3.3 Drupal

O Drupal[14] é um sistema de gestão de conteúdos (CMS) que permite uma fácil edição de paginas, gestão de menus e outros tipos de conteúdo web. Identifica-se como uma plataforma altamente extensível.

É sobre esta tecnologia que se encontra desenvolvido a página web principal de Ciências.

Os vários serviços disponíveis são desenvolvidos numa tecnologia independente chamada CakePHP, também na linguagem de programação PHP, de forma a ter o máximo de controlo possível sobre as aplicações e o seu comportamento e integração com outros serviços disponibilizados pela instituição.

É através dos serviços desenvolvidos para o Drupal que grande parte do trabalho administrativo de Ciências é realizado. Entre estes, encontra-se um painel de administração que permite configurar dinâmicamente os serviços autorizados a utilizar o SSO para autenticar os seus utilizadores (ver Secção 3.1).

Capítulo 4

Alterações ao processo de autenticação em Ciências

O processo de autenticação de Ciências, embora já integrado com um sistema de *Single Sign-on* robusto, requeria um grande conjunto de modificações como forma de respostas aos requisitos propostos, como a autenticação de dois fatores e a integração com a CMD.

Durante o processo inicial de reconhecimento da arquitetura dos sistemas e analisando a integração com a Chave Móvel Digital, foi considerada a possibilidade de transitar o sistema de SSO para a aplicação *Shibboleth*. Esta alteração era justificada pela existência de um *plugin* desenvolvido pela Fundação para Ciências e Tecnologia (FCT), que integrava esta delegação. No entanto, e tendo em conta a quantidade de sistemas desenvolvidos que estão dependentes do protocolo CAS para realizar autenticações, essa possibilidade não se demonstrou viável.

Assim sendo, foi decidido que seria a versão mais recente do CAS a ser modificada para entrar em concordância com os requisitos deste projeto.

4.1 Atualização

O primeiro passo do trabalho realizado foi a decisão de atualização do CAS, visto se encontrar na versão 4, inicialmente lançada em Maio de 2014, para garantir uma melhor compatibilidade de implementações futuras e para melhorar a qualidade geral do CAS. Naturalmente, o CAS conta com correções de erros, atualizações de segurança e melhorias no desempenho. A versão utilizada foi a 6.2.5, por ser a mais recente à data da realização do projeto. Esta versão conta com algumas modificações provenientes de requisitos deste trabalho, conforme irá ser referido neste capítulo.

4.2 Delegação da autenticação

Como forma de reforçar a autenticação, foi decidido incorporar um sistema de 2FA. A escolha recaíu na utilização do Cháve Móvel Digital (CMD).

Como referido anteriormente, a AMA disponibiliza o IdP *Autenticação.gov* que dispõe de interfaces para a delegação de autenticação que podem ser utilizadas pelo CAS de forma a permitir a utilizadores credenciarem-se com a Chave Móvel Digital.

Este método é particularmente vantajoso por, em teoria, impossibilitar a partilha de credenciais, que representam uma falha de segurança.

Mediante análise da documentação fornecida pela Agência para a Modernização Administrativa, AMA, a Autenticação.gov pode utilizar um de dois protocolos, *SAML2* e *OAuth2 Implicit Grant*.

O CAS utiliza a biblioteca de autenticação *pac4j* para executar os pedidos de autenticação a outras entidades. Esta biblioteca é bastante utilizada em aplicações Java e é ativamente desenvolvida.

4.2.1 CAS a delegar SAML

A utilização de *Security Assertion Markup Language* com a Autenticação.gov requer a utilização de um algorítmo de síntese do tipo SHA1. Este tipo de síntese aparenta não ser suportado pela biblioteca *pac4j*, apesar de ser uma especificação lançada em 1995.

Os algorítmos de síntese procuram calcular uma redução de qualquer informação digital para uma *assinatura* dessa mesma informação. Estas sínteses são calculadas através de fórmulas matemáticas e não são reversíveis. É desta forma que se procura armazenar palavras-chave, visto ser muito pouco provável que exista uma colisão de assinaturas entre duas informações digitais diferentes. No entanto, existem algoritmos em que essa colisão foi encontrada, como por exemplo o SHA1.[1]

Pela AMA, foi disponibilizada documentação descritiva, sem recurso a ficheiro de metadados, referente à implementação da autenticação através deste método. Para ser realizada uma integração com o CAS, foi necessário proceder à criação manual deste ficheiro.

O protocolo SAML2 tinha sido o método escolhido inicialmente por ter sido testada uma delegação com o *SimpleSAML* utilizado em Ciências para autenticação federada. Verificou-se, assim, que um dos requisitos para esta integração, não consegue ser cumprido. Este requisito obriga à definição, no pedido de autenticação, dos atributos dos utilizadores que se pretendem obter.

Dada a complexidade de desenvolvimento e testes relativamente à modificação desta biblioteca e integração com uma versão, também ela modificada, do CAS, tornou-se inviável a ”atualização”do *pac4j* para utilizar um *digest* mais fraco e menos utilizado.

4.2.2 CAS a delegar OAuth2

O protocolo *OAuth2* foi criado com o objetivo de agilizar a autenticação entre diferentes websites sem ser necessário o tratamento de mensagens, como no protocolo SAML, podendo também disponibilizar os atributos dos utilizadores, como o nome ou o endereço de correio eletrónico, a uma plataforma sem que esta tenha que armazenar as credenciais de autenticação dos mesmos.

Ao executar a delegação, o CAS encaminha o utilizador para a Autenticação.gov através de um pedido HTTP do tipo GET, indicando na *query* os seguintes parâmetros:

- O tipo de resposta, *response_type*, que se pretende obter do servidor de autenticação, conforme os métodos referidos no Capítulo 2.3.5.
- O identificador (*id*) do serviço que está a requerer autenticação, quais os atributos que se pretende que sejam devolvidos.

OAuth2 na Autenticação.gov

A AMA disponibiliza, na Autenticação.gov, o método de autenticação *Implicit Grant* do protocolo OAuth2. Este método difere do método mais utilizado, *Code Grant*, no seu tipo de resposta. Para ser utilizado, a biblioteca de autenticação *pac4j* e o CAS tiveram que ser modificados por não ser possível modificar o parâmetro `response_type` no endereço de redirecionamento do pedido de autenticação. Estas alterações foram brevemente discutidas e aprovadas pela comunidade *Open Source* que as representa. Deste projeto, surgiram uma sugestão ao projeto *pac4j*[15] e um *Pull Request*[16] no repositório do CAS, aumentando, assim, a flexibilidade destas aplicações para a sua comunidade de utilizadores.

[1] https://security.googleblog.com/2017/02/announcing-first-sha1-collision.html

Em resultado das alterações realizadas ao *pac4j* e ao CAS no âmbito deste projeto, foi possível a AMA fornecer o parâmetro `client_id` que seria necessário para o pedido de autenticação ser válido no seu portal. Desta forma, foi detetado que o portal *Autenticação.gov* não seguia as recomendações da norma RFC [10]. Como já foi referido, o CAS utiliza o sistema de *tickets* para identificar o utilizador ao longo da autenticação. Ao ser delegada, o SSO envia um *ticket* para o IdP e espera que este seja retornado para dar seguimento a este processo. Acontece que o portal *Autenticação.gov* não devolve esta informação ao redirecionar o utilizador para o CAS, resultando na abertura de uma excepção interna por não reconhecer o utilizador neste processo. Este identificador previne ataques do tipo *Cross-Site Request Forgering*, garantindo que o utilizador iniciou o processo de autenticação no CAS.

A AMA foi contactada relativamente ao tratamento deste identificador no processo de autenticação por OAuth2, ao que apenas garantiu a sua utilização na fase final do projeto, impedindo uma investigação sobre a forma de mitigar as limitações que poderiam ocorrer durante a implementação.

Quando o identificador foi implementado, foi necessário haver uma nova adaptação do código-fonte das aplicações acima descritas, para interpretar a resposta do método *Implicit Grant* de forma a obter os atributos do utilizador e, posteriormente, validá-lo com o diretório de Ciências. No entanto, conforme foi descrito no Capítulo 2.3.5, a resposta à delegação de autenticação não pode ser interpretada pelo CAS porque os fragmentos no URI não são incluídos no pacote HTTP. De forma a contornar este obstáculo, foi criado um serviço de abstração e interpretação da delegação *OAuth2 Implicit Grant* que atua como um IdP *OAuth2 Authorization Code Grant*, protocolo suportado pelo CAS.

Serviço de abstração

Conforme descrito anteriormente, é pretendido que este serviço de abstração respeite os termos definidos no RFC3986[10] relativamente à autenticação por *OAuth2 Authorization Code Grant*. Ao mesmo tempo, deve conseguir obter o *token* devolvido numa autenticação bem sucedida através dos métodos disponibilizados pelo serviço Autenticação.gov.

Conforme demonstrado nas Figuras 4.1 e 4.2 , uma autenticação bem sucedida realiza-se na seguinte forma:

1. O CAS delega a autenticação ao serviço de abstração com os dados necessários para realizar autententicação na Autenticação.gov;
2. O serviço de abstração guarda o estado do utilizador referente ao CAS e redireciona-o para a Autenticação.gov
3. Após uma autenticação bem sucedida, é apresentada uma página de espera que permite a manipulação da âncora, que contém um *token* referente à Autenticação.gov, indicada no URL através de *Javascript*, transformando-a em *query* do protocolo HTTP;
4. O serviço de abstração utiliza o *token* para obter os dados do utilizador e guarda-os em memória. Neste passo, é gerado um código de autorização;
5. O utilizador é redirecionado para o CAS com o código de autorização referente ao serviço de abstração presente na *query*;
6. O CAS utiliza este código para obter um *token* de acesso aos dados armazenados no serviço de autenticação
7. O CAS utiliza o *token* de acesso para obter os dados do utilizador armazenados no passo 4;
8. O serviço de abstração apaga os dados do utilizador e respetivos identificadores;

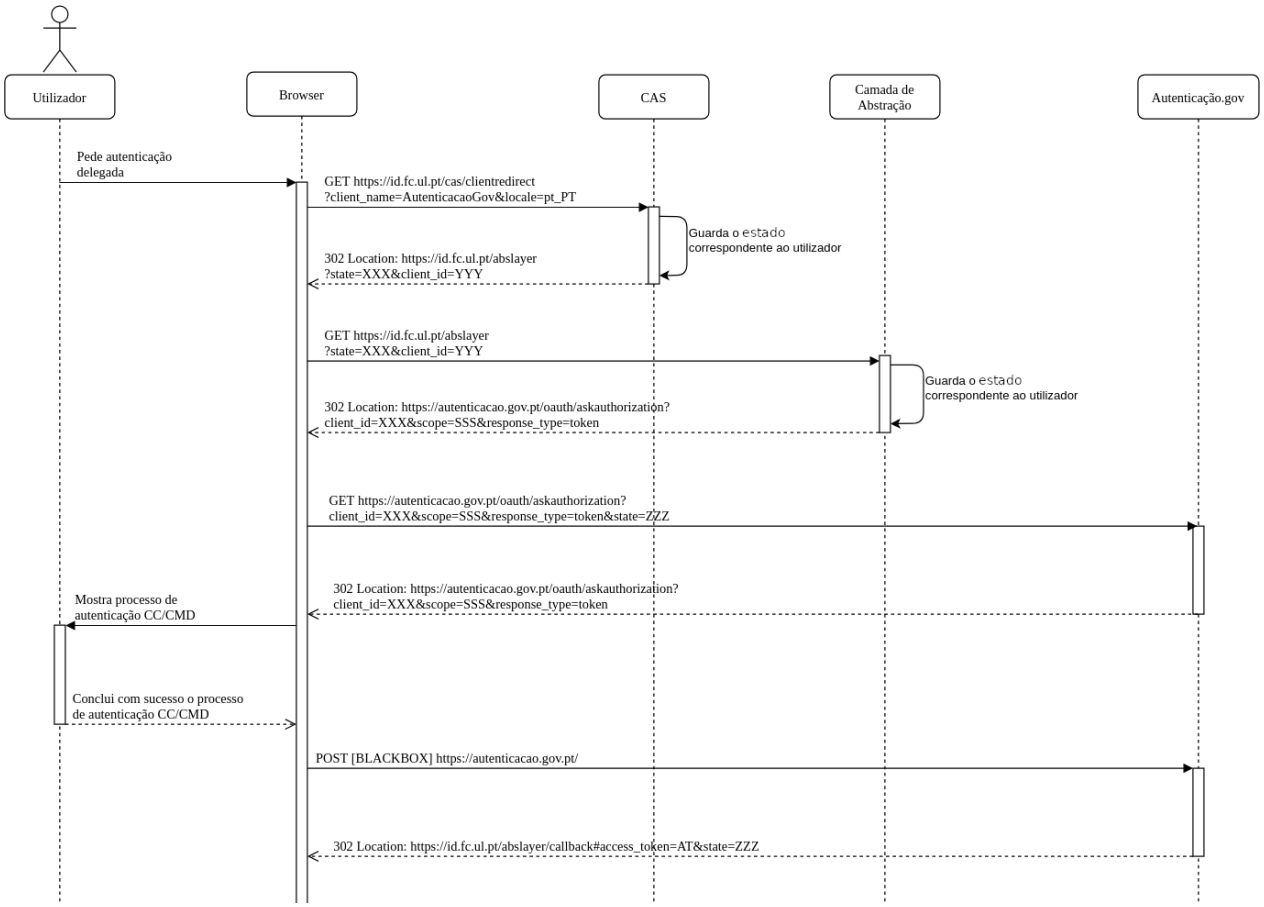

Figura 4.1: Diagrama de execução do serviço de abstração

9. Por fim, o CAS verifica que o utilizador existe na diretoria de utilizadores, autenticando-o com sucesso.

A solução apresentada garante a identidade do utilizador utilizando o mesmo identificador de estado que o CAS determinou para cada utilizador. Assim que um processo de autenticação é concluído, todos os dados que estão em memória, relativamente a esta sessão, serão apagados deste serviço, diminuindo a utilização de recursos no servidor.

4.3 Autenticação de dois fatores

A autenticação de dois fatores, 2FA (*Two Factor Authentication*), garante que são utilizadas duas categorias diferentes de confirmação da identidade do utilizador. Existem três tipos diferentes que se relacionam diretamente com o utlizador. Para simplificar esta percepção, é possível classifica-los na primeira pessoa: *algo que sei*, *algo que tenho* e *algo que sou*. Simplificando, podem ser utilizadas analogias diretas com o âmbito do tema desta dissertação:

- *Algo que sei* é a forma mais simples de autenticação e também, por isso mesmo, o método mais propício a falhas de segurança. Os utilizadores de Ciências utilizam este método, autenticando-se com o seu endereço institucional e a sua palavra-chave. Torna-se uma excelente superfície de ataque por depender diretamente da responsabilidade dos utlizadores fortalecer essa combinação. Muitas vezes,

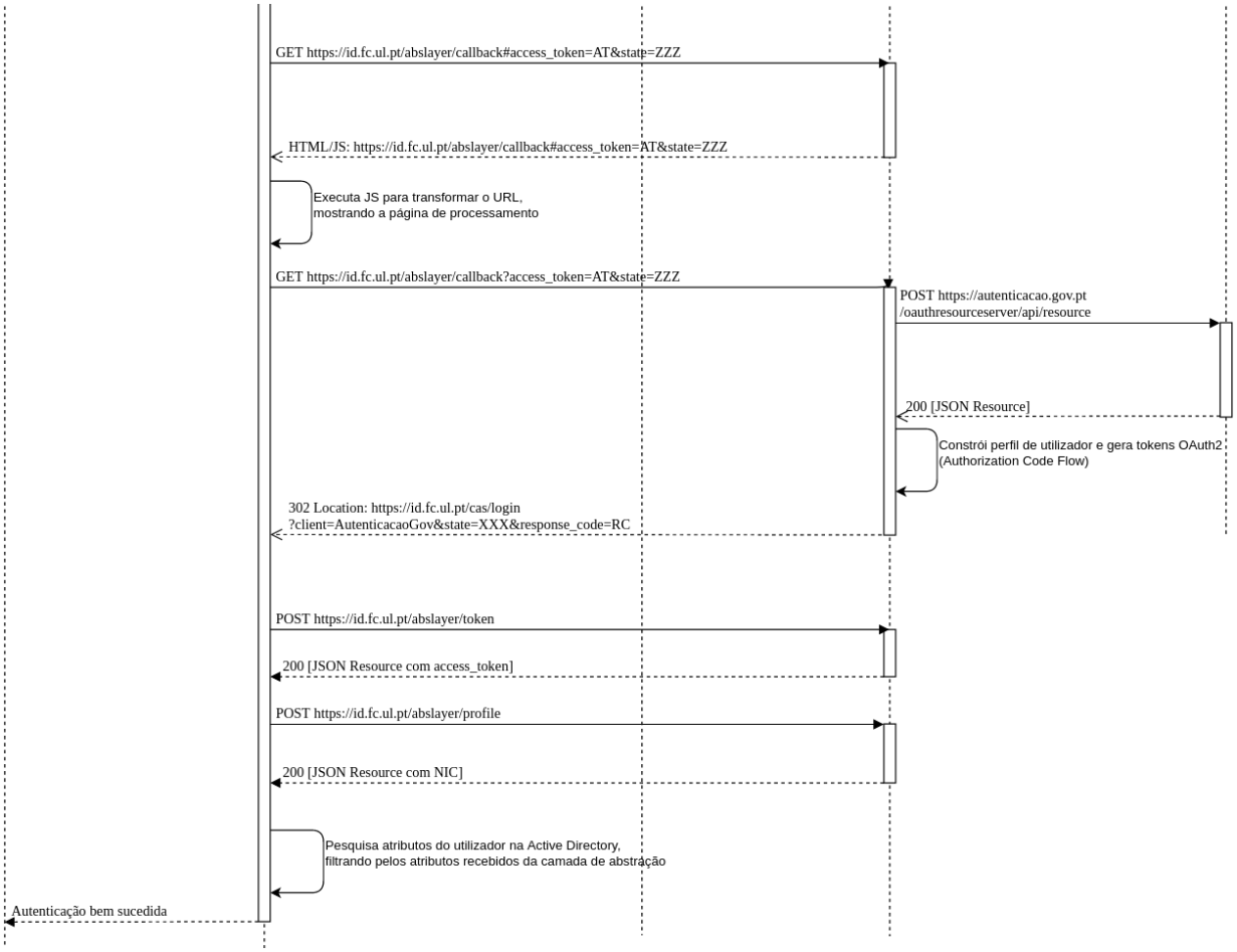

Figura 4.2: Diagrama de execução do serviço de abstração (continuação)

para simplificar a sua digitação ou lembrança, a combinação torna-se fraca, a nível de complexidade, e bastante fácil de atacar, por exemplo, com ataques de dicionário. Em suma, estes ataques consistem na tentativa de senhas obtidas através de palavras comuns.

- *Algo que tenho* representa um objeto que o utilizador tem na sua posse. Como exemplo simples, a autenticação nos serviços da empresa *Google* permitem este tipo de segundo fator. Um utilizador autentica-se com as credenciais comuns, email e palavra-chave, e é questionado no seu *smartphone*, já ele previamente autenticado, se é verdadeiramente a sua autenticação que está a ser executada.

- *Algo que sou* remete-nos para uma característica física do utilizador, nomeadamente impressões digitais ou a retina. Hoje em dia, este tipo de autenticação é muito utilizada em *smartphones* de forma a permitir uma forma mais rápida de aceder ao seu conteúdo, comparativamente com o tradicional PIN ou código de acesso.

Nesta instituição é pretendido implementar um segundo fator de autenticação aos seus utilizadores através da utlização de chaves que são geradas e descartáveis neste processo. Ao autenticar-se com as credenciais já existentes, o sistema de autenticação de Ciências irá enviar uma chave curta e simples para o número de telemóvel associado ao utilizador (*SMS Token*), obrigando-o a introduzi-la na continuação deste processo.

Este processo, garante que a autenticação é acreditada por dois dos fatores referidos na introdução deste capítulo: *algo que sei*, diga-se o utlizador e palavra-chave, e *algo que tenho*, referindo o telemóvel e a respetiva senha gerada.

Apesar de criar uma camada extra de segurança, a utilização de *SMS Tokens* cria outro tipo de vulnerabilidades no processo de autenticação. Muitas autenticações simultâneas fazem com que o SSO da instituição possa ficar sujeito a um ataque do tipo *Denial of Service*, DoS. Diretamente afetado pela sua capacidade de processamento, o equipamento que Ciências disponibiliza para enviar mensagens de texto é utilizado para diversos fins e a sua capacidade não é escalável. Ao haver uma sobrecarga deste sistema, impede que a autenticação seja realizada em todos os serviços, mesmo que o SSO tenha disponibilidade para o fazer.

Desta forma, foi decidido que este método iria apenas ser implementado após a completa integração da Chave Móvel Digital, visto que esta passa a representar uma redundância para a autenticação, dado este caso.

4.4 Notificações

O sistema de notificação tem como objetivo assegurar a tomada de conhecimento do utilizador de informações relevantes, procedendo-se ao registo nos sistemas de informação de Ciências do momento e condições em que a tomada de conhecimento ocorreu. Para tal, foi desenvolvida uma ferramenta que, após a autenticação, impede a utilização dos serviços até que o utilizador confirme explicitamente que tomou conhecimento da informação que lhe foi apresentada.

4.4.1 Gestão das Notificações

De forma a agilizar o processo de gestão das Notificações, o portal de Ciências foi modificado de forma a mostrar um painel de administração a utilizadores autorizados para este efeito, permitindo a criação e modificação destas notificações.

Pretendeu-se uma arquitetura simples, que se materializou na estrutura da base de dados conforme representada na Figura 4.3.

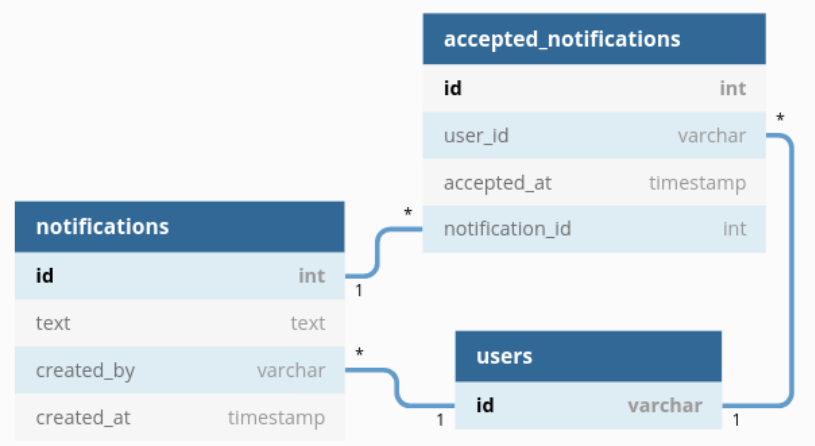

Figura 4.3: Estrutura da base de dados relativa às Notificações

Esta funcionalidade integra o CAS no fluxo de autenticação, impedindo o utilizador de avançar neste processo caso não tenha aceite alguma notificação, garantindo que os serviços de Ciências são utilizados apenas após a tomada de conhecimento das notificações pendentes. Desta forma, foi desenhado um fluxo de autenticação conforme representado no diagrama da Figura 4.4.

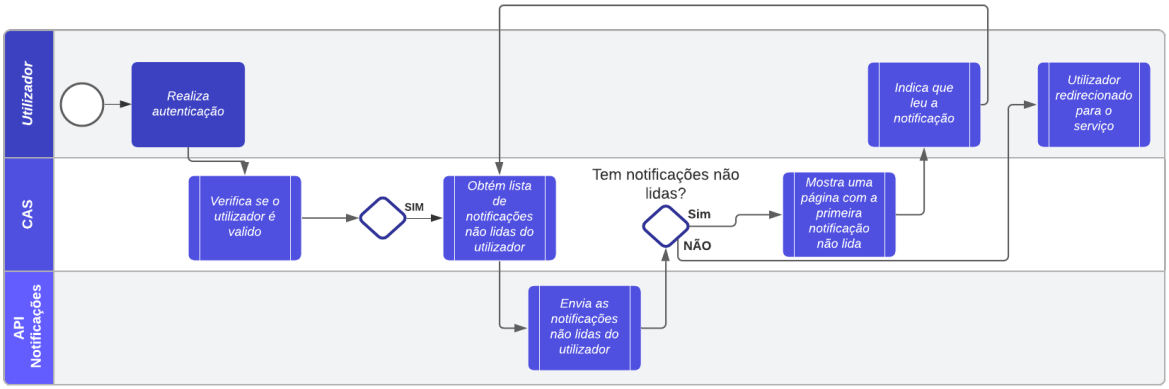

Figura 4.4: Fluxo de autenticação relativo às Notificações

Para existir um canal de comunicação universal entre o painel de gestão e o SSO, foram criados serviços web do tipo REST que pretendem centralizar a gestão dos registos de aceitação dos utilizadores. Foi utilizada a tecnologia *Vue.js*[17] em conjunto com a biblioteca *Axios*[18] no portal para agilizar o seu desenvolvimento.

Figura 4.5: Visão de Notificações Gerais criadas no software de gestão

Os *endpoints* são utilizados pelo CAS para obter as notificações não lidas e para submeter a indicação de leitura do utilizador são:

- `GET /entries?principalName=PRINCIPAL_NAME&showAccepted=BOOLEAN` - devolve todas as entradas, na base de dados, relativas às notificações aceites. Para limitar o resultado desta pesquisa, foram utilizados parâmetros na *query* do pedido HTTP, conforme podemos verificar no exemplo. O parâmetro *principalName* deve incluir o identificador do utilizador dentro do domínio de Ciências, por exemplo, o identificador de um aluno seria *fcXXXXX@alunos.fc.ul.pt*. O parâmetro *showAccepted* é um identificador booleano que, por omissão, toma o valor *true*. Por defeito, mostra as notificações já aceites por um utilizador, caso contrário, devolve todas as notificações que o utilizador identificado não tenha ainda aceite.
- `POST /entries` - disponibiliza um método que o CAS utiliza para popular a base de dados com a nova aceitação do utilizador.

Figura 4.6: Exemplo da listagem de Notificações aceites

Abstraindo esta lógica do CAS e devolvendo sempre o mesmo formato de resposta, qualquer alteração futura ao sistema de notificações não implica uma adaptação do SSO.

4.4.2 Integração com o CAS

A modificação do *webflow* utilizado pelo CAS foi necessária para entrar em conformidade com estes requisitos. Infelizmente, não existe qualquer tipo de documentação relativa a esta personalização do CAS. Ainda assim, como base de trabalho, foram utilizados diversos módulos, que integram este fluxo. Foi utilizada a lógica do módulo *Acceptable Usage Policy*[19](política de utilização aceitável) que guarda apenas um valor binário: se foi aceite ou não.

Este módulo utiliza, como fonte de informação, uma interface Java com métodos predefinidos que representa um repositório, que consiste em dois métodos com as funcionalidades de verificar o estado da política de utilização e a submissão de uma aceitação. No caso do utilizador não concordar, o fluxo de autenticação é interrompido.

Assim, foram criadas classes que utilizam esta interface. No método que verifica as notificações não lidas, foi utilizada uma biblioteca chamada *Retrofit*[20], que utiliza *webservices* REST transformando as respostas em objetos nativos, em Java. Todavia, foram encontradas dificuldades na interação do utilizador com a aceitação destas políticas. Era necessário, obviamente, apresentar o corpo da mensagem e, de certa forma, obter o seu identificador para registar a ação do utilizador. Assim, foi detetada a necessidade de implementar modificações à biblioteca *Thymeleaf* [21] como forma de mostrar o conteúdo pretendido ao utilizador. Inicialmente, os identificadores das mensagens mostradas e do utilizador eram obtidos através da utilização de *Javascript* na página mostrada ao utilizador, mas não era a situação ideal por estarem sujeitas a manipulação no cliente. Para resolver este problema, foi necessário modificar novamente o *webflow* para utilizar determinados *Beans* que reescrevem a forma como as variáveis são disponibilizadas à biblioteca *Thymeleaf* para gerar as páginas para o utilizador.

O endereço onde este API é executado é indicado no ficheiro de configuração do CAS, o que torna possível a execução desta nova funcionalidade nos diferentes ambientes utilizados na DSI, nomeadamente Desenvolvimento, Qualidade e Produção. Desta forma, facilita-se a troca de serviços de disponibilização de notificações, desde que respeita a estrutura de dados definida.

Figura 4.7: Exemplo de Notificação Legal durante o processo de autenticação

4.5 Aviso de propinas

Esta funcionalidade não está incluída no planeamento inicial, visto que foi requisitada durante o desenvolvimento deste projeto. Tem como finalidade validar se um aluno tem propinas em atraso, mostrando-lhes, ou não, uma mensagem da sua situação, interrompendo temporariamente o fluxo de autenticação.

4.5.1 Integração com o CAS

O desenvolvimento desta funcionalidade foi semelhante ao das Notificações. Foi utilizado o mesmo módulo inicial, mas, neste caso, não permitimos que o utilizador possa cancelar o processo de autenticação.

Pela equipa de desenvolvimento do DSI foi desenvolvido um *webservice*, na plataforma *FénixEdu*, que disponibiliza informações relativamente à situação das propinas dos utilizadores. Assim, foi apenas necessário utilizar a biblioteca *Retrofit* para realizar a comunicação com este serviço.

Seguindo a mesma metedologia utilizada para as Notificações, também fica disponível uma opção de configuração que torna apenas nececessária a indicação do *endpoint* a utilizar para obter as informações do utilizador referentes à respetiva situação financeira com Ciências.

4.6 Gestão de Serviços

Foi desenvolvido um interface de programação, API, baseado em REST para gerir os serviços autorizados a utilizar a autenticação do CAS. Este projeto pretende uniformizar as alterações definidas no portal de Ciências e altera a base de dados que o SSO utiliza para validar os domínios autorizados a utilizar a autenticação da instituição.

Apesar de já existirem operações diretas à base de dados para a versão anterior, esta interface abstrai o portal da base de dados. Esta solução foi necessária pois, na nova versão do CAS, a estrutura da base de dados prevê o armazenamento de objetos Java serializados, que não podem ser interpretados pelo painel de gestão anterior, que utiliza PHP.

Capítulo 5

Avaliação

A autenticação utilizando a chave móvel digital foi colocada em produção a 5 de fevereiro de 2021. A entrada em produção foi precedida de uma campanha de divulgação junto de toda a comunidade de Ciências e que teve três objetivos: i) alertar os utilizadores para a mudança de interface que a página de autenticação iria sofrer nessa data; ii) encorajar os utilizadores a adotar a chave móvel digital como método preferencial de autenticação e; iii) sensibilizar a comunidade para os perigos do roubo de identidade.

Os resultados apresentados na Fig. 5.1 revelam que o serviço teve uma adesão muito limitada por parte da comunidade, que continuou massivamente a optar pelo mecanismo clássico do par *login/password*. Das 133137 autenticações realizadas entre 5 (data de lançamento do serviço) e 18 de fevereiro de 2021, apenas 646 (0.49%) beneficiaram da segurança acrescida dada pelos 2 fatores de autenticação da chave móvel digital. O mesmo padrão é verificável numa janela temporal de 135 dias, que vai desde a data de lançamento do serviço até 19 de junho e durante a qual foram registadas 2562893 autenticações com sucesso mas apenas 19568 (0.76%) utilizaram a chave móvel digital.

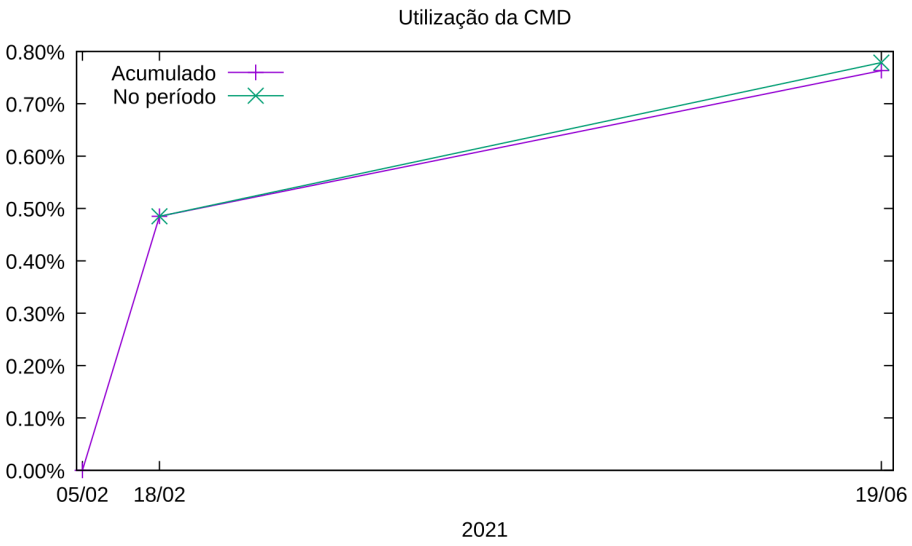

Figura 5.1: Utilização de 2 fatores de autenticação em Ciências

A avaliação por período, também representada na Fig. 5.1 não sugere uma tendência de crescimento. Contabilizando apenas as autenticações realizadas entre 18 de fevereiro e 19 de junho, a percentagem de utilização da chave móvel digital não vai além dos 0.78%.

Estes resultados podem ser atribuídos à complexidade adicional colocada aos utilizadores pela chave

móvel digital. A autenticação utilizando o modelo clássico beneficia da facilidade oferecida pela maioria dos *browsers* para armazenar o login e password dos utilizadores. Isto permite que os utilizadores realizem a sua autenticação primindo um único botão no portal. Em contraste, a autenticação utilizando a chave móvel digital implica necessariamente:

- navegar por pelo menos 5 páginas;
- a digitação do número de telemóvel, PIN e código de segurança (19 algarismos no total);
- aceder ao telemóvel para obter o código de segurança.

Reforçar junto dos utilizadores as vantagens da utilização de mecanismos de autenticação mais seguros apesar do esforço adicional sai fora do âmbito da dissertação e poderá requerer a imposição deste método aos utilizadores. Infelizmente, a chave móvel digital não se encontra ainda suficientemente generalizada para garantir que todos os utilizadores de Ciências dela dispõem. O problema é agravado por a comunidade de Ciências incluir utilizadores estrangeiros, a quem o acesso a este serviço é dificultado ou mesmo impossível.

Capítulo 6

Trabalho Relacionado

A implementação do projeto contribuiu para a adoção de novos métodos de trabalho e ferramentas por parte da equipa de desenvolvimento de Ciências e que são marginalmente abordadas neste capítulo.

6.1 Gitlab

O Gitlab é uma ferramenta de gestão de repositórios de código e foi implementado na DSI como tentativa de modernização das ferramentas de gestão de código da DSI. Algumas das funcionalidades desta plataforma revelaram-se bastante úteis durante a gestão de vários projetos. As mais relevantes incluem:

- *Issues*, onde se regista, gere e atribui diversos temas, problemas ou ideias a programadores.
- *Merge Requests* permite uma colaboração visual, entre programadores, num determinado contexto que propõe alterar o código-fonte do projecto. Aqui, é possível comentar linhas específicas do código e verificar as alterações propostas numa determinada ramificação do código-fonte (*branch*) do projecto.
- *Container Registry* permite o registo de *containers* baseados em Docker. Esta funcionalidade é utilizada para guardar as versões de qualidade e de produção do CAS de Ciências pronta a ser executada, faltando apenas o ficheiro de configuração, o que permite uma maior flexibilidade do projeto e a execução em diferentes ambientes sem alteração do código-fonte.
- *Continuous Integration/Continuous Delivery*, CI/CD, tal como o nome indica, propõe-se dar uma integração contínua ao projecto e entregá-lo ao destino final. Esta ferramenta permite criar um *pipeline*, determinando o ciclo de vida do código ao ser sumbetido no repositório. Desta forma, é possível automatizar o teste e validação do *software*, compilação e, finalmente, a atualização do servidor.

Sem dúvida que, destas, a funcionalidade mais importante foi o CI/CD. Possibilitou a automatização do processo de compilação e lançamento do produto final num servidor. Foi adotado um fluxo de trabalho conforme representado na Figura 6.1. Este fluxo é executado quando novo código é submetido no ramo *master* do repositório deste projeto e o projeto é compilado no primeiro passo. Em seguida, construímos um *container* que fica registado e alojado nesta instância do GitLab e, finalmente, são executados comandos que atualizam a versão da aplicação compilada no servidor remoto.

6.2 Kubernetes

Como tentativa de melhorar a disponibilidade do SSO, foi montado um *cluster* baseado na tecnologia *Kubernetes*, permitindo uma agilização da distribuição da carga por diferentes nós, em oposição a criar máquinas

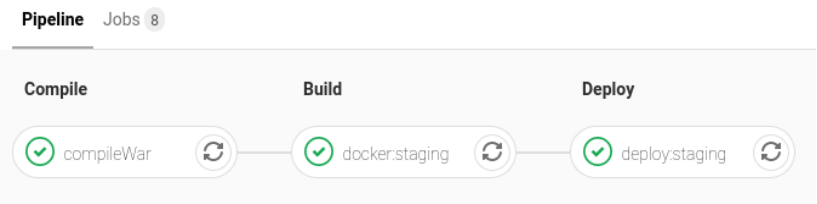

Figura 6.1: *Pipeline* no GitLab CI/CD

virtuais específicas para este efeito. Contudo, esta implementação foi abandonada por, segundo a equipa de Sistemas da DSI, estar planeada uma atualização do software *VMWare* que vem suportar a orquestração de *containers*.

Este software permite uma simplificação da gestão dos diferentes características de cada serviço que este SSO permite que se autentiquem.

Capítulo 7

Conclusão

Ciências já contava com um sistema de autenticação bastante robusto e seguro, no entanto, existem sempre melhorias que podem ser implementadas.

Durante a atualização do CAS para a versão 6, a documentação disponibilizada demonstrou-se pouco pormenorizada, dificultando, assim, este processo. Apesar disso, consegui, com sucesso, integrar serviços pretendidos, o que se traduziu numa modularização do SSO, facilitanto o acompanhamento com as novas versões, melhoradas e presumivelmente mais seguras, deste sistema.

Um dos maiores benefícios pessoais, que sem dúvida irei utilizar ao longo da minha carreira profissional, foi a agilização da interpretação de código-fonte de terceiros com pouco ou nenhum apoio, sobre uma tecnologia que conhecia muito superficialmente. Isto fez com que a minha capacidade de abstração dos problemas aumentasse, permitindo que não visse um excerto de código como uma solução para um problema, mas como uma parte integrante dessa solução. A utilização de *Spring Boot* foi, sem dúvida, um grande desafio. Visa simplificar o desenvolvimento de aplicações, mas o fluxo dos dados não procede da maneira convencional lecionada na Licenciatura.

A nível de segurança, foi-me possível aprofundar conhecimentos especialmente em protocolos de autenticação e as diferentes formas de a trespassar, formas essas que não podem ser permitidas pelo software desenvolvido.

Como trabalho futuro, seria benéfico que todos os *softwares*, em particular os de gestão de serviços do CAS e de notificações, se tornassem projetos públicos e de código aberto, proporcionando aos inúmeros utilizadores deste sistema de SSO um leque maior de funcionalidade e aplicações de apoio à integração.

Concluíndo, este projeto deixa, à Faculdade de Ciências, um sistema de *Single Sign-on* integrado com dois serviços, o Aviso de Propinas e as Notificações em ambiente de produção, com a documentação por mim fornecida.

A integração com o IdP Autenticação.gov dependia de funcionalidades que o CAS, à data, não possuia. Ainda assim, foi possível concretizar esta solução.

Referências

[1] *Direção dos Serviços Informáticos*. Out. de 2019. URL: https://ciencias.ulisboa.pt/pt/unidade.

[2] *Spring.io*. Nov. de 2019. URL: https://spring.io/projects/spring-boot.

[3] *Spring Dependency Injection*. URL: https://docs.spring.io/spring/docs/current/spring-framework-reference/core.html#beans-factory-collaborators.

[4] *TCP/IP Protocol Architecture Model*. URL: https://docs.oracle.com/cd/E19683-01/806-4075/ipov-10/index.html.

[5] *RFC3986 - Uniform Resource Identifier (URI): Generic Syntax*. URL: https://tools.ietf.org/html/rfc3986#section-1.1.3.

[6] *SAML*. Nov. de 2019. URL: http://saml.xml.org/.

[7] *Wikipédia: SAML*. Nov. de 2019. URL: https://en.wikipedia.org/wiki/Security_Assertion_Markup_Language.

[8] *SAML Specification*. URL: http://docs.oasis-open.org/security/saml/Post2.0/sstc-saml-tech-overview-2.0.html.

[9] *SAML2 Metadta*. URL: http://docs.oasis-open.org/security/saml/v2.0/saml-metadata-2.0-os.pdf.

[10] *OAuth2.0 RFC 6749*. URL: https://tools.ietf.org/html/rfc6749.

[11] *Which OAuth 2.0 Flow Should I Use?* URL: https://auth0.com/docs/authorization/which-oauth-2-0-flow-should-i-use#is-the-client-a-single-page-app-.

[12] *Central Authentication Service*. URL: https://apereo.github.io/cas/6.2.x/index.html.

[13] *simpleSAMLphp*. Nov. de 2019. URL: https://simplesamlphp.org/.

[14] *Drupal - Open-source CMS*. URL: https://www.drupal.org/.

[15] *pac4j - Issue 1583*. URL: https://github.com/pac4j/pac4j/issues/1583.

[16] *CAS Pull Request 4871*. URL: https://github.com/apereo/cas/pull/4871.

[17] *Vue.js*. URL: https://vuejs.org/.

[18] *Axios*. URL: https://github.com/axios/axios.

[19] *CAS Acceptable Usage Policy*. URL: https://apereo.github.io/cas/6.2.x/webflow/Webflow-Customization-AUP.html.

[20] *Retrofit*. URL: https://square.github.io/retrofit/.

[21] *Thymeleaf*. URL: https://www.thymeleaf.org/.

Glossário

2FA *2 Factor Authentication*. 20

AMA Agência para a Modernização Administrativa. 2, 20–22

API *Application Programming Interface*. 27, 28

CAS *Central Authentication System*. vi, viii, 2, 13–15, 18, 20–23, 25–28, 31, 33

CC Cartão de Cidadão. 2

CMD Chave Móvel Digital. 2, 20

CMS *Content Management System* - Sistema de gestão de conteúdo. 18

CN Common Name. 16

CSRF Cross-Site Request Forgery. 15

DC Domain Component. 16

DN Distinguished Name. 16

DSI Direção dos Serviços Informáticos. 2, 27, 28, 31, 32

FCT RTEPLACE. 20

HTTP *Hypertext Tranfer Protocol*. 4–6, 8, 11, 15, 21, 22, 26

IdP *Identity Provider*. 7–12, 14, 17, 20, 22, 33

IoC Inversion of Control. 4

NIC Número de Identificação de Civil. 2

OU Organizational Unit. 16

PIN *Personal Identification Number*. 2

REST *Representational state transfer*. 26–28

RFC *Request for Comments*. viii, 11–14, 22

SAML *Security Assetion Markup Language*. 7–11, 13, 17, 21

SP *Service Provider*. 7–9, 11, 12, 14, 17

SSO RTEPLACE. viii, 1, 2, 7, 8, 11, 15, 17, 19, 20, 22, 25–28, 31–33

ST Service Ticket. 15

TGT Ticket Granting Ticket. 15

TLS Transport-Layer Security. 6

URI *Uniform Resourse Identifier*. 5, 11–13, 22

URL *Uniform Resourse Location*. 22

[title=Lista de Acrónimos]