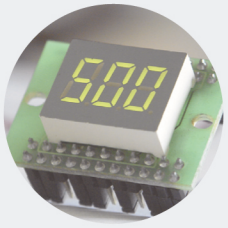
INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES






SINGLE SIGN-ON E USER EXPERIENCE DO IPORTALDOC

MIGUEL ÂNGELO COSTA SANTOS
julho de 2017

POLITÉCNICO
DO PORTO

SINGLE SIGN-ON E USER EXPERIENCE DO IPORTALDOC

Miguel Ângelo da Costa Santos

Departamento de Engenharia Eletrotécnica

Instituto Superior de Engenharia do Porto

2017

Este relatório satisfaz, parcialmente, os requisitos que constam da Ficha de Disciplina de Tese/Dissertação, do 2º ano, do Mestrado em Engenharia Eletrotécnica e de Computadores, do Ramo de Automação e Sistemas (MEEC-AS)

Candidato: Miguel Ângelo da Costa Santos, Nº 1120587, 1120587@isep.ipp.pt

Orientação científica: Paula Maria Marques Moura Gomes Viana, pmv@isep.ipp.pt

Empresa: IPBRICK, SA

Supervisão: Telma Salgueiro, tsalgueiro@ipbrick.com

Departamento de Engenharia Eletrotécnica

Instituto Superior de Engenharia do Porto

20 de julho de 2017

Agradecimentos

Este documento sinaliza o final do meu percurso académico, por isso apresenta grande importância pessoal. Desta forma, gostaria de agradecer a quem me ajudou a concretizar este feito:

Em primeiro lugar à minha família e amigos pelo apoio, que foi crucial para o meu sucesso, durante todo o meu progresso académico.

Ao Instituto Superior de Engenharia do Porto e aos seus docentes pela minha formação e preparação para a vida profissional.

A todos os membros da IPBRICK, especialmente ao engenheiro Marco Pinto e à engenheira Telma Salgueiro pela oportunidade e pela orientação durante este trabalho.

À engenheira Paula Viana, do ISEP, pela orientação na elaboração deste documento.

Resumo

Este trabalho é dedicado ao aperfeiçoamento da Experiência de Utilização (*User Experience* – UX) do software iPortalDoc da empresa IPBRICK SA consistindo na aplicação de um novo método de *Single Sign-on* (SSO), na substituição das mensagens de alerta JavaScript por funções da biblioteca NOTY, e numa análise das soluções para assinaturas digitais através de *smart cards* e *tokens* USB neste software.

De modo a encontrar a nova solução de SSO, foram avaliados vários métodos diferentes com o objetivo de resolver os problemas presentes na solução antiga, tal como a incompatibilidade com os diferentes Browsers. A solução escolhida foi o Kerberos e a acompanhar esta implementação foram introduzidas duas páginas, uma para gestão das configurações e dos utilizadores de SSO e uma para a autenticação através do Kerberos. Foi também desenvolvido um procedimento para automatização da configuração do SSO quando se introduz um servidor AD ou Samba 4 no iPortalDoc.

A implementação passou pela utilização da biblioteca NOTY para substituir os alertas de JavaScript com o objetivo de resolver situações onde estes são desativados pelos utilizadores e causam disfuncionalidade no software.

A análise dos métodos para realizar assinaturas digitais com *smart cards* e *tokens* USB, que tinha como objetivo principal ultrapassar a indesejável necessidade de descarregar um programa para a máquina local, levou à conclusão que ainda não existem implementações que cumpram os requisitos desejados.

Palavras-Chave

UX, SSO, Kerberos, alerta JavaScript, *smart card*, *token* USB, NOTY.

Abstract

This work is dedicated to the improvement of the User Experience (UX) of iPortalDoc, a software by IPBRICK SA, consisting on the application of a new Single Sign-on (SSO) method, the replacement of JavaScript alert messages with functions of the NOTY library, and an analysis of solutions for digital signatures through smart cards and USB tokens in this software.

To find the new SSO solution, several different methods were evaluated with the objective of solving the problems present in the old solution, such as the incompatibility with the different Browsers. The solution chosen was Kerberos and to follow this implementation two pages were introduced, one for managing the SSO settings and users and one for authenticating through Kerberos. A procedure for automating the SSO configuration was also developed when introducing an AD or Samba 4 server in iPortalDoc.

The implementation went through the usage of the NOTY library to replace the JavaScript alerts with the intent of solving situations where they are disabled by users and cause dysfunctionalities in the software.

The analysis of methods to perform digital signatures with smart cards or USB tokens, whose main objective was to overcome the undesirable need to download a program to the local machine, led to the conclusion that there are still no implementations that fulfil the desired requirements.

Keywords

UX, SSO, Kerberos, JavaScript alerts, smart card, USB token, NOTY.

Índice

AGRADECIMENTOS ..... **I**

RESUMO ..... **III**

ABSTRACT ..... **V**

ÍNDICE ..... **VII**

ÍNDICE DE FIGURAS ..... **IX**

ACRÓNIMOS ..... **XI**

1. INTRODUÇÃO **1**

1.1. CONTEXTUALIZAÇÃO E OBJETIVOS 1
1.2. ORGANIZAÇÃO DO RELATÓRIO 3

2. *USER EXPERIENCE* (UX) **5**

2.1. AS VÁRIAS MODALIDADES DE USER EXPERIENCE 6
2.2. JAVA E UX 8

3. *SINGLE SIGN-ON* (SSO) **11**

3.1. SOLUÇÕES DE AUTENTICAÇÃO PARA SSO 11
3.2. SERVIÇOS DE DIRETÓRIO COM SSO 18
3.3. COMPARAÇÃO DOS DIFERENTES SERVIÇOS DE SSO 20

4. ARQUITETURA DO SSO NO IPORTALDOC **23**

4.1. CENÁRIOS 23
4.2. REQUISITOS ADICIONAIS IMPOSTOS PELO IPORTALDOC 25
4.3. DECISÃO DA IMPLEMENTAÇÃO A USAR 28

5. DESENVOLVIMENTO **29**

5.1. KERBEROS COM AD 30
5.2. KERBEROS COM SAMBA 4 34
5.3. INTEGRAÇÃO COM O IPORTALDOC 36
5.4. BIBLIOTECA NOTY PARA FERRAMENTA DE NOTIFICAÇÃO 42

6. SMART CARD E TOKEN USB **45**

6.1. CARACTERÍSTICAS DESEJADAS PARA A IMPLEMENTAÇÃO DA ASSINATURA DIGITAL 45
6.2. ANÁLISE DE SOLUÇÕES 46
6.3. SÍNTESE 47

7. CONCLUSÕES **49**

7.1. PERSPETIVAS FUTURAS 50

REFERÊNCIAS DOCUMENTAIS...**53**

Índice de Figuras

Figura 1 – Protocolo Kerberos 12
Figura 2 – Autenticação NTLM 13
Figura 3 – SSO em SAML 2.0 14
Figura 4 – Fluxo de mensagens de autorização 16
Figura 5 – Fluxo de mensagens no OpenID Connect 17
Figura 6 – Cenários da implementação 24
Figura 7 – Funcionamento ideal da implementação do SSO no iPortalDoc 25
Figura 8 – Teste de funcionamento do Kerberos 34
Figura 9 – Página de autenticação Kerberos 37
Figura 10 – Alteração do método de SSO 38
Figura 11 – *Layout* da interface de gestão de utilizadores do SSO 39
Figura 12 – Campos necessários para o Kerberos 41
Figura 13 – Alerta NOTY 43

Acrónimos

AD – Active Directory

ADDS – Active Directory Domain Services

ADFS – Active Directory Federated Services

API – Application Programming Interface

AS – Authentication Server

DC – Domain Controller

DNS – Domain Name System

FQDN – Fully Qualified Domain Name

HTTP – Hyper Text Transfer Protocol

ID – Identifier

IDP – Identity Provider

IP – Internet Protocol

JSON – JavaScript Object Notation

JWT – JavaScript Object Notation Web Token

KDC – Key Distribution Centre

LDAP – Lightweight Directory Access Protocol

NTLM – New Technology Lan Manager

ONA – Open Network Architecture

OP – OpenID Provider

PHP – Hypertext Pre-processor

RP – Relying Party

SAML – Security Assertion Mark-up Language

SP – Service Provider

SSL – Secure Socket Layer

SSO – Single Sign-On

TGS – Ticket-granting Server

TGT – Ticket-granting Ticket

UI – User Interface

URL – Uniform Resource Locator

UX – User Experience

1. INTRODUÇÃO

Este documento descreve o trabalho realizado no âmbito da unidade curricular de Tese e Dissertação (TEDI), do 2º ano do Mestrado em Engenharia Eletrotécnica e de Computadores – Ramo de Automação e Sistemas (MEEC-AS), do Departamento de Engenharia Eletrotécnica (DEE), do Instituto Superior de Engenharia do Porto. O trabalho tem como objetivo principal o aperfeiçoamento da Experiência de Utilização (*User Experience* – UX) do software iPortalDoc da empresa IPBRICK, SA. O aspeto principal a abordar está relacionado com a implementação de *Single Sign-On* (SSO), seguido da introdução de uma biblioteca (NOTY) para substituição das mensagens de alerta e confirmação JavaScript e a avaliação de uma nova solução para assinaturas digitais usando *smart cards* e *tokens* USB.

1.1. CONTEXTUALIZAÇÃO E OBJETIVOS

O software iPortalDoc é um sistema de gestão de documentação e *workflows* que permite aos seus utilizadores a gestão dos fluxos dos documentos, como também, simplesmente, proceder ao seu arquivo para posterior gestão. Esta solução de gestão documental encara a informação digital como um substituto do papel, sendo esperado que introduza uma série de benefícios para a atividade de uma organização.

A garantia de penetração de um produto no mercado está claramente relacionada não só com as funcionalidades oferecidas, mas também com a facilidade de utilização. A área de UX, abordada neste trabalho, foca-se na avaliação da relação entre o utilizador e o software, tendo como objetivo a sua melhoria, satisfazendo, desta forma, o cliente.

A UX está integrada em todas as componentes deste projeto, sendo que todas as resoluções dos problemas a enfrentar terão que ter em consideração o impacto, positivo ou negativo, desta. Para que este impacto seja o mais positivo possível, será necessário o estudo dos princípios orientadores, regras, ou considerações atualmente utilizadas em aplicações com bons níveis de UX. Este estudo será explorado no próximo capítulo.

Sendo uma área muita vasta, o tema principal a abordar neste trabalho é o *Single Sign-On*, que é uma característica que permite que um dado utilizador efetue apenas um *login*, com um *Identifier* (ID) de utilizador e uma password ou outra informação de autenticação, tendo acesso integrado a um conjunto de aplicações e funcionalidades.

Em domínios empresariais é habitual esta autenticação ser efetuada através do *login* de um colaborador da empresa no seu posto (computador utilizado por este), estando esta máquina presente no domínio da empresa. Após esta autenticação, este deve ter acesso a todos os serviços sem necessitar de preencher novamente as suas credenciais.

A implementação correta de SSO está enquadrada como uma medida positiva de UX, pois esta facilita o acesso ao software, tornando a autenticação do utilizador um procedimento único e não repetitivo.

O iPortalDoc já possui um método de SSO baseado em Java *Applets*. No entanto, este método apresenta diversas falhas das quais se destacam as seguintes:

- Necessidade do utilizador do iPortalDoc ter o software Java instalado na sua máquina e as possíveis incompatibilidades entre as versões instaladas na máquina e utilizada no código.

- A necessidade de download da *Applet* para a máquina do utilizador pois consome recursos, quer a nível de espaço no disco, quer a nível de tempo, enquanto este espera que o procedimento se efetue.

A proposta de implementação deverá resolver os problemas atuais, sendo também compatível com clientes a utilizarem diversos sistemas operativos ou Browsers.

No que toca ao NOTY, este será introduzido com o objetivo de substituir alertas de JavaScript que apresentam problemas de funcionamento – nomeadamente a sua possível desativação por um utilizador causando disfuncionalidade no software.

Por fim, será realizada uma avaliação sobre possíveis soluções para assinatura digital com *smart cards* e *tokens* USB de modo a contextualizar o estado atual das soluções e verificar se estas são superiores à solução Java que se encontra implementada no software.

1.2. ORGANIZAÇÃO DO RELATÓRIO

No capítulo 1 faz-se uma contextualização do trabalho a desenvolver, são identificados os objetivos a atingir, e é descrita a organização do relatório.

No capítulo seguinte, 2, são discutidos conceitos relacionados com UX através de um estudo sobre esta área.

No terceiro capítulo são avaliados diferentes métodos existentes para implementação de SSO, identificando as vantagens e desvantagens de cada um, tendo em conta as necessidades anteriormente descritas.

No quarto capítulo é apresentada a arquitetura de SSO desejada, considerando os cenários presentes nos clientes do iPortalDoc e como esta implementação será integrada no software. Por fim, determina-se a tecnologia SSO mais apropriada para esta arquitetura.

No capítulo 5 descreve-se a implementação do método SSO e a integração deste e da ferramenta NOTY no iPortalDoc.

No sexto capítulo é feito um levantamento de soluções, para substituição do método Java, que realizem assinaturas digitais através de *smart cards* e *tokens* USB.

Em último lugar, capítulo 7, são apresentadas as principais conclusões e indicados caminhos para futuros desenvolvimentos.

2. *USER EXPERIENCE* (UX)

Uma boa UX [1] é uma consequência da apresentação, funcionalidade, performance, comportamento interativo e capacidades auxiliares de um sistema interativo, quer de hardware quer de software. É também a consequência das experiências, atitudes, capacidades e hábitos anteriores e personalidade do utilizador. A UX também inclui aspetos emocionais, tais como questões de satisfação com o emprego e a eliminação da monotonia quando interpretados na perspetiva dos objetivos pessoais do utilizador.

Desenvolver um software tendo em conta a sua UX envolve considerar, quando apropriado, impactos organizacionais, documentação para o utilizador, ajuda online, suporte e manutenção (incluindo *helpdesks* e pontos de contacto com o consumidor), formação, entre outros. A UX de outros sistemas e sistemas anteriores e outras questões, como o *branding* e a publicidade também deve ser considerada. A necessidade de considerar estes diferentes fatores e as suas interdependências tem implicações para o plano do projeto. As qualidades, limitações, preferências e expectativas também devem ser consideradas quando especificando as atividades que serão efetuadas pelos utilizadores e as que serão efetuadas pela tecnologia.

2.1. AS VÁRIAS MODALIDADES DE USER EXPERIENCE

A UX no contexto de software [2] é a reflexão objetiva da interação entre clientes e os serviços de informação. Uma UX rica destaca a emoção do utilizador. Tendo em conta o nível de eficácia, os sistemas de informação podem ajudar os utilizadores a atingir diferentes propósitos, assim, a experiência pode ser dividida em três níveis: o primeiro relaciona o serviço com a satisfação da experiência funcional do utilizador, o segundo foca-se na questão deste primeiro nível ter ou não sido atingido com eficiência, por fim, o terceiro aborda a estética do serviço, visto que esta pode tornar o cumprimento das tarefas agradável não só a um nível físico, mas também mental.

É importante distinguir a UX total da *User Interface* (UI), apesar da UI ser um fator extremamente importante na conceção de uma aplicação [3].

Também se deve distinguir UX de usabilidade: de acordo com a definição de usabilidade, esta é uma qualidade de atributo do UI, que analisa se o sistema é fácil de aprender, eficiente ao usar, agradável, entre outros. Todos estes fatores são importantes e a UX integra e expande este e outros conceitos.

A usabilidade [4] pode também ser relacionada com a utilidade. Estes dois conceitos são igualmente importantes e juntos determinam se uma aplicação de software é útil. Não importa muito que algo seja fácil de usar se não possuir as funções desejadas. Também não é relevante se o sistema pode, hipoteticamente, fazer o que é necessário, mas o utilizador não o consegue realizar porque a UI é muito complexa. Sendo assim, uma aplicação de software útil é a que fornece as funcionalidades desejadas e, ao mesmo tempo, é fácil e agradável ao uso.

Na *Web*, a usabilidade torna-se uma condição necessária para a sobrevivência. Se um site é difícil de usar, se a página principal não consegue demonstrar claramente o que uma empresa oferece e o que os utilizadores podem fazer, se os utilizadores se perdem no site, ou se a informação é difícil de ler ou não responde às questões do utilizador, estes deixam de usar o site. Um utilizador nunca lerá um manual de utilização de um site ou gastará demasiado tempo a tentar descobrir como usar uma UI. Há provavelmente muitos outros sites disponíveis, o que leva muitos utilizadores a trocarem de um site para outro quando encontram dificuldades.

Em intranets, a usabilidade é uma questão de produtividade do trabalhador. O tempo que se perde quando um colaborador não consegue, ou tem dificuldade a aceder a um recurso do software resulta em dinheiro perdido pois este não está a ser eficaz no trabalho.

2.1.1. PRINCÍPIOS ORIENTADORES DE USABILIDADE

A literatura identifica dez princípios orientadores [5], normalmente usados, para o bom desenho de uma UI:

- Visibilidade do estado do sistema – O sistema deve, sempre, manter os utilizadores informados sobre o que está a acontecer através de *feedback* em tempo razoável.
- Correspondência com o mundo real – O sistema deve usar a linguagem do utilizador, com palavras, frases e conceitos familiares ao utilizador, em vez de termos orientados ao sistema. Deve seguir convenções do mundo real, apresentando informação numa ordem natural e lógica.
- Controlo e liberdade do utilizador – Os utilizadores, por vezes, escolhem funções do sistema por acidente e necessitarão de uma "saída de emergência" para sair dessa situação.
- Consistência e normas – Palavras, situações ou ações têm que significar sempre o mesmo.
- Prevenção de erros – Ainda melhor que boas mensagens de erro é um design cuidado que previna que um problema ocorra de início. Eliminar condições que poderão induzir erros ou verificar estas e apresentar ao utilizador uma opção de confirmação antes destes executarem a ação.
- Reconhecimento em vez de lembrança – Minimizar a necessidade de memorização do utilizador ao apresentar objetos, ações e opções visíveis. O utilizador não tem que se recordar de informação anterior ao transitar de uma parte do diálogo para outra. Instruções para o uso do sistema devem estar visíveis ou serem facilmente obtidas quando apropriado.
- Flexibilidade e eficiência de uso – Aceleradores, não identificados por um utilizador novo, podem aumentar a velocidade da interação de utilizadores experientes, de forma a que o sistema suporte utilizadores com qualquer nível de conhecimento do software. Permitir aos utilizadores adaptar ações frequentes.

- Estética e design minimalista – Diálogos não devem conter informação que é irrelevante ou raramente necessária. Qualquer informação extra num diálogo compete com a informação relevante reduzindo a sua visibilidade.
- Ajudar o utilizador a reconhecer, analisar e recuperar a partir de erros – Mensagens de erro devem ser exprimidas em linguagem simples, sem códigos, indicando precisamente o problema e, construtivamente, sugerir uma solução.
- Ajuda e documentação – Embora o ideal seja que o sistema possa ser usado sem documentação, pode ser necessário fornecer ajuda e documentação. Qualquer informação deve ser fácil de procurar, focada na tarefa do utilizador, listando passos concretos para serem efetuados, e não deve ser demasiado extensa.

Estas recomendações ajudam o desenvolvimento de uma aplicação, pois dão indicações sobre os cuidados necessários ao design de um software que é útil a todo o tipo de utilizadores.

2.2. JAVA E UX

Atualmente, dois dos Browsers mais utilizados, o Firefox [6] e o Google Chrome [7], não suportam Java *Applets*. Desta forma, qualquer implementação que possa usar esta tecnologia irá criar problemas de compatibilidade nos utilizadores. Isto é um fator indesejado durante o uso de um software, pois um utilizador irá ser impedido de usar este código, resultando numa má UX. Assim, qualquer implementação que use *Applets* terá que ser substituída por outra solução, de forma a ser compatível com qualquer Browser.

Para além de *Applets*, algumas funções de JavaScript começaram a poder ser desativadas em alguns Browers. Um exemplo disto encontra-se descrito na subseção seguinte.

2.2.1. AVISOS, *FEEDBACK* E CONFIRMAÇÕES

De acordo com o princípio orientador referente à visibilidade do estado do sistema, o *feedback* relativo ao que se encontra em segundo plano e não é visível ao utilizador deve ser de alguma forma indicado na página atual. Os avisos de confirmação antes de uma alteração significativa também devem estar devidamente apresentados.

Em várias soluções de software este *feedback* é proveniente de alertas baseados em JavaScript, ou seja, funções da biblioteca que podem ser utilizadas para avisos de erro e

para pedidos de confirmação. Estes avisos e pedidos são funções bloqueantes que param o funcionamento da página enquanto estes não forem pressionados.

Estes alertas começaram a revelar problemas nas UI que as utilizavam quando Browsers, como o Google Chrome, começaram a permitir que estas fossem desativadas. Quando um alerta aparece pela segunda vez, este vem com uma nova opção a sugerir para não aparecerem mais avisos deste tipo. A desativação destes alertas numa UI pode levar a um colapso total do funcionamento de uma aplicação, pois na situação onde o utilizador efetuaria alguma mudança, ou realizaria um funcionamento com erro, estes alertas iriam aparecer. Encontrando-se desativados, a página irá bloquear, sem demonstrar nenhum aviso com o qual se possa interagir de forma a continuar o funcionamento normal do software. Esta ocultação deixa um utilizador completamente desorientado em relação ao uso do software, pois este bloqueia, não avisa porquê, e não apresenta soluções para a sua resolução. Assim, qualquer implementação de uma UI, deve evitar o uso destas funções do JavaScript encontrando soluções mais atrativas e amigáveis para os utilizadores do software.

2.2.2. SSO

O SSO é considerado uma medida de UX uma vez que pode contribuir para aumentar a satisfação dos utilizadores. Esta satisfação resulta da simplicidade no acesso a uma página pelo facto de, ao ter introduzido as suas credenciais quando iniciou sessão no seu computador, não necessitar de preencher de novo o formulário relativo à sua autenticação. Assim, todos os processos de autenticação que poderiam estar presentes numa aplicação deixam de ser um fator repetitivo no dia a dia de um utilizador.

De acordo com o que foi discutido antes, um método de SSO baseado em Java *Applets* é uma solução pouco atrativa. Assim, é necessário encontrar soluções mais fiáveis e compatíveis, de modo a fornecer a melhor UX possível. No próximo capítulo faz-se a análise de algumas soluções.

3. *SINGLE SIGN-ON* (SSO)

Uma implementação de SSO garante acesso repetido a um ou vários serviços através de uma única autenticação inicial. Neste capítulo aprofunda-se este tópico.

3.1. SOLUÇÕES DE AUTENTICAÇÃO PARA SSO

Irão ser analisadas várias alternativas para a implementação, descrevendo as suas funcionalidades, de forma a avaliar as escolhas mais aptas para o SSO através de autenticação local, ou seja, autenticação no posto de trabalho dentro da empresa. Assim, garante-se o acesso ao iPortalDoc sem necessitar de introduzir novamente as credenciais.

3.1.1. KERBEROS

Kerberos [8] é o nome dado ao serviço de autenticação, ao protocolo e ao código utilizado para a implementação desse serviço. O Kerberos garante um meio de verificação de identidades numa *Open Network Architectur*e (ONA) [9], ou seja, numa rede onde todos os utilizadores possuem a mesma permissão de acessos a serviços. O Kerberos realiza a autenticação usando a abordagem *shared secret key* convencional.

A autenticação básica do Kerberos funciona da seguinte forma: um cliente envia um pedido ao *Authentication Server* (AS) para obter as credenciais para um dado servidor. O AS responde com estas credenciais, encriptadas na chave do cliente. As credenciais

consistem num *Ticket-granting Ticket* (TGT) para o servidor. O cliente transmite o TGT, que contém a identidade do cliente e uma cópia da chave de sessão, totalmente encriptadas pela chave do servidor, para o servidor. A chave de sessão, agora partilhada pelo cliente e pelo servidor, é usada para autenticar o cliente e pode ser utilizada opcionalmente para autenticar o servidor. Pode também ser utilizada para continuar a encriptar mensagens futuras, transmitidas entre as duas entidades, ou para trocar uma *sub-session key* em separado para ser usada na encriptação entre comunicações futuras.

A implementação do protocolo básico pode utilizar um ou mais AS. Estes AS mantêm uma base de dados onde os utilizadores, servidores ou serviços são representados por *principals* [10]. As suas chaves secretas também se encontram nesta base de dados. O serviço responsável pela geração de todos os *tickets* do Kerberos é o *Key Distribution Center* (KDC). Este é constituído pelo AS e o *Ticket-Granting Server* (TGS). A Figura 1, apresenta o funcionamento do protocolo Kerberos.

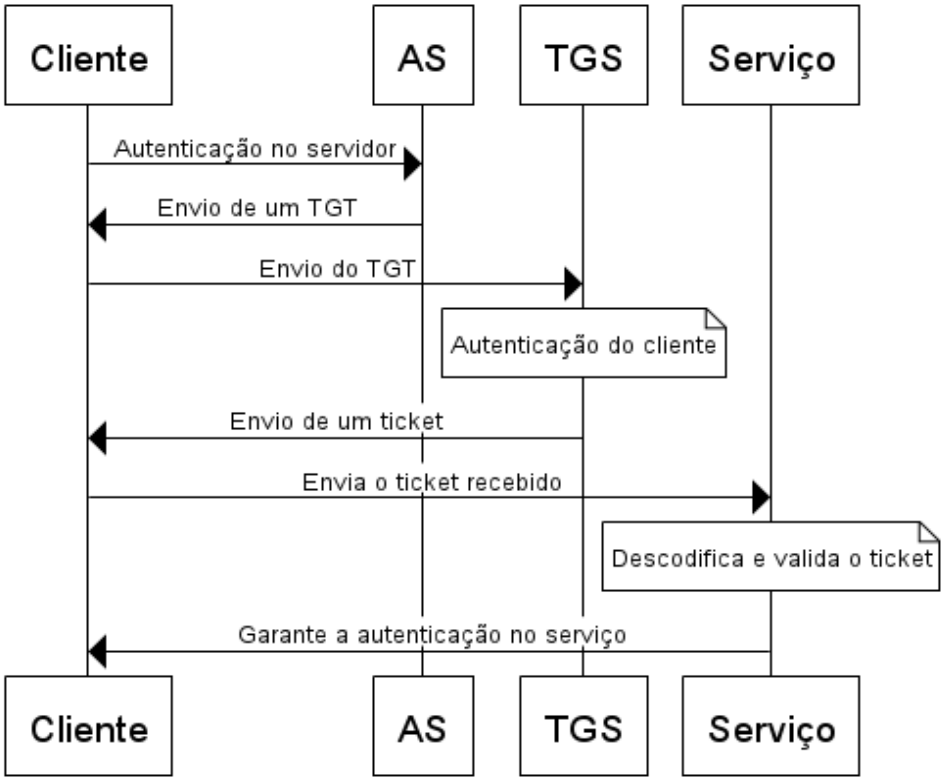


Figura 1 – Protocolo Kerberos

3.1.2. NEW TECHNOLOGY LAN MANAGER (NTLM)

O protocolo de autenticação NTLM [11] é usado em Windows para autenticação entre clientes e servidores. Foi introduzido no sistema operativo Windows 2000 Server tendo continuado a ser suportado em versões subsequentes. Este método continua presente, mas atualmente o método utilizado por omissão é o Kerberos. O NTLM é usado em protocolos de aplicação para autenticar utilizadores remotos, e, opcionalmente, fornecer uma sessão segura quando pedida pela aplicação. As mensagens NTLM são embebidas nos pacotes do protocolo da aplicação que requerem autenticação de um utilizador.

As credenciais NTLM [12] baseiam-se na informação obtida durante o processo de *login* interativo e consistem num nome de domínio, num nome de utilizador e na password encriptada desse utilizador.

Uma autenticação, que pode ser necessária para permitir acesso de um utilizador já autenticado a um recurso, tipicamente envolve três sistemas: um cliente, um servidor e um *Domain Controller* (DC) que faz os cálculos das autenticações no lugar do servidor. A Figura 2 representa este funcionamento.

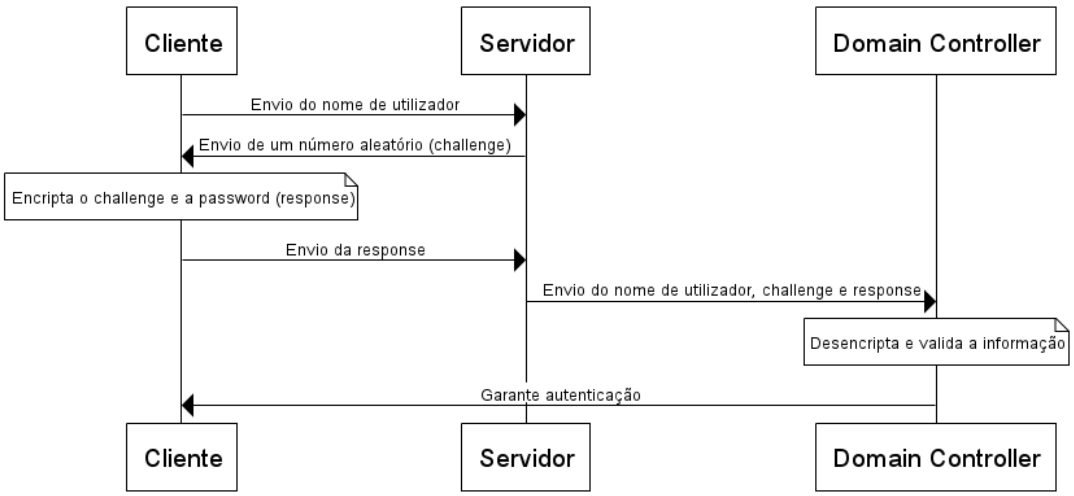


Figura 2 – Autenticação NTLM

3.1.3. SECURITY ASSERTION MARKUP LANGUAGE 2.0 (SAML 2.0)

Os protocolos descritos nos subcapítulos anteriores apresentavam formas de autenticação numa aplicação através de uma autenticação no domínio empresarial. As próximas tecnologias desenvolvem métodos de autenticação fora do domínio, necessitando uma

autenticação numa identidade federada (servidor externo) que partilhe o mesmo contexto de segurança.

O standard SAML [13] define uma *framework* XML para troca de informações de segurança entre entidades. Todas as informações enviadas são mensagens XML e têm o nome de asserções. Este *framework* introduz dois conceitos:

- *Identity Provider* (IDP) que é o sistema, ou domínio administrativo, que afirma a informação sobre o sujeito. Por exemplo: Este utilizador é o John Doe, que possui o email john.doe@acompany.com, e foi autenticado neste sistema usando um mecanismo de password.

- *Service Provider* (SP) que é o sistema, ou domínio administrativo, que depende da informação que lhe é enviada pelo IDP. É o SP que decide se confia nas asserções que lhe são enviadas. Deve se ter em conta que, apesar do SP poder confiar nas asserções que lhe foram transmitidas, a política de acesso local define se o sujeito deve ter, ou não, acesso aos recursos locais. Desta forma, apesar do SP confiar que o sujeito é o John Doe, não significa que este tem acesso a todos os recursos.

O SAML oferece a possibilidade de consolidar várias identidades locais numa identidade única (identidade federada – conceito desenvolvido nos próximos subcapítulos).

A sequência de SSO do SAML encontra-se ilustrada na Figura 3.

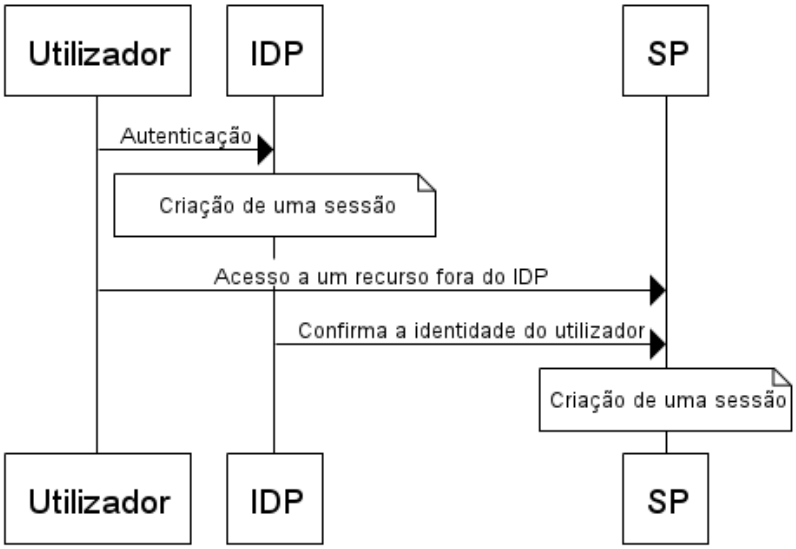


Figura 3 – SSO em SAML 2.0

O SAML define três tipos de asserções [14]:

- Autenticação: indicando que o sujeito foi autenticado previamente por algum meio (tal como uma password, hardware *token*, ou uma chave pública).
- Decisão de autorização: indicando que o sujeito deve ter acesso, ou não, ao recurso.
- Atribuição: indicando informação referente ao sujeito associado.

Usando um subconjunto de XML, o SAML define um protocolo pedido-resposta. Um pedido SAML tanto pode pedir uma asserção específica conhecida, como realizar pedidos de decisão relativos à autenticação, atribuição e autorização. A resposta SAML fornece as asserções do pedido.

A asserção XML [15] é um dos fatores diferenciadores deste serviço. O exemplo seguinte demonstra as possíveis componentes de uma asserção:

- Data da asserção.
- Entidade emissora.
- Informação do utilizador.
- Validade da asserção.
- Método de autenticação utilizado – por exemplo, *password-protected* (autenticação usando *Secure Sockets Layer* (SSL)).

3.1.4. OAUTH 2.0

O OAuth [16] é um protocolo de autorização que separa o papel do cliente e o do utilizador. Esta tecnologia define quatro papeis:

- Dono dos recursos (utilizador) – Uma entidade capaz de garantir acesso a um recurso protegido.
- Servidor de recursos – O servidor, que contém os recursos protegidos, capaz de aceitar e responder a pedidos na forma de *tokens* de acesso para obtenção destes recursos.

- Cliente – Uma aplicação que efetua o pedido ao recurso protegido, no lugar do utilizador e com a sua autorização.

- Servidor de autorização – O servidor que imite *tokens* de acesso para o cliente após autenticar com sucesso o utilizador e obter autorização.

No OAuth quando o cliente pede acesso a recursos controlados pelo servidor de recursos recebe um conjunto de credenciais diferentes daquelas presentes nesse servidor. Estas credenciais são depois verificadas e, se válidas, emitidas como *tokens* de acesso pelo servidor de autorização.

O servidor de autorização pode ser o mesmo que o servidor de recursos ou uma entidade separada. Um único servidor de autorização pode emitir *tokens* aceites por vários servidores de recursos.

O fluxo de mensagens para obtenção de *tokens* de acesso, como de *tokens* de atualização (para renovar a sessão) está ilustrado na Figura 4, onde o servidor de autorização é o mesmo que o servidor de recursos.

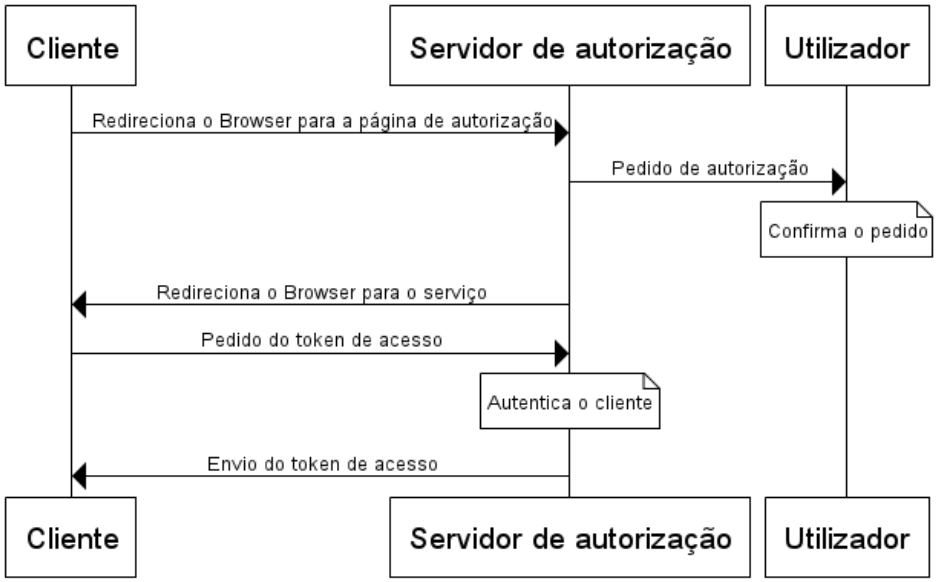


Figura 4 – Fluxo de mensagens de autorização

3.1.5. OPENID CONNECT

O OAuth 2.0 fornece uma *framework* geral para aplicações obterem e usarem recursos *Hyper Text Transfer Protocol* (HTTP), define mecanismos de obtenção e uso de *tokens* de

acesso para aceder a recursos, mas não define métodos padrão para fornecer informação de identidade. O OpenID Connect 1.0 [17] é uma camada de identidade simples, que funciona sobre o protocolo OAuth 2.0, permitindo a verificação de identidade do utilizador e a obtenção de informação básica de perfil.

Informação sobre a autenticação realizada é devolvida num JavaScript *Object Notation* (JSON) *Web Token* (JWT) chamado ID *Token*. Os servidores de autenticação OAuth 2.0 que implementam OpenID Connect são referidos como *OpenID Providers* (OPs) e os clientes como *Relying Parties* (RPs).

Um exemplo de fluxo mensagens no OpenID Connect encontra-se representado na Figura 5.

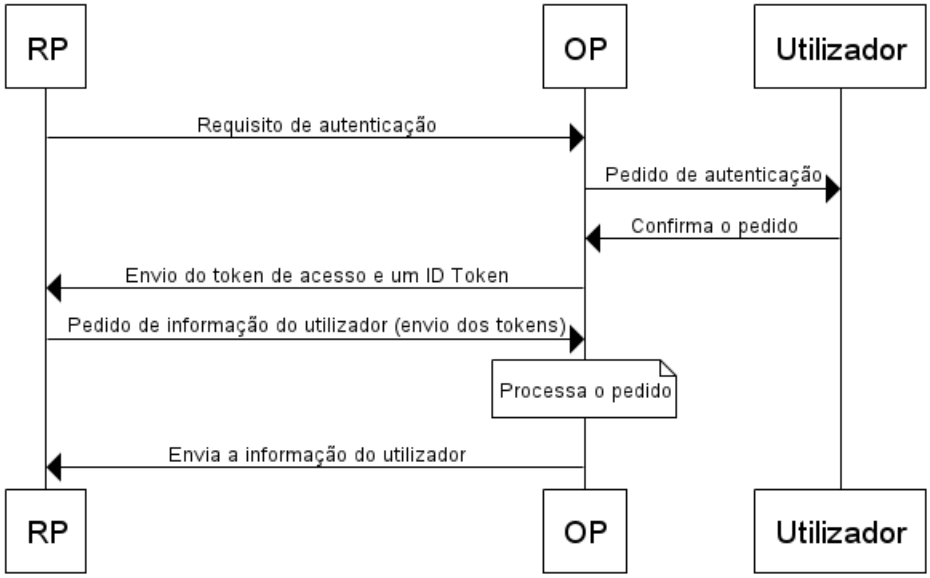


Figura 5 – Fluxo de mensagens no OpenID Connect

O ID *Token* é um *token* de segurança que contém informação sobre a autenticação de um utilizador num servidor de autorização, e é usado pelo cliente. Os campos seguintes encontram-se presentes no ID *Token*:

- `iss` (obrigatório) – ID do emissor para a entidade que emitirá a resposta.
- `sub` (obrigatório) – ID do sujeito. Um ID local único e nunca reatribuído pelo emissor ao utilizador. Este tem como objetivo ser utilizado pelo cliente.

- `aud` (obrigatório) – RPs que este *token* é alvo. Deve conter o `client_id` da RP como o valor.

- `exp` (obrigatório) – Tempo de expiração a partir do qual o ID *Token* não pode ser aceite. O processamento deste parâmetro requer que a data atual seja anterior à de expiração listada no valor.

- `iat` (obrigatório) – Tempo referente à emissão do JWT.

- `auth_time` – Tempo referente à altura da autenticação do utilizador. Quando este parâmetro é pedido como uma informação essencial, então este passa a ser obrigatório. Noutras situações é opcional.

- `nonce` – Valor usado para associar a sessão do cliente a um ID *Token*. O valor é passado, sem modificações, do pedido de autenticação para o *token*. Se este parâmetro estiver presente, os clientes têm que verificar se o valor é igual tanto no *token* como no pedido de autenticação.

- `acr` (opcional) – Identifica o contexto da autenticação realizada. As entidades a usar este parâmetro devem definir os valores a usar no contexto específico.

- `amr` (opcional) – Referências de métodos de autenticação.

- `azp` (opcional) – Grupo autorizado. O grupo ao qual o ID *Token* foi emitido. Se presente, este deve conter o ID de cliente deste grupo.

Os ID *Token* podem conter outras informações. Se alguma destas não for entendida será ignorada. O excerto seguinte representa um exemplo deste *Token*.

```
{   "iss": "https://server.example.com",
"sub": "24400320",
"aud": "s6BhdRkqt3",
"nonce": "n-0S6_WzA2Mj",
"exp": 1311281970,
"iat": 1311280970,
"auth_time": 1311280969,
"acr": "urn:mace:incommon:iap:silver"  }
```

3.2. SERVIÇOS DE DIRETÓRIO COM SSO

Quando se fala de uma solução de SSO tem que se ter em consideração o uso de serviços de diretório. Um serviço de diretório traz benefícios à organização e gestão de uma rede

empresarial pois este desenvolve uma estrutura hierárquica onde todos os postos se ligam sendo possível a administração e partilha de informação entre máquinas de uma forma facilitada. Estes serviços vêm, na sua maioria, acompanhados com uma solução de SSO.

O tipo de identidade definida nos serviços de diretório pode diferenciar o tipo de SSO que será usado neste serviço. Uma identidade de domínio é uma identidade dentro de um domínio apenas (por exemplo, uma empresa). Uma identidade federada [18] é uma identidade comum a vários domínios e que resulta de um acordo entre todas as identidades diferentes de um utilizador.

O SAML 2.0, OAuth 2.0 e o OpenID Connect utilizam métodos de autenticação e autorização de identidade federada. O Kerberos e o NTLM utilizam métodos que funcionam com identidades do domínio.

De seguida iremos investigar serviços de diretório amplamente usados.

3.2.1. ACTIVE DIRECTORY (AD)

O *Active Directory* (AD) [19] foi primeiro introduzido como parte do sistema operativo Microsoft Windows 2000 e está disponível como parte dos produtos de sistema operativo do Windows 2000 Server, do Windows Server 2003, entre outros.

O AD é um serviço de diretório [20] que atribui nomes do domínio a registos em objetos através de pedidos recebidos pelo *Lightweight Directory Access Protocol* (LDAP) [21]. Para além de LDAP o AD utiliza, normalmente, autenticação com base em Kerberos. O AD usando Kerberos garante SSO num ambiente empresarial (identidade de domínio). No entanto, este necessita da compra de uma licença, pois é um software privado.

O Windows Azure AD [22] é um serviço de diretório presente nos servidores Windows mais recentes. Estes AD funcionam com cinco serviços de diretório. O Azure não oferece diretamente estes cinco serviços, mas permite serem implementados. O mais familiar é o *Active Directory Domain Services* (ADDS) que usa Kerberos para autenticação com a ideia de juntar as máquinas do domínio, sendo fundamental para a gestão de uma infraestrutura. Outro destes serviços de diretório é o *Active Directory Federated Services* (ADFS) que tem como função a gestão de identidades federadas. A autenticação deste não é feita pelo Kerberos. Este é designado para funcionar não no interior da empresa, mas na Internet. O

equivalente do Kerberos no ADFS é o seu suporte de tecnologias como o SAML e o OAuth 2.0 + OpenID Connect para autenticação e autorização.

3.2.2. SAMBA 4

O Samba [23] é um software que simula o funcionamento de um servidor Windows em ambientes Unix com o propósito de garantir a partilha e gestão de ficheiros numa rede.

Esta tecnologia é uma componente importante para integrar transparentemente servidores e postos de trabalho Linux/Unix em ambientes AD. Pode funcionar tanto como um controlador (DC) ou como um membro regular do domínio.

Em mais detalhe, um servidor Samba oferece os seguintes serviços:

- Partilha de uma ou mais árvores de diretório.
- Partilha de uma ou mais árvores de sistemas de ficheiros distribuídos.
- Partilha de impressoras num servidor com clientes Windows na rede.
- Autenticação de clientes ligando-se a domínios Windows.

Com a versão 4 do Samba é possível a criação de um servidor com funcionamento idêntico ao AD do Windows. Esta funcionalidade permite autenticação Kerberos da mesma forma que um AD.

3.3. COMPARAÇÃO DOS DIFERENTES SERVIÇOS DE SSO

Em ambientes de autenticação através de uma identidade de domínio, a solução Kerberos é a mais usada devido à sua maior segurança quando comparada com o NTLM. Outra característica importante que poderá contribuir para a sua grande utilização é o facto de se tratar de uma solução não proprietária, sendo possível integrar-se Kerberos em outros sistemas para além de Windows. Já o NTLM é específico de Windows.

Com a ascensão dos serviços de *cloud*, aparecem aplicações que não suportam autenticação com identidade de domínio. Estas irão usar tecnologias de identidade federada como o SAML 2.0 ou, cada vez mais, OAuth 2.0 e OpenID Connect.

O OAuth 2.0 define um protocolo de delegação que é útil para decisões de autorização em redes de aplicações Web. Este é usado numa variedade de aplicações, incluindo mecanismos para autenticação de utilizador (por exemplo, OpenID Connect) [24]. Este é um protocolo de delegação de autorização, não de autenticação. Desta forma, caso se escolha implementar usando este protocolo, será necessária a integração do OpenID Connect para a autenticar o utilizador com segurança.

O ID *Token* do OpenID Connect traz ainda mais segurança aos *tokens* usados em OAuth 2.0. Isto deve-se à presença do campo `nonce` que permite validar a integridade da resposta e à necessidade do *token* de acesso passar por um processo de encriptação que protege o seu conteúdo.

O OAuth 2.0 juntamente com o OpenID Connect são protocolos que permitem a autenticação e autorização de um cliente, que pode pertencer a outros domínios, em relação a um recurso de um dado utilizador. O SAML 2.0, analogamente, efetua autorização e autenticação dando acesso de um dado recurso a um dado utilizador. Desta forma, podemos comparar o SAML 2.0 com o OpenID Connect + OAuth 2.0.

No SAML quando se efetua um pedido de autenticação para uma página *Web*, o utilizador é redirecionado para o IDP. No OpenID Connect o utilizador é redirecionado para o OP. Em ambas as situações o IDP/OP controla o *login* e evita expor segredos (por exemplo, passwords) à página *Web*. No SAML temos o SP, no Open ID Connect temos a RP. Em SAML as asserções são documentos XML com informações sobre o utilizador que autenticou, o emissor e outros dados. O equivalente de OpenID Connect é o ID *Token* no formato de JWT. A grande diferença entre estas abordagens é que a comunicação *back end* entre o RP e o OP no OpenID Connect transmite os *tokens*, entre estes, nunca os revelando ao Browser, enquanto que no SAML as asserções que serão usadas para autenticar o utilizador passam sempre pelo Browser.

O Kerberos efetua uma autenticação segura localmente, sendo uma alternativa que se encontra presente em vários serviços, como o AD, quer recente quer antigo, e o servidor Samba 4. Isto torna esta solução muito versátil. Esta versatilidade é fulcral quando se necessita de uma solução global que ofereça a todos os utilizadores do software a oportunidade de possuírem esta implementação.

Relativamente ao SAML e ao OpenID Connect + OAuth 2.0, o último seria mais vantajoso. Isto deve-se ao fator positivo da troca de *tokens* sem o acesso do Browser a estes, evitando ataques onde alguém que tem acesso ao Browser se faz passar pelo dono dos recursos para obter os seus *tokens*. No SAML a troca das asserções pelo Browser equivale a uma falha de segurança que é necessário e favorável evitar.

Tendo em conta os fatores abordados nos últimos parágrafos, no próximo capítulo será escolhida a implementação a usar.

4. ARQUITETURA DO SSO NO IPORTALDOC

De forma a estruturar e determinar a implementação de SSO a utilizar é necessário ter em consideração os cenários empresariais em que esta será utilizada, o funcionamento ideal da solução e os requisitos adicionais impostos pelo iPortalDoc.

4.1. CENÁRIOS

É necessário identificar as principais diferenças entre clientes da IPBRICK no que toca a servidores de diretório e servidores do iPortalDoc. Estas são representadas como diferentes cenários para a implementação do SSO.

A arquitetura a implementar terá que cumprir os requisitos de quatro cenários. A Figura 6 ilustra os quatro cenários presentes em clientes do iPortalDoc.

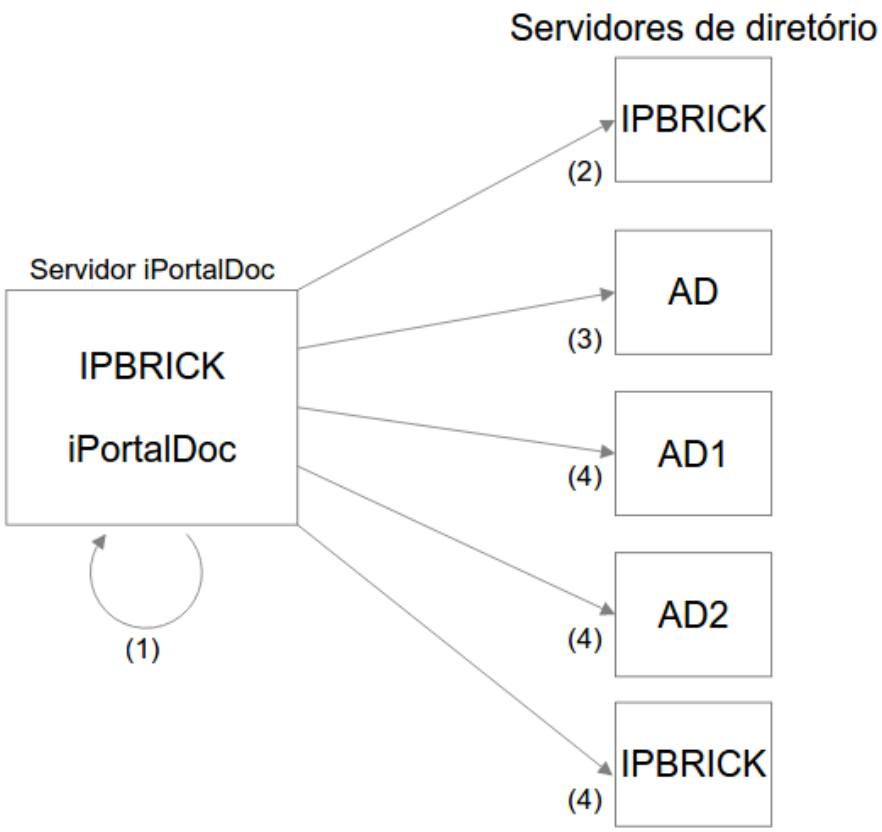


Figura 6 – Cenários da implementação

No primeiro cenário (1) o iPortalDoc encontra-se instalado no interior de um servidor IPBRICK que utiliza um serviço de diretório Samba 4 AD. Ou seja, o servidor IPBRICK realiza a funcionalidade de servidor de recursos e de servidor de diretório. Este cenário é utilizado nas empresas de tamanho muito reduzido que utilizam o iPortalDoc.

No segundo cenário (2) o servidor IPBRICK do iPortalDoc funciona como servidor de recursos enquanto que o servidor de diretório IPBRICK funciona como um AD de Samba 4.

O terceiro cenário (3) é semelhante ao segundo cenário, mas neste caso o servidor de diretório que será utilizado é um AD Windows. Tanto o segundo, como o terceiro cenário encontram-se presentes em empresas clientes do iPortalDoc de tamanho médio.

No quarto cenário (4) o servidor IPBRICK do iPortalDoc irá comunicar com vários servidores de diretório AD do Windows, ou servidores IPBRICK com AD do Samba 4. Este cenário encontra-se presente em empresas clientes do iPortalDoc de grandes dimensões, onde são necessários vários servidores de forma a dividir a carga por cada

servidor de diretório. A própria IPBRICK encaixa-se neste cenário, pois possui vários servidores diferentes com a funcionalidade de servidor de diretório.

O desenvolvimento focar-se-á em dois grupos de cenários: o grupo com servidores AD – cenários 3 e 4 e o grupo com servidores AD de Samba 4 – cenários 1, 2 e 4.

O funcionamento ideal da solução do SSO no iPortalDoc encontra-se apresentada na Figura 7.

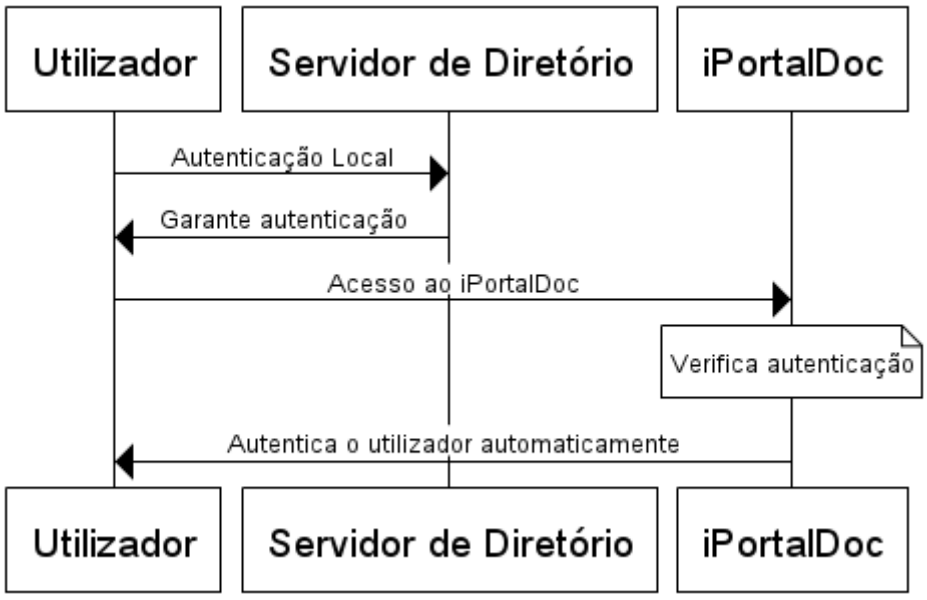


Figura 7 – Funcionamento ideal da implementação do SSO no iPortalDoc

4.2. REQUISITOS ADICIONAIS IMPOSTOS PELO IPORTALDOC

Para complementar a implementação do SSO, é necessário cumprir requisitos impostos pelo serviço Web iPortalDoc. Serão necessárias as seguintes configurações para o SSO:

- Módulo de SSO – ativa ou inativa o uso de SSO em todo o software.
- Variável de SSO presente em cada utilizador – define se o utilizador em questão utiliza ou não SSO.
- Variável do método de SSO – define qual método de SSO se encontra em uso no iPortalDoc.

Para além destas condições globais serão necessárias as seguintes mudanças:

- Elaboração de uma página que cria sessões automáticas no iPortalDoc para utilizadores de SSO.

- Criação de uma interface de configuração e gestão do SSO que permite a alteração dos valores das configurações.

- Automatização do SSO:

    - Introdução automática das condições necessárias à implementação na altura da instalação do software.

    - Configuração automática de servidores para utilizar no iPortalDoc.

4.2.1. CRIAÇÃO DE UMA SESSÃO NO IPORTALDOC

Sempre que um utilizador acede ao seu iPortalDoc introduz as suas credenciais para se autenticar. Se esta autenticação for sucedida será criada uma sessão para esse utilizador. Este funcionamento tem que ser coordenado de forma a criar uma sessão automaticamente. Para isto será precisa uma página *Web*, dentro de um local protegido, onde se inicia uma sessão apenas para utilizadores com permissão de acesso ao iPortalDoc. Esta página terá as seguintes funções:

- Identificar o utilizador e o seu servidor de autenticação.

- Através dos parâmetros anteriores verificar se existe um utilizador correspondente no iPortalDoc e se este se encontra ativo.

- Caso exista um utilizador e esteja ativo, testar se o SSO se encontra ligado neste utilizador.

- Após todas estas verificações, criar uma sessão para o utilizador.

- Redirecionar o utilizador para o iPortalDoc, quer para iniciar sessão com credenciais (caso o SSO esteja inativo), quer para começar a utilizar o software (caso esteja ativo).

- Fornecer o respetivo *feedback* ao utilizador de modo a garantir uma boa UX.

É necessário ter em consideração possíveis erros de configuração do servidor de diretório ou falta de permissões no acesso aos recursos protegidos. Nestes casos, o servidor do iPortalDoc tem que ter capacidade para redirecionar o utilizador para a página de autenticação com credenciais. Da mesma forma, se o módulo do SSO estiver inativo os utilizadores do iPortalDoc não serão redirecionados para a página de autenticação automática, mas sim para a página de autenticação com credenciais.

4.2.2. INTERFACE DE CONFIGURAÇÃO E GESTÃO DO SSO

O módulo de SSO é ativo ou inativo na página de interface de configurações já existente no software e apenas o administrador, ou utilizadores com permissões *superuser* (controlo total) podem alterá-lo. Na mesma página será possível aceder à interface de gestão do funcionamento deste módulo.

Nesta interface será possível:

- Escolher o método de SSO a usar.
- Pesquisar e filtrar os utilizadores por grupo e servidor de diretório e diferenciar os que utilizam, ou não, SSO – facilita a gestão em ambientes com muitos utilizadores e torna este processo agradável para os administradores do serviço *Web*.
- Ativar ou desativar o SSO dos utilizadores do iPortalDoc através da alteração da variável presente em cada utilizador. Esta pode ser realizada em massa através da seleção de vários utilizadores.

4.2.3. AUTOMATIZAÇÃO DO SSO

De modo a fornecer uma configuração facilitada do SSO será necessário automatizar este procedimento. Esta automatização pode ser descrita através dos seguintes passos:

- Introdução automática de variáveis do SSO na base de dados e criação ou edição de ficheiro necessários.
- Alteração da página de configuração de servidores do iPortalDoc para:
    - Requisitar informação necessária durante a introdução de um servidor de autenticação.

- Configurar e testar o SSO através da informação obtida.
- Enviar feedback adequado ao resultado do teste.

Se o *feedback* retornado após a configuração de um servidor for positivo, os utilizadores deste poderão utilizar SSO. Caso contrário, os utilizadores receberão uma mensagem indicando o problema em questão.

4.3. DECISÃO DA IMPLEMENTAÇÃO A USAR

O método de SSO a implementar será o Kerberos, este foi escolhido devido à sua segurança dentro do domínio empresarial, versatilidade e devido à incompatibilidade dos métodos mais recentes, como o SAML e o OpenID Connect + OAuth 2.0 com os clientes do iPortalDoc. Esta incompatibilidade deve-se à necessidade de um servidor de identidade federada (um IdP ou OP) e um ADFS (em Windows) para estes métodos. Isto não se encontra disponível para a maioria dos clientes do iPortalDoc, tornando estas implementações não ideais. Por outro lado, o ADDS e, consequentemente, o Kerberos encontram-se presentes em qualquer versão de servidores Windows, quer recente, quer antiga.

De modo a aumentar a compatibilidade e a gama de utilizadores que utilizam SSO, decidiu-se manter o funcionamento por Java *Applet*, como método de reserva, caso o Kerberos não se encontre configurado. Isto também melhora a UX dos utilizadores porque é preferível realizar SSO, mesmo sendo com um método problemático, do que não o realizar.

5. DESENVOLVIMENTO

Nos próximos subcapítulos irão ser apresentadas e demonstradas as implementações utilizadas para cumprir os objetivos do projeto. Estas são referentes aos procedimentos utilizados para a configuração inicial do Kerberos, à implementação global da integração do SSO no iPortalDoc e à introdução da biblioteca NOTY para substituição das mensagens de alerta do JavaScript.

Serão necessárias duas implementações iniciais do Kerberos. Estas complementarão os grupos de cenários presentes nas empresas clientes do iPortalDoc. A primeira implementação funcionará num ambiente onde o servidor de diretório utiliza um AD Windows e a segunda funcionará num ambiente onde o servidor de diretório é um servidor IPBRICK com um AD de Samba 4.

Todos os servidores IPBRICK, incluindo o do iPortalDoc, têm como base um servidor HTTP Apache [25] e usam uma distribuição Debian [26]. Assim, terá que se ter em consideração a compatibilidade da implementação com estes servidores.

5.1. KERBEROS COM AD

Para o grupo de cenários com AD como servidor de diretório a implementação do Kerberos tem que ser configurada como cliente do lado do servidor do iPortalDoc e como administrador do lado do AD.

Os servidores AD já têm o Kerberos configurado como administrador e funcionam como um KDC. As mudanças necessárias nestes são: a criação de um *principal* para o iPortalDoc mapeado a um utilizador e a configuração do *Domain Name System* (DNS) do AD para ser possível a comunicação com o servidor do iPortalDoc.

É necessário configurar o cliente Kerberos para comunicar com o AD. Assim é possível receber *tickets* dos utilizadores desse servidor de diretório e autenticá-los no iPortalDoc.

Foi necessário instalar um módulo chamado `mod_auth_kerb` [27]. Este módulo fornece autenticação Kerberos para servidores Apache e será útil para os passos presentes nos próximos subcapítulos.

5.1.1. COMUNICAÇÃO COM O SERVIDOR DE DIRETÓRIO

Um dos ficheiros responsáveis pela configuração do Kerberos é o `/etc/krb5.conf`. É neste que se indica o domínio de Kerberos que servirá como administrador e se configura definições do Kerberos cliente. O excerto seguinte demonstra este ficheiro:

```
[libdefaults]
      default_realm = IPDOCWS2008.LOCAL
[realms]
      IPDOCWS2008.LOCAL = {
            kdc =
ipdws2008.ipdocws2008.local
            admin_server =
ipdws2008.ipdocws2008.local
      }
      WIN2012.PT = {
            kdc = win2012.win2012.pt
            admin_server =
win2012.win2012.pt
      }
```

Para testar o cenário 3 será usado apenas um AD presente num Windows Server 2008 R2 (`IPDOCWS2008.LOCAL`). Para o cenário 4 serão usados dois ADs, um no mesmo Windows Server 2008 R2 e o segundo num Windows Server 2012 R2 (`WIN2012.PT`).

É necessário configurar os servidores de DNS a utilizar. Estes servidores terão que comunicar entre si e resolver os *Fully Qualified Domain Name* (FQDN) correspondentes. Como alternativa a esta configuração pode-se utilizar o *Internet Protocol* (IP) dos servidores diretamente na configuração do Kerberos.

A conectividade entre o cliente do Kerberos e o KDC do AD testa-se através do pedido de um *ticket* ao AD usando `kinit` [28] da seguinte forma:

```
kinit administrador@WIN2012.PT
```

Se a comunicação for realizada com sucesso, é pedida a password do utilizador administrador. Introduzindo a password corretamente será fornecido um TGT. Este pode ser visualizado através do comando `klist` que retorna uma resposta como a seguinte:

```
Ticket cache: FILE:/tmp/krb5cc_0
Default principal: administrador@WIN2012.PT

Valid starting    Expires           Service
principal
20/02/2017 16:08  21/02/2017 02:08
krbtgt/WIN2012.PT@WIN2012.PT
        renew until 21/02/2017 16:08
```

5.1.2. "KERBERIZAÇÃO" DO IPORTALDOC

Para "Kerberizar" o iPortalDoc é necessário criar um *principal* para o site. Como este *principal* serve uma aplicação e não um utilizador chama-se *principal* de serviço e tem que ser associado a um utilizador no AD. Este utilizador deve possuir uma password que não expire (para não ser necessário reconfigurar) e o mais segura possível. O excerto seguinte apresenta esta associação:

```
1° - setspn -A
HTTP/iportaldoc.ipdocws2008.local kerbdummy1
2° - ktpass
-princ
HTTP/iportaldoc.ipdocws2008.local@WIN2012.PT
-pass Kerber0s
-mapuser kerbdummy1
-Ptype KRB5_NT_PRINCIPAL
-out c:\krb52.keytab
```

Estes comandos irão também devolver um ficheiro do tipo *keytab* [29] que será usado pelo servidor para autenticar o iPortalDoc no AD. Assim é possível receber e desencriptar os *tickets* dos utilizadores deste serviço Web.

Para cada AD é necessário a criação de um utilizador para associar ao iPortalDoc e é necessário um ficheiro *keytab*. No cenário 3 foram necessários um ficheiro e um utilizador,

para o cenário 4 foram necessários dois destes ficheiros e dois utilizadores, um em cada servidor.

Quando se utiliza vários servidores ao mesmo tempo é necessário juntar todos os ficheiros *keytab* num só porque o Kerberos só lê o ficheiro presente no diretório de *keytab* da sua configuração. É importante definir o Apache como dono da *keytab* e dar-lhe apenas permissão de leitura, assim asseguramos a segurança do ficheiro.

É possível testar o funcionamento do Kerberos através da comparação dos *tickets* pedidos ao AD e os obtidos com a *keytab*. O excerto seguinte demonstra esta comparação, onde é possível observar o número dos dois *tickets* nas respostas aos comandos 1 e 2, verificando que são iguais e assim válidos.

```
1° - kvno
HTTP/iportaldoc.ipdocws2008.local@WIN2012.PT
Resposta:   kvno = 3
2° - klist -k -t /etc/krb5.keytab
Resposta: 3 09/05/17 11:55
HTTP/iportaldoc.ipdocws2008.local@WIN2012.PT
```

5.1.3. MOD_AUTH_KERB

De forma a permitir a comunicação entre o servidor Apache e o Kerberos é necessário a presença do módulo `mod_auth_kerb`. Para utilizar este módulo no iPortalDoc é necessário modificar o ficheiro referente à aplicação no Apache. Neste será adicionado um excerto de modo a chamar o módulo quando se abre o iPortalDoc:

```
<Location /secured>
      AuthType Kerberos
      Krb5KeyTab /etc/krb5.keytab
      KrbMethodNegotiate on
      KrbMethodK5Passwd off
      KrbServiceName Any
      KrbSaveCredentials on
      Require valid-user
</Location>
ErrorDocument 401 "<html> <meta http-
eqiv=\"refresh\" content=\"0;
url=http://?noker=1\" /></html>"
ErrorDocument 500 "<html> <meta http-
eqiv=\"refresh\" content=\"0;
url=http://?noker=1\" /></html>"
```

Os elementos deste excerto indicam o seguinte:

- `Location` – Indica a localização que o Kerberos irá proteger.
- `AuthType` – Define a autenticação a usar, ou seja, o Kerberos.

- `Krb5KeyTab` – Designa o caminho para o ficheiro *keytab*. Desta forma o Apache pode usar este ficheiro com o Kerberos.

- `KrbMethodNegotiate` – Permite a negociação do método para a autenticação (Kerberos por defeito).

- `KrbMethodK5Passwd` – Quando se encontra em `on` é apresentado um formulário de autenticação caso a autenticação com o Kerberos falhe. No iPortalDoc isto não é necessário, pois este já possui a página inicial para este propósito.

- `KrbServiceName` – A escolha `Any` representa o uso de qualquer tipo de serviço (por exemplo, HTTP).

- `KrbSaveCredentials` – Guarda as credenciais de um utilizador, de forma a que, após o ticket expire, ou caso seja necessário voltar a entrar no serviço, este não necessite de voltar a pedir as credenciais do utilizador, mas apenas um novo *ticket*.

- `Require valid-user` – Restringe acesso desta localização a utilizadores que não se autentiquem com o Kerberos. Isto é complementado com o comando `ErrorDocument 401` ou `500` que redireciona um utilizador para a página raiz com uma variável que identifica um *login* com credenciais. O erro `401` (falta de autorização) ocorre quando o utilizador não possuí permissões para aceder ao Kerberos. O erro `500` ocorre quando existem erros internos no servidor. A redirecção neste último caso tem como objetivo impedir que a página do Kerberos fique presa sem o utilizador poder realizar qualquer operação. Assim, o utilizador é redirecionado para a página de autenticação com credenciais.

Para validar a implementação foram realizados testes em três Browsers (Internet Explorer, Mozilla Firefox e Google Chrome). A Figura 8 demonstra o sucesso destes testes de funcionamento do Kerberos cumprindo assim os requisitos dos cenários 3 e 4.

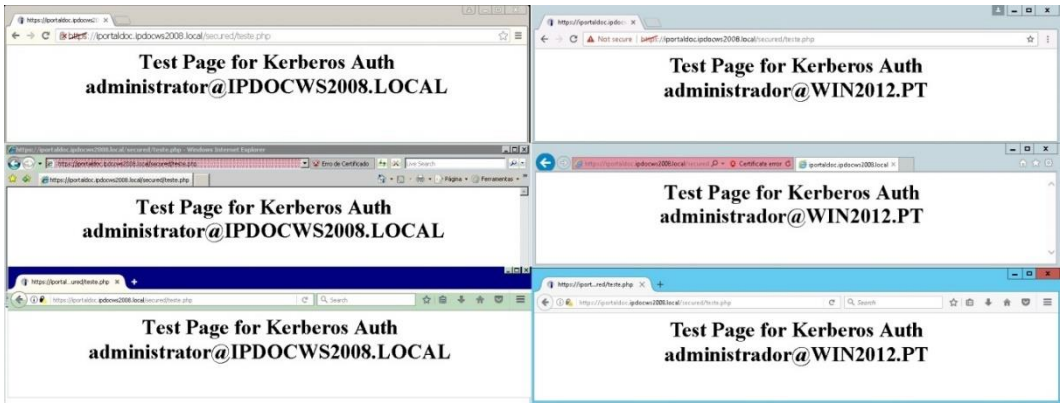


Figura 8 – Teste de funcionamento do Kerberos

5.2. KERBEROS COM SAMBA 4

Para o grupo de cenários com Samba 4 como servidor de diretório a implementação do Kerberos tem que ser configurada como cliente do lado do servidor do iPortalDoc e como administrador do lado do Samba 4 (no cenário 1 o servidor do iPortalDoc realiza ambas as funções).

Será necessária a criação de um *principal* para o iPortalDoc mapeado a um utilizador e a configuração do DNS do servidor Debian para ser possível a comunicação com o servidor do iPortalDoc. É necessário configurar o Kerberos, para comunicar com o Samba 4, de forma a obter *tickets* dos utilizadores e autenticá-los no iPortalDoc.

Os servidores IPBRICK possuem o Samba 4 instalado, mas não completamente configurado. Assim, será necessário terminar esta configuração.

5.2.1. CONFIGURAÇÃO DO SAMBA 4

Os servidores IPBRICK, atualmente, possuem uma versão de Samba, inferior à versão 4 completamente funcional. No entanto, apenas o Samba 4 é capaz de integrar e simular um servidor de diretório com funcionamento semelhante ao AD. Desta forma, será necessária a configuração de um servidor de diretório Samba 4 que funcione com a IPBRICK.

É necessário alterar o ficheiro de configuração do Bind9 [30], o serviço responsável pelo DNS no servidor IPBRICK, com a informação da zona (domínio) a criar. Nesta implementação o domínio do servidor IPBRICK é `ipdocdev10.net`.

Estando o Bind9 configurado é possível criar o servidor de diretório Samba 4 como DC principal. Para isto, é necessário utilizar o comando `samba-tool domain provision` [31] com atenção aos parâmetros seguintes:

- `Realm = ipdocdev10.net`: Reino de Kerberos.
- `Domain = ipdocdev10`: Domínio do servidor.
- `Server role = DC`: Papel do servidor.
- `Function level = 2008_R2`: Versão do AD a simular.
- `DNS-Backend = BIND9_DLZ`: DNS a utilizar.

Assim obtemos um servidor Samba 4 completamente configurado para funcionar com o Kerberos.

5.2.2. CONFIGURAÇÃO DO KERBEROS

Devido à semelhança do Samba 4 com o AD, a configuração do Kerberos é análoga à apresentada na implementação anterior, com exceção do procedimento do subcapítulo 5.1.2 onde esta configuração será diferente.

Caso se tenha seguido a implementação anterior do Kerberos até ao passo presente em 5.1.2, já terá sido instalado, no servidor do iPortalDoc, o módulo do Kerberos relativo à conexão com o Apache e já terá sido configurado o ficheiro `krb5.conf` de modo a detetar o reino desejado, que neste exemplo será o `IPDOCDEV10.NET`.

A diferença presente a partir deste passo da implementação será relativa à criação do *principal* de serviço do iPortalDoc. Este será criado no servidor Samba 4, através das ferramentas deste serviço, em vez das ferramentas do Windows. O excerto seguinte apresenta a criação deste *principal*:

```
# Criação de um utilizador para o iPortalDoc
samba-tool user create --random-password http-iportaldoc

# Remoção da data de validade da password do utilizador
samba-tool user setexpiry --noexpiry http-iportaldoc

# Criação do principal de serviço a partir do utilizador criado
samba-tool spn add HTTP/iportaldoc.ipdocdev10.net http-iportaldoc

# Exportação do ficheiro keytab relativo a este serviço
samba-tool domain exportkeytab /etc/krb5.keytab --
principal=HTTP/iportaldoc.ipdocdev10.net@IPDOCDEV10.NET
```

O procedimento seguinte será a mudança de permissões e de utilizador da *keytab* e, no servidor do iPortalDoc, a realização do procedimento em 5.1.3.

5.3. INTEGRAÇÃO COM O IPORTALDOC

De forma a atingir os requisitos impostos pelo iPortalDoc no que toca à solução de SSO foi desenvolvida uma página para a criação de sessão com o Kerberos, outra página responsável pela gestão do módulo e utilizadores de SSO e uma automatização das configurações do Kerberos, quando se introduz um servidor de diretório, ou quando se instala o iPortalDoc, de modo a tornar a configuração do Kerberos mais amigável para o administrador.

5.3.1. CRIAÇÃO DE UMA SESSÃO NO IPORTALDOC

Como no iPortalDoc é necessário que o utilizador tenha acesso ao site mesmo não possuindo uma sessão válida de Kerberos, foi criada uma pasta chamada `secured` onde os utilizadores que utilizam este SSO serão redirecionados. Esta pasta é o único recurso que restringe acesso a utilizadores sem Kerberos, e o seu conteúdo baseia-se numa página *Hypertext Pre-processor* (PHP), com o nome de `index.php`, que efetua a autenticação e inicia uma sessão no iPortalDoc.

Esta página efetua, inicialmente, os seguintes passos:

- Cria uma ligação às bases de dados necessárias ao funcionamento e à autenticação do iPortalDoc.
- Guarda e trata a informação do *principal*, através da leitura de variáveis globais estabelecidas pelo Kerberos.

- Obtém o, ou os servidores de autenticação a utilizar através de um pedido à base de dados.

- Utilizando a informação do *principal* realiza um pedido à base de dados para obter a conta do utilizador com o mesmo nome e domínio, se esta existir.

- Verifica se o SSO se encontra ligado, ou não, na conta em questão.

Caso não exista uma conta válida com o nome presente no *principal*, o utilizador é redirecionado para a página inicial do iPortalDoc para realizar autenticação com as suas credenciais. Se houver e esta possuir o SSO ligado será iniciada, ou renovada a sessão para o utilizador em questão sendo depois redirecionado para o seu iPortalDoc.

De acordo com o que foi abordado no capítulo de UX, uma página tem que fornecer o *feedback* necessário ao utilizador sobre o que se encontra em funcionamento. Assim, esta página fornece informação sobre a autenticação automática do utilizador, a acontecer em segundo plano. A Figura 9 apresenta o *layout* desta página.

Figura 9 – Página de autenticação Kerberos

Foram também necessárias as seguintes alterações na página inicial do iPortalDoc:

- Verificação do estado do módulo de SSO. Caso esteja inativo, a página irá apresentar o formulário para preenchimento das credenciais.

- Verificação do método de SSO que se encontra em funcionamento:

    - Se o escolhido for o Java *Applet*, a página irá executar esta aplicação.

    - Se o escolhido for o Kerberos, o utilizador será redirecionado para a página `index.php` na pasta `secured` para realizar o funcionamento Kerberos.

5.3.2. INTERFACE DE CONFIGURAÇÃO E GESTÃO DO SSO

Esta página foi concebida de forma a diminuir a carga de trabalho dos administradores durante a tarefa de gestão e tem duas funcionalidades principais: a escolha do método de SSO a utilizar (Kerberos ou Java *Applet*) e a escolha dos utilizadores que utilizam o SSO.

A primeira funcionalidade é concretizada pelos seguintes passos:

- A página realiza um pedido à base de dados para obter o valor da variável `SSO_TYPE`. Esta retorna 1 caso esteja em modo de integração com AD (Kerberos) ou 0 caso esteja em modo Java *Applet*.

- Sabendo este valor é apresentada a opção referente no formulário. Caso seja necessário a alteração do seu valor, o utilizador pode modificar a opção do formulário e pressionar o botão `Alterar`.

- Após o clique do botão será apresentada uma mensagem de confirmação com o *feedback* relativo às mudanças a realizar. Caso esta seja confirmada, é escrito o valor da opção na base de dados e a página é atualizada.

A Figura 10 representa esta secção da página e a mensagem de confirmação referente a esta alteração.

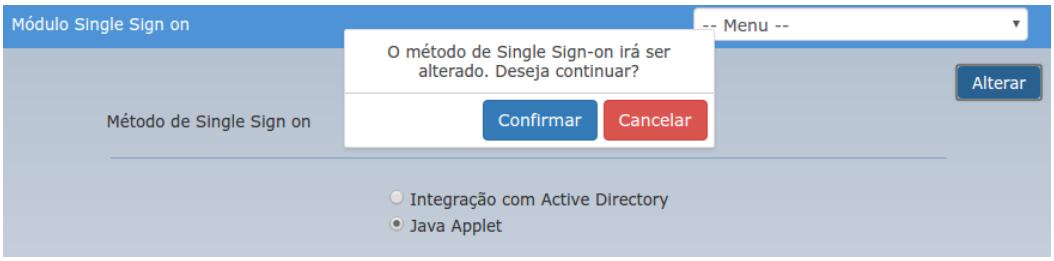


Figura 10 – Alteração do método de SSO

A segunda funcionalidade oferece pesquisas e filtragens de grupos e servidores de autenticação. Assim, o administrador do servidor consegue aceder facilmente aos utilizadores alvo.

Existe uma lista de servidores, uma lista de grupos e duas listas de utilizadores, uma para os que têm SSO inativo e outra SSO ativo. A Figura 11 ilustra estas listagens.

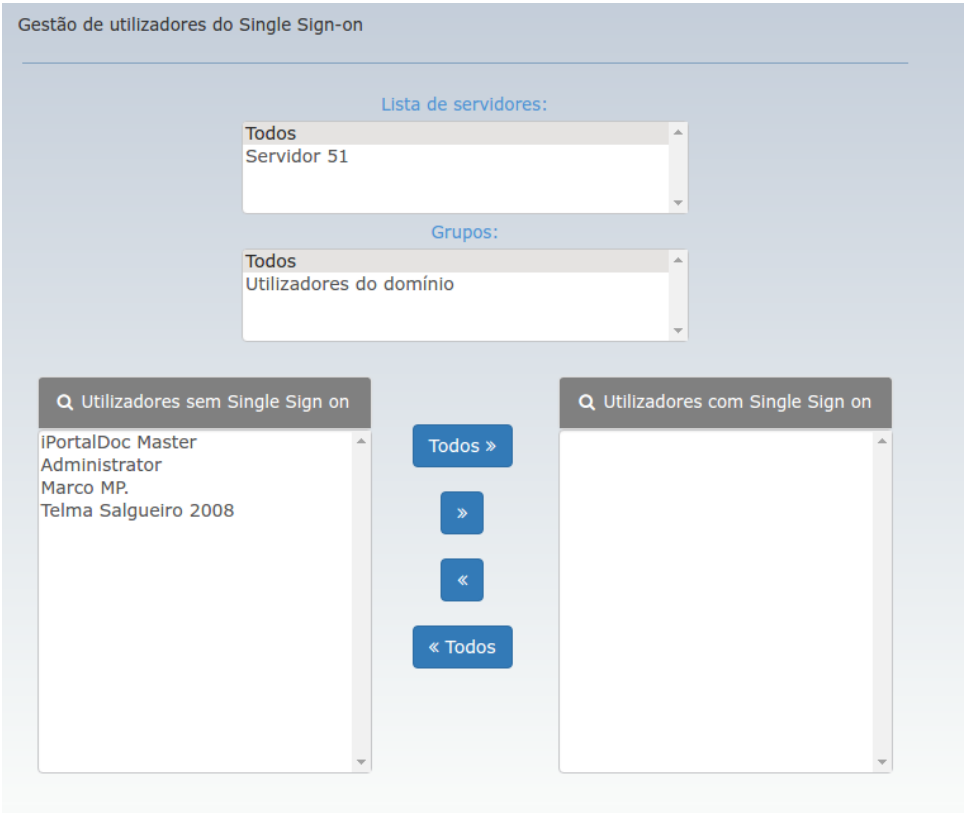


Figura 11 – *Layout* da interface de gestão de utilizadores do SSO

A filtragem é realizada através da escolha de uma opção das duas primeiras listas. Se esta for um servidor, apenas os grupos e utilizadores deste servidor ficarão visíveis. Se for um grupo, apenas os utilizadores deste grupo ficarão visíveis. Estas filtragens podem ser usadas em simultâneo.

A pesquisa de utilizadores pode ser utilizada individualmente para a lista de utilizadores com SSO ou para a lista de utilizadores sem SSO. Há que ter em conta que esta pesquisa é influenciada pelos filtros, podendo utilizar estas duas ferramentas para simplificar ainda mais a gestão de utilizadores.

Cada alteração efetuada nesta página é apresentada como *feedback* na altura da confirmação das mudanças. Um exemplo disto encontra-se na Figura 10. Caso o utilizador confirme as mudanças na página, o servidor guardará as alterações na base de dados.

5.3.3. AUTOMATIZAÇÃO DA CONFIGURAÇÃO KERBEROS

A automatização da configuração do Kerberos necessita de condições iniciais que serão geradas na altura da instalação ou atualização do iPortalDoc. Estas alterações baseiam-se na:

- Criação de uma variável para o método de SSO na base de dados: `SSO_TYPE` onde o valor 1 representa Kerberos e 0 representa Java *Applet*. Este passo é realizado através da criação de um *script* para realizar comandos diretamente na base de dados que será executado durante a instalação do software.

- Introdução de um ficheiro `krb5.conf` modelo na pasta de ficheiros do iPortalDoc.

- Instalação do `mod_auth_kerb` – realizada através de uma atualização ou instalação do servidor IPBRICK.

- Introdução automática das linhas presentes no excerto do subcapítulo 5.1.3 no ficheiro do Apache relativo ao iPortalDoc. Para este gerar as linhas desejadas automaticamente será necessário introduzi-las na base de dados.

Foi necessário alterar a página referente à introdução de servidores de diretório no iPortalDoc para configurar o Kerberos automaticamente durante esse processo. Nesta página quando se pressiona um botão de criação ou alteração de um servidor aparece um formulário que requisita informação ao utilizador. Para o Kerberos foi necessário a requisição de mais três campos: o reino Kerberos, o ficheiro *keytab* e o *Uniform Resource Locator* (URL) do iPortalDoc. A Figura 12 apresenta estes e outros campos.

Figura 12 – Campos necessários para o Kerberos

A informação nesse formulário é submetida para outra página que realiza testes e devolve o *feedback* correspondente. O primeiro teste realizado pela página é o teste de conexão com a LDAP. Se este teste for realizado com sucesso é possível começar a testar o Kerberos.

O primeiro passo é a edição do ficheiro `krb5.conf`. Durante o teste, o ficheiro editado será um ficheiro temporário, que será ignorado caso não se consiga comunicar com o reino do Kerberos ou gravado num ficheiro `krb5.conf` final caso este teste seja realizado com sucesso. Podemos verificar o conteúdo introduzido durante um teste no seguinte excerto:

```
[realms]
        IPDOCWS2008.LOCAL = {
                kdc = 172.31.4.51
                admin_server = 172.31.4.51
        }
```

Onde `IPDOCWS2008.LOCAL` representa o reino Kerberos e o `172.31.4.51` representa o IP.

Em seguida, executa-se o comando `kinit` para o reino em questão utilizando o nome de utilizador e a password do administrador. Caso este comando falhe, o utilizador receberá *feedback* sobre o que ocorreu e a configuração terminará com erro. Caso este comando seja sucedido, o servidor obterá um TGT para este utilizador e será possível testar o ficheiro *keytab*.

Este teste é semelhante ao do subcapítulo 5.1.2 e é resumido pelos seguintes passos:

- Requisição de um *ticket* para o iPortalDoc ao reino Kerberos especificado.
- Leitura da *keytab* carregada no formulário.
- Comparação do número do *ticket* devolvido no primeiro passo com o número presente na leitura do segundo passo.

Se estes números forem diferentes ocorre um erro de configuração, se forem iguais o teste é considerado um sucesso e o servidor é introduzido no iPortalDoc com SSO a funcionar.

A presença de erros durante a configuração impede o uso do Kerberos no servidor em questão, mas não impede a introdução deste servidor na aplicação.

5.4. BIBLIOTECA NOTY PARA FERRAMENTA DE NOTIFICAÇÃO

Para além do SSO, sempre houve necessidade de arranjar uma alternativa para os alertas de JavaScript. Isto encontra-se relacionado com os problemas de UX provenientes destes alertas. Desta forma, foi necessário arranjar uma alternativa. Esta teria que realizar as mesmas funções que os alertas anteriores, ou seja, identificar erros e pedir confirmações de alterações a realizar, mas sem causar transtornos ao utilizador durante o seu uso.

Para este efeito foi escolhida a biblioteca NOTY [32] que apresenta todas as capacidades das mensagens de JavaScript e possui funções não bloqueantes. Assim, o seu funcionamento nunca deixa um utilizador preso numa página, ao contrário do que acontece com os alertas anteriores.

O NOTY permite criar vários tipos de mensagem:

- Sucesso: caixa de mensagem com tons de verde.
- Alerta: caixa de mensagem branca.
- Aviso: caixa de mensagem com tons de amarelo.
- Informação: caixa de mensagem com tons de azul.
- Erro: caixa de mensagem a vermelho.

Cada um destes tipos pode ter um número ilimitado de botões, que utilizam classes para alterar o seu aspeto. Por exemplo, a classe `primary` apresenta um botão azul e a classe `danger` apresenta um botão vermelho.

Foram implementadas duas funções NOTY no iPortalDoc para simular o funcionamento das mensagens de JavaScript. O `alertNoty` e o `confirmNoty`.

É possível observar o `alertNoty` no seguinte excerto:

```
function alertNoty(mensagem){
    noty({
    text: mensagem,
    type: 'alert',
    layout: 'topCenter',
    buttons: [{
                addClass: 'btn btn-primary', text: 'Ok', onClick:
function($noty) {
                $noty.close();
                $.noty.closeAll()
                }
        }]
    });
}
```

Esta função tem apenas um parâmetro referente à mensagem a imprimir e apresenta uma caixa de mensagem do tipo `alert` com um botão que fecha a caixa quando pressionado. Esta função substitui efetivamente o `alert` em JavaScript. É possível observar o resultado desta função na Figura 13.

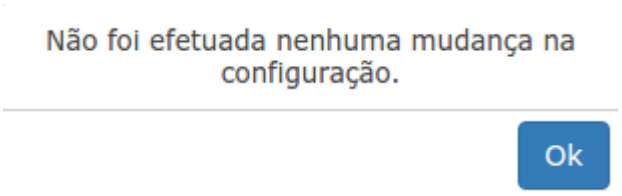


Figura 13 – Alerta NOTY

O NOTY já foi utilizado em todas as mensagens de *feedback* dos desenvolvimentos anteriores. Podemos observar o resultado da função `confirmNoty` que substitui o `confirm` do JavaScript na Figura 10.

6. SMART CARD E TOKEN USB

Um *smart card* ou um *token* USB são ferramentas que fornecem informação pessoal ou empresarial e podem ser usadas para validar uma identidade. Esta validação é realizada através de chaves presentes nestes dispositivos. Na solução atual do iPortalDoc usa-se o cartão de cidadão (*smart card*) para efetuar assinaturas digitais [33]. Uma assinatura digital serve para validar alterações realizadas, em informação, por uma entidade de confiança. Isto aumenta significativamente a segurança em trocas de informações com clientes.

6.1. CARACTERÍSTICAS DESEJADAS PARA A IMPLEMENTAÇÃO DA ASSINATURA DIGITAL

De forma a interagir com o cartão de cidadão o iPortalDoc atualmente recorre a um programa Java que apresenta entraves à sua utilização: problemas de compatibilidade entre as diferentes versões do Java, necessidade de descarregar o programa localmente para a sua execução e bloqueios relativos aos padrões de segurança definidos em cada máquina causando interrupções na execução do código em alguns dispositivos.

Assim, surgiu a necessidade de se encontrar uma solução com as seguintes características:

- Utilização do lado do cliente.
- Compatibilidade nos vários Browsers.
- Independência de sistema operativo em uso.
- Ausência de instalação de plugins, extensões ou executáveis.
- Versatilidade na gama de cartões e *tokens* compatíveis.

6.2. ANÁLISE DE SOLUÇÕES

Os cartões e os *tokens* USB podem ser utilizados em software através da *Application Programming Interface* (API), que define funções em C, especificada pela norma PKCS11 [34]. Esta apresenta uma interface onde qualquer um destes dispositivos é tratado de forma idêntica sendo identificados como *tokens* criptográficos.

Normalmente quando se introduz um leitor de *smart cards*, num computador, são instalados controladores para o PKCS11. Estes controladores estão restritos para o tipo de cartões compatíveis com o leitor.

A solução OpenSC [35] contraria esta restrição através da disponibilização de um controlador compatível com uma variedade elevada de *smart cards* e *tokens* USB.

A solução atual do iPortalDoc necessita de uma biblioteca Java do PKCS11 [36] e de um controlador. Nesta solução o programa é descarregado na máquina do utilizador e é executado, realizando a assinatura. Para uma solução global será necessário recorrer a tecnologias que não dependam de uma execução na máquina do utilizador, mas sim no Browser.

Quando os controladores dos cartões ou dos *tokens* USB se encontram instalados é possível ter acesso às suas chaves através de uma API própria dos Browsers. No entanto, esta API não se encontra disponível para interação e programação com o JavaScript.

A WebCrypto API [37] é uma interface comum entre vários Browsers que estabelece várias funções de criptografia. Entre estas encontram-se as seguintes:

- `verify` – verifica uma dada assinatura a partir de uma chave.

- `sign` – realiza uma assinatura com uma chave privada numa dada informação.

O grande entrave desta API é a ausência da deteção de chaves, tais como as dos *smart cards* e *tokens*. Nesta apenas é possível gerar, exportar ou importar chaves a partir de valores obtidos por outras fontes.

A WebCrypto Key Discovery [38] define a descoberta de chaves que não foram criadas a partir das funções da WebCrypto. Apesar desta nova proposta parecer apelativa esta não tem como alvo as chaves provenientes de *smart cards* ou *tokens* USB.

Há que ter em conta que esta API ainda não se encontra presente nos Browsers ao contrário da WebCrypto API que já se encontra disponível na maioria destes.

O trabalho apresentado em [39] propõe uma API para a descoberta de chaves provenientes de *smart cards* e *tokens*. Uma chave, quando necessária para uso numa página Web, seria requisitada ao utilizador, tomando como exemplo uma forma semelhante à requisição da permissão de uso da câmara Web do computador, mas com um botão para mostrar a lista de chaves dos certificados. Após isto, o utilizador teria que escolher a chave desejada e confirmar a sua seleção. Desta forma, garantia-se a segurança e o consentimento do utilizador. Quando estas chaves fossem usadas, o PIN do cartão ou *token* seria requisitado.

Esta API propõe a definição de uma classe chamada X509Certificate, um método para procurar certificados e chaves e um método para permitir a exportação destas. Tal como o WebCrypto Key Discovery, esta API não se encontra presente nos Browsers, existindo apenas um *plugin* realizado pelos responsáveis do artigo [40] que implementa o funcionamento desta API. Um *plugin* é tão problemático como a solução Java, pois também requer a instalação em cada Browser a utilizar e depende da versão e da sua compatibilidade.

6.3. SÍNTESE

O uso de *smart cards* e *tokens* USB é elevado principalmente em países europeus [41] onde os cartões de identificação do cidadão são deste tipo e são usados para várias tarefas de forma a garantir uma identidade. Assim, é difícil de crer a ausência de suporte ao desenvolvimento de alternativas após a remoção, na maioria dos Browsers, do suporte das Java *Applets*, que conseguiam realizar esta tarefa.

Sem se conseguir prever desenvolvimentos futuros é impossível ter a certeza se estas APIs alguma vez irão ser implementadas ou se aparecerão novas alternativas para este propósito. Por enquanto, a solução do iPortalDoc tem que ser mantida para assegurar o uso destes dispositivos para assinaturas digitais.

7. CONCLUSÕES

No início deste trabalho observava-se várias falhas a nível do SSO, em relação às mensagens de alerta JavaScript e na realização de assinaturas digitais com *smart cards* e *tokens* USB. De forma a contornar estes problemas foram propostas e implementadas soluções que permitiram melhorar os principais problemas identificados.

Em relação ao SSO, investigaram-se quatro métodos que poderiam ser introduzidos no software: o NTLM, o Kerberos, o SAML 2.0 e o OAuth + OpenID Connect. Destes concluiu-se que o Kerberos era o mais adequado pois é seguro e é o mais compatível com os clientes da IPBRICK SA.

A acompanhar este método foi criada uma página de *login* onde se inicia uma sessão no iPortalDoc para os utilizadores de SSO por Kerberos. Foi também elaborada uma página para gestão dos utilizadores do Kerberos, onde é possível pesquisar e filtrar utilizadores dos vários servidores e grupos de uma forma simples e agradável. Finalmente, foi introduzida uma vertente no software que configura automaticamente um servidor que utilize Kerberos através da introdução de informação desse domínio. Estas três componentes colaboram entre si e tornam esta solução de SSO uma excelente UX, quer para o utilizador regular, quer para o administrador de um servidor.

Tendo em conta os fatores abordadores anteriormente pode-se assim afirmar que a componente relativa ao SSO foi concluída com sucesso.

As mensagens de alerta do JavaScript foram substituídas pelo NOTY que funciona de forma idêntica a estes, mas pode ser personalizado para ter o aspeto desejado e não pode ser desativado pelo utilizador. Foi possível recriar funções como `alert` e o `confirm` para serem usados no iPortalDoc com esta biblioteca, sendo justo se afirmar que esta substituição foi implementada com êxito.

Em relação à assinatura digital com *smart cards* e *tokens* USB foram analisadas soluções, no lado do cliente, que poderiam servir como substitutas da implementação Java presente no software. Chegou-se à conclusão que o ideal seria a execução da tarefa num Browser, mas não foram encontradas soluções concretas para resolver o problema apresentado.

7.1. PERSPETIVAS FUTURAS

Em relação à solução de SSO implementada, seria possível a expansão desta para todas as aplicações da empresa e não só o iPortalDoc. Assim, os clientes da empresa teriam SSO em qualquer software e consequentemente uma UX superior.

Por outro lado, como a IPBRICK tem investido recentemente em tecnologias *cloud* para desenvolvimento e integração no seu sistema, abre-se a possibilidade para a utilização ou criação de um servidor de identidade federada, no futuro, para autenticação dos clientes das empresas. Existindo um servidor deste tipo seria possível o desenvolvimento de implementações mais recentes, tais como o SAML 2.0 ou o OAuth 2.0 + OpenID Connect. Estas poderiam ser integradas no iPortalDoc como outra alternativa para o SSO e nas restantes aplicações da IPBRICK como uma solução global para autenticação.

A primeira perspetiva é mais facilmente alcançada, pois já existem condições e uma implementação do Kerberos que pode ser utilizada como guia para as outras aplicações. A segunda poderá ser atualmente impraticável visto que nem todos os clientes se encontram preparados para as tecnologias *cloud*, o que prejudica a versatilidade de uma solução destas. No entanto, esta poderá ser uma solução a considerar assim que haja condições.

Em relação à assinatura digital com *smart cards* ou *tokens* USB, uma alteração benéfica para a solução atual seria a modificação dos controladores que esta usa. Atualmente, o executável Java utiliza o controlador do cartão de cidadão português, mas este pode utilizar

um controlador como o do OpenSC de forma a suportar *tokens* USB e cartões de vários fornecedores em vez de suportar apenas o cartão de cidadão português.

Referências Documentais

[1] Standard ISO 9241-210, International Organization for Standardization, hsevi.ir/RI_Standard/File/7436, acedido a 26 de janeiro de 2017.

[2] Ma Handa, Xue Anrong, Web Information System Construction Technology based on user experience, Jiangsu University.

[3] The definition of User Experience, https://www.nngroup.com/articles/definition-user-experience/, acedido a 26 de janeiro de 2017.

[4] Introduction to Usability, https://www.nngroup.com/articles/usability-101-introduction-to-usability/, acedido a 26 de janeiro de 2017.

[5] Ten Usability Heuristics, https://www.nngroup.com/articles/ten-usability-heuristics/, acedido a 26 de janeiro de 2017.

[6] Java plug-in does not work in Firefox, https://java.com/en/download/help/firefox_java.xml, acedido a 26 de janeiro de 2017.

[7] Java and Google Chrome Browser, https://java.com/en/download/faq/chrome.xml, acedido a 26 de janeiro de 2017.

[8] The Kerberos Network Authentication Service (V5), C. Neuman, T. Yu, S. Hartman, and K. Raeburn, https://tools.ietf.org/html/rfc4120, acedido a 12 de janeiro de 2017.

[9] Open Network Architecture, Federal Standard 1037, https://www.its.bldrdoc.gov/fs-1037/fs-1037c.htm, acedido a 12 de janeiro de 2017.

[10] Security Principals, Microsoft, https://technet.microsoft.com/en-us/library/cc780957(WS.10).aspx, acedido a 12 de janeiro de 2017.

[11] NT Lan Manager, https://msdn.microsoft.com/en-us/library/cc236627.aspx, acedido a 12 de janeiro de 2017.

[12] Microsoft NTLM, https://msdn.microsoft.com/en-us/library/windows/desktop/aa378749(v=vs.85).aspx, acedido a 12 de janeiro de 2017.

[13] Security Assertion Markup Language (SAML) 2.0 Technical Overview, OASIS, http://xml.coverpages.org/SAML-TechOverviewV20-Draft7874.pdf, acedido a 16 de janeiro de 2017.

[14] Wu kaixing and Yu xiaolin, A Model of Unite-Authentication Single Sign-On Based on SAML underlying Web, School of Information & Electronic Engineering, Hebei University of Engineering.

[15] SAML v2.0 Technical Overview, http://docs.oasis-open.org/security/saml/Post2.0/sstc-saml-tech-overview-2.0.pdf, acedido a 24 de janeiro de 2017.

[16] OAuth 2.0, IETF, https://tools.ietf.org/html/rfc6749, acedido a 23 de janeiro de 2017.

[17] OpenID connect 1.0, http://openid.net/specs/openid-connect-core-1_0.html, acedido a 23 de janeiro de 2017.

[18] Federated Identity, http://saml.xml.org/federated-identity, acedido a 25 de janeiro de 2017.

[19] Active Directory Technical Specification v20160714, Microsoft Corporation, https://msdn.microsoft.com/en-us/library/cc223122.aspx, acedido a 25 de janeiro de 2017.

[20] Active Directory Architecture, https://technet.microsoft.com/en-us/library/bb727030.aspx#EEAA, acedido a 25 de janeiro de 2017.

[21] LDAP, RFC 4511, https://tools.ietf.org/html/rfc4511, acedido a 25 de janeiro de 2017.

[22] Azure Active Directory and Normal Active Directory, https://blogs.msdn.microsoft.com/plankytronixx/2014/05/09/the-differencerelationship-between-azure-active-directory-and-normal-active-directory/, acedido a 25 de janeiro de 2017.

[23] About Samba, https://www.samba.org/samba/, acedido a 26 de janeiro de 2017.

[24] User Authentication with OAuth 2.0, https://oauth.net/articles/authentication/, acedido a 24 de janeiro de 2017.

[25] Apache HTTP Server Project, http://httpd.apache.org/, acedido a 19 de fevereiro de 2017.

[26] Debian, https://www.debian.org/index.pt.html, acedido a 19 de fevereiro de 2017.

[27] Kerberos module for apache, http://modauthkerb.sourceforge.net/, acedido a 19 de fevereiro de 2017.

[28] User Commands, https://web.mit.edu/Kerberos/krb5-1.13/doc/user/user_commands/index.html, acedido a 20 de fevereiro de 2017.

[29] Keytab, https://web.mit.edu/kerberos/krb5-1.13/doc/basic/keytab_def.html, acedido a 21 de fevereiro de 2017.

[30] Bind9, https://wiki.debian.org/Bind9, acedido a 15 de março de 2017.

[31] Samba-tool, https://www.samba.org/samba/docs/man/manpages-3/samba-tool.8.html, acedido a 15 de março de 2017.

[32] NOTY notification library, http://ned.im/noty/, acedido a 20 de março de 2017.

[33] Digital Signature Standard, DSS, http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf, acedido a 21 de abril de 2017.

[34] PKCS11 Standard, https://www.cryptsoft.com/pkcs11doc/STANDARD/pkcs-11v2-20.pdf, acedido a 24 de abril de 2017.

[35] OpenSC, https://github.com/OpenSC/OpenSC, acedido a 24 de abril de 2017.

[36] Sun PKCS11, http://docs.oracle.com/javase/7/docs/technotes/guides/security/p11guide.html, acedido a 24 de abril de 2017.

[37] WebCrypto API, https://www.w3.org/TR/WebCryptoAPI/, acedido a 24 de abril de 2017.

[38] WebCrypto Key Discovery, https://www.w3.org/TR/webcrypto-key-discovery/, acedido a 27 de abril de 2017.

[39] Using the W3C WebCrypto API for document signing, http://ceur-ws.org/Vol-1011/2.pdf, acedido a 24 de abril de 2017.

[40] WebCrypto Key Certificate Discovery, https://github.com/InventiveDesigners/webcrypto-key-certificate-discovery-js, acedido a 24 de abril de 2017.

[41] eID World Wide, https://www.tractis.com/help/?p=3670, acedido a 24 de abril de 2017.