

Mémoire de projet de fin d'étude

Présenté par :

Badr Eddine Zaim

En vue de l'obtention du diplôme

Licence Professionnelle

**Systèmes d'informations et analyse des
données**



**La mise en œuvre d'un système
d'authentification unique SSO avec CAS**

Soutenu le 18 Juillet 2023 devant le Jury composé de :

Encadrant interne :

Pr Nassim Kharmoum PA, CNRST

Encadrant externe :

M. Mohamed Boukhelif Ingénieur, CNRST

Membres du Jury :

Mme. Soumia Ziti

M. Sami Ait Ali Oulahcen

M. Anouar Riadsolh

PES, FSR – UM5R

Ingénieur, CNRST

PH, FSR – UM5R

Présidente

Examineur

Examineur

Année Universitaire 2022-2023

Remerciements

Au terme de ce stage, nous souhaitons exprimer notre profonde gratitude et nos sincères remerciements à notre encadrant, Monsieur Mohamed Boukhelif, pour son investissement sans faille. Nous sommes reconnaissants envers lui pour le temps précieux qu'il nous a consacré, ses conseils éclairés et la qualité de son suivi tout au long de notre projet.

Nous tenons également à remercier chaleureusement Monsieur Nassim Kharmoum, notre examinateur et professeur, pour son évaluation minutieuse de notre travail et ses commentaires constructifs. Sa contribution a grandement enrichi notre rapport de stage.

Nos plus vifs remerciements s'adressent également aux membres du jury, dont nous apprécions l'expertise et les échanges lors de la soutenance. Leurs observations et suggestions nous ont permis de consolider nos connaissances et d'améliorer notre rapport.

Nous souhaitons également exprimer notre gratitude envers Mme Jamila Al Alami, présidente du CNRST, pour avoir rendu possible cette opportunité de stage. Son soutien et sa confiance en notre travail ont été des moteurs essentiels de notre réussite.

Enfin, nous tenons à remercier toutes les personnes qui ont contribué de près ou de loin à l'élaboration de ce rapport de stage. Leurs conseils, leur soutien et leur expertise ont été précieux pour mener à bien ce projet.

Nous sommes profondément reconnaissants envers chacun d'entre vous pour votre précieuse contribution et votre accompagnement tout au long de ce stage. Vos efforts ont joué un rôle déterminant dans notre apprentissage et notre développement professionnel.

Résumé

Ce rapport présente la mise en œuvre d'un système d'authentification unique (SSO) basé sur CAS (Central Authentication Service). L'objectif de ce projet était de développer un mécanisme centralisé permettant aux utilisateurs d'accéder à plusieurs applications web sans avoir à se reconnecter à chaque fois.

Le rapport décrit les différentes phases du projet, notamment l'analyse des besoins, le développement du système SSO avec CAS, l'intégration de l'authentification LDAP, les tests et la documentation.

Les résultats obtenus ont démontré l'efficacité du système SSO dans la simplification de l'expérience utilisateur et l'amélioration de la sécurité des applications web.

Mots clés : SSO, CAS, LDAP

Abstract

This report presents the implementation of a Single Sign-On (SSO) system based on CAS (Central Authentication Service). The aim of this project was to develop a centralized mechanism that allows users to access multiple web applications without having to reauthenticate each time.

The report describes the different phases of the project, including requirements analysis, development of the SSO system with CAS, integration of LDAP authentication, testing, and documentation.

The results obtained demonstrated the effectiveness of the SSO system in simplifying the user experience and enhancing the security of web applications.

Keywords : SSO, CAS, LDAP

Table des matières

Remerciements	i
Résumé	ii
Abstract	iii
Introduction	viii
1 Contexte générale du stage	1
Introduction	1
1.1 Présentation de l'organisme	1
1.1.1 Missions	1
1.1.2 Administrations	2
1.1.3 La coopération scientifique et technique	2
1.1.4 Département	3
1.1.5 Division d'accueil	4
1.1.6 Organigramme du CNRST	5
1.2 Cahier des charges	5
1.2.1 Contexte	5
1.2.2 Description du projet	6
1.2.3 Contraintes et exigences	6
1.2.4 Problématique	6
1.2.5 Objectifs du stage	6
1.2.6 Diagramme de Gantt	7
Conclusion	7
2 Le Single Sign-On et Le mécanisme CAS	8
Introduction	8
2.1 Introduction au SSO	8
2.2 Avantages et intérêts du SSO	9
2.3 Architecture d'un SSO	10
2.3.1 SSO côté client	10
2.3.2 SSO côté serveur	11
2.3.3 SSO hybrides	13
2.4 Principaux protocoles SSO	14
2.4.1 Le Protocole (CAS)	14
2.4.2 Autre Protocoles	14

2.5	Présentation de CAS	15
2.6	Le mécanisme CAS	16
2.6.1	Architecture du serveur CAS	16
2.6.2	Fonctionnement de base	17
2.6.3	Fonctionnement multi-tiers	20
2.7	Présentation d'OpenLDAP	22
2.7.1	Introduction d'OpenLdap	22
2.7.2	Le fonctionnement de LDAP	22
	Conclusion	23
3	Mise en œuvre	24
	Introduction	24
3.1	Installation et configuration de CAS	24
3.1.1	Présentation du projet	24
3.1.2	Prérequis	24
3.1.3	Installation de CAS	26
3.1.4	Configuration de CAS	27
3.1.5	Exécution de CAS	28
3.2	Intégration de l'authentification OpenLDAP	31
3.2.1	Installation et configuration d'OpenLDAP	31
3.2.2	Liaison entre CAS et OpenLDAP	35
3.3	Réalisation du client CAS avec une application Spring Boot	36
3.3.1	Outils de développement de l'application	36
3.3.2	Configuration du client CAS	37
3.3.3	Utilisation du client CAS	40
	Conclusion	41
	Conclusion générale	42

Table des figures

1.1.1 Organigramme du CNRST	5
1.2.1 Diagramme de Gantt - Projet SSO avec CAS	7
2.3.1 SSO coté client avec un agent de serveur d'authentification	11
2.3.2 SSO coté client avec un agent de serveur d'authentification	12
2.3.3 SSO coté client serveur avec agent de reverse proxy	13
2.6.1 Architecture du serveur CAS	17
2.6.2 Premier accès d'un navigateur au serveur CAS	18
2.6.3 Authentification d'un navigateur auprès du serveur CAS	18
2.6.4 Redirection par le serveur CAS d'un navigateur vers un client CAS après authentification.	19
2.6.5 Validation du PT pour l'accès à une ressource	21
2.6.6 Fonctionnement de n-tiers.	21
2.7.1 Exemple d'un DIT de racine "dc=mon-entreprise,dc=com"	22
3.1.1 Clonnage et Construction.	26
3.1.2 Clonnage réussie.	27
3.1.3 Fichier cas.properties	27
3.1.4 Extrait du fichier log4j2.xml	28
3.1.5 Création de fichier keystore	29
3.1.6 Importer le keystore dans le JDK cacerts	29
3.1.7 Execution du serveur	30
3.1.8 Serveur est "Ready"	30
3.1.9 Interface d'authentification de CAS	31
3.2.1 le mot de passe administrateur	32
3.2.2 Confirmez le mot de passe	32
3.2.3 Executer Ldap dans ubuntu	35
3.3.1 Extrait de fichier pom.xml	38
3.3.2 Extrait de fichier application.yml	38
3.3.3 Code JSON dans les services de CAS	39
3.3.4 Interface d'authentification de CAS client	40
3.3.5 La Page du CAS apres l'authentification	41

Acronymes

CAS Central Authentication Service

LDAP Lightweight Directory Access Protocol

DIT Directory Information Tree

PT Proxy Ticket

PGT Proxy Granting Ticket

TGC Ticket Granting Cookie

SSO Single Sign-On

TS Ticket Service

IDE Integrated Development Environment

DNS Domain Name System

HTTP Hypertext Transfer Protocol

HTTPS Hypertext Transfer Protocol secure

URL Uniform Resource Locator

JDK Java Development Kit

SSL Secure Sockets Layer

Introduction

Dans un monde en constante évolution, les avancées technologiques dans le domaine de l'informatique ont un impact significatif sur la sécurité et l'authentification des systèmes. Dans ce contexte, le système d'authentification unique (SSO) joue un rôle essentiel en permettant aux utilisateurs d'accéder à plusieurs applications sans avoir à se reconnecter à chaque fois.

Ce rapport présente le travail réalisé dans le cadre de mon stage, dont l'objectif était de mettre en place un système de SSO avec CAS (Central Authentication Service) et d'intégrer l'authentification OpenLDAP. Le stage a été effectué au sein de CNRST, où j'ai pu bénéficier d'un environnement propice à l'apprentissage et à l'expérimentation.

Le rapport est structuré en plusieurs sections. Tout d'abord, le contexte général du stage est présenté, notamment en ce qui concerne l'organisme CNRST. Ensuite, une description détaillée sur SSO, CAS et OpenLdap, mettant en évidence ses avantages et les protocoles clés associés.

Le dernier chapitre dédié à la mise en œuvre de notre projet, nous allons procéder à l'installation et à la configuration de CAS (Central Authentication Service), OpenLDAP et à la liaison entre les deux systèmes. Nous aborderons également la réalisation du client CAS avec Spring Boot. Ces étapes essentielles nous permettront de mettre en place un système d'authentification centralisé (SSO) efficace, garantissant la sécurité et la simplicité d'accès aux applications.

Chapitre 1

Contexte générale du stage

Introduction

Le présent chapitre est consacré à la présentation du contexte général du stage, dans lequel s'inscrit notre projet d'intégration d'un système d'authentification centralisé (SSO) avec CAS et LDAP au sein de l'organisme CNRST (Centre National pour la Recherche Scientifique et Technique). Ce chapitre permettra de poser le cadre et de comprendre le contexte dans lequel s'inscrit notre projet. Il nous permettra également de présenter les objectifs du stage et de définir les principales étapes qui seront abordées tout au long de notre travail.

1.1 Présentation de l'organisme

Le Centre National pour la Recherche Scientifique et Technique est un établissement public doté de la personnalité morale et de l'autonomie financière. Il a été créé par Dahir (Décret Royal) du 5 août 1976 et a démarré ses activités en 1981. Il est actuellement placé sous la tutelle administrative du Ministère de l'Enseignement Supérieur, de la Formation des Cadres et de la Recherche Scientifique

1.1.1 Missions

Les missions du CNRST sont en relation étroite avec les nouvelles orientations qui lui sont confiées. Il sera notamment chargé :

- de mettre en œuvre des programmes de recherche et de développement technologique dans le cadre des choix et priorités fixés par l'autorité gouvernementale de tutelle
- de contribuer à la diffusion de l'information scientifique et technique, et à la publication de travaux de recherche et d'assurer des travaux de veille technologique
- d'apporter son concours au renforcement de l'infrastructure nationale de recherche
- d'effectuer des prestations de services au profit des opérateurs de recherche et de contribuer à la valorisation et au transfert des résultats de recherche

- d'établir des conventions ou contrats d'association, dans le cadre des activités de recherche ou des services, avec les établissements et organismes de recherche publics ou privés
- de créer des synergies entre les différentes équipes de recherche qui travaillent sur des thématiques prioritaires (réseaux, pôles de compétence)
- de procéder à l'évaluation et d'assurer le suivi de toutes les activités de recherche ou de services dans lesquelles il est impliqué.

1.1.2 Administrations

Le Centre National pour la Recherche Scientifique et Technique comprend, outre le Conseil d'administration et le Directeur mentionnés à l'article 5 de la loi du CNRST n° 80-00, une administration centrale et des unités de recherche propres et associées. L'administration centrale comporte :

- Le Conseil scientifique
- Les Comités scientifiques
- Le Département des Affaires scientifiques et techniques
- Le Département des Affaires générales et de la Coopération.

Le Conseil scientifique, présidé par le Directeur du Centre, est placé sous l'autorité directe du Conseil d'administration. Il est, conformément aux dispositions de l'article 10 de la loi précitée, chargé des questions scientifiques relatives aux activités du Centre

1.1.3 La coopération scientifique et technique

Le CNRST travaille en partenariat avec plusieurs autres organismes scientifiques et techniques étrangers à savoir :

- **CNRS** : Centre National de Recherche Scientifique français
- **DFG** : organisme allemand de promotion de la recherche
- **CNRI** : Conseil National de Recherche italien
- **CSIC** : Conseil Supérieur de la Recherche Scientifique espagnol
- **INSERM** : Institut National de la Santé et de Recherche Médicale français

- **ICCTI** : Institut de Coopération Scientifique et Technologique International portugais.

Le bénéfice de ses accords de coopération est ouvert à l'ensemble de la communauté scientifique marocaine. Des appels à projets sont régulièrement publiés dans la lettre d'information du Centre.

1.1.4 Département

Le Département Affaires Générales Financières et Systèmes d'Information a pour mission principale de coordonner les activités de recherche et d'assister la Directrice en matière de gestion et d'exécution des décisions du Conseil d'administration. Le Département des Affaires scientifiques et techniques comporte plusieurs divisions et services :

- Division Systèmes d'Information

- Service Etudes et Développement
- Service Production et Exploitation

- Division Affaires Générales

- Service de gestion des Ressources humaines
- Service Affaires Juridiques et Contentieux
- Service Patrimoine et Gestion Technique

- Division Affaires Financières

- Service Budget et Comptabilité
- Service Achats et Logistique
- Service Paiement

1.1.5 Division d'accueil

Mon stage s'est déroulé au sein du service "Production et Exploitation", qui fait partie de **la Division des Systèmes d'Information** au CNRST Rabat. Cette division est rattachée au Département des Affaires Générales Financières et Systèmes d'Information. Au CNRST Rabat, la Division des Systèmes d'Information est constituée de deux services principaux : le service "Études et Développement" et le service "Production et Exploitation". Cette division joue un rôle essentiel dans la gestion et l'optimisation des systèmes d'information de l'organisme.

- Le service "Études et Développement" est chargé de concevoir et de développer de nouvelles solutions informatiques adaptées aux besoins spécifiques du CNRST. Il travaille en étroite collaboration avec les différents départements pour identifier les défis et proposer des solutions novatrices.
- Le service "Production et Exploitation" est responsable de la mise en place et de la maintenance des systèmes informatiques en production. Son rôle est de garantir le bon fonctionnement et la performance des applications et des infrastructures, tout en assurant la sécurité et la disponibilité des données.

Ces deux services collaborent étroitement pour assurer le bon déroulement et l'évolution continue des systèmes d'information au CNRST. La Division des Systèmes d'Information fait partie intégrante du Département des Affaires Générales Financières et Systèmes d'Information, qui regroupe différentes entités chargées de soutenir et faciliter les activités administratives et financières de l'organisme.

1.1.6 Organigramme du CNRST

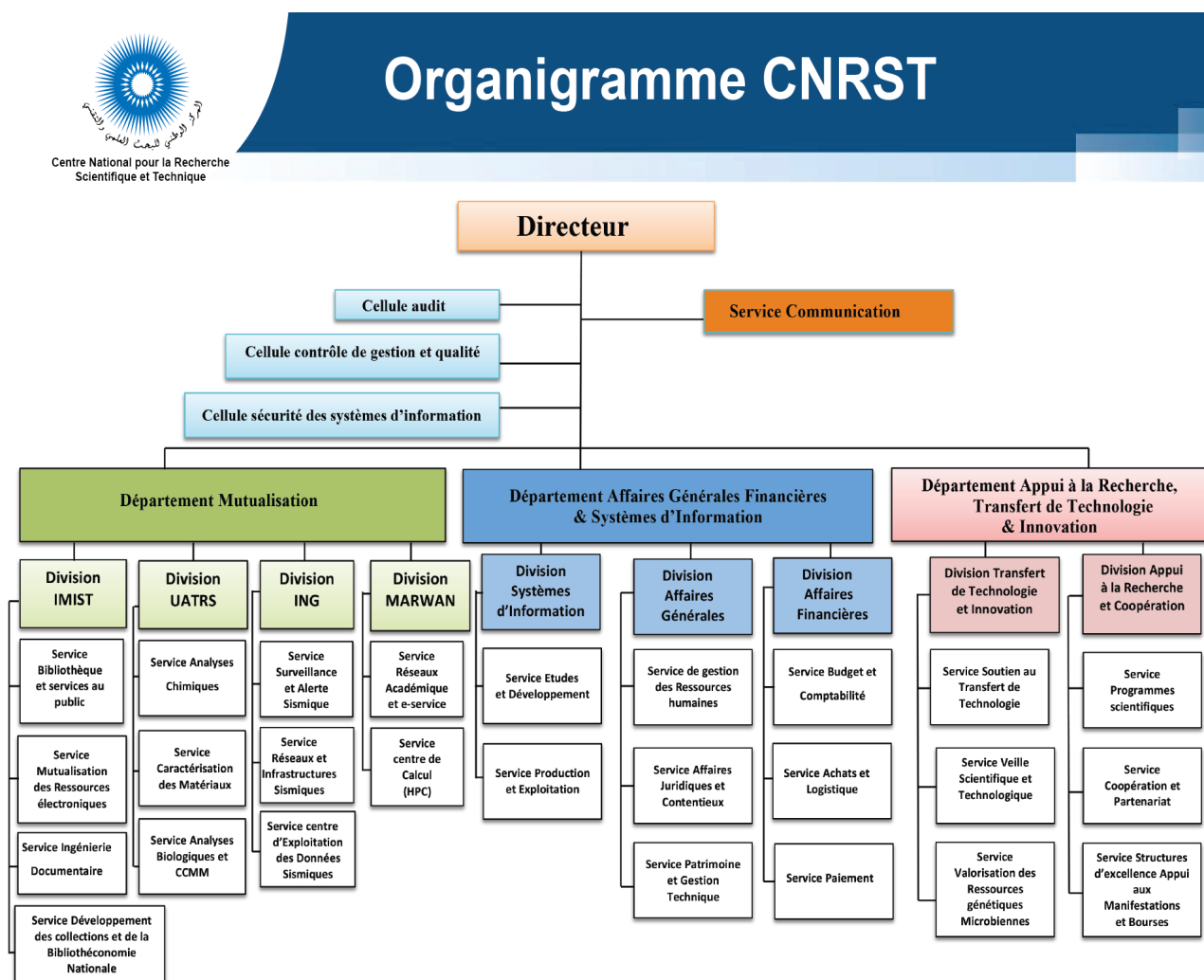


FIGURE 1.1.1 – Organigramme du CNRST

1.2 Cahier des charges

1.2.1 Contexte

Le stage se déroule dans le cadre d'un projet d'intégration d'un système d'authentification centralisé (SSO) avec CAS au sein de l'organisme CNRST. L'objectif est de mettre en place une solution SSO qui permettra aux utilisateurs d'accéder à plusieurs applications web sans avoir à s'authentifier à chaque fois.

1.2.2 Description du projet

Le projet consiste à intégrer le système d'authentification CAS (Central Authentication Service) à l'infrastructure existante de l'organisme CNRST, en incluant une connexion avec LDAP (Lightweight Directory Access Protocol). Cela implique la configuration du serveur CAS et du serveur LDAP, la modification des applications web pour prendre en charge l'authentification CAS avec LDAP, et la mise en place d'un mécanisme de synchronisation des utilisateurs et des rôles entre CAS, LDAP et les applications.

1.2.3 Contraintes et exigences

Le projet doit respecter les contraintes et les exigences suivantes :

- Assurer la sécurité des informations d'identification des utilisateurs.
- Permettre l'accès transparent aux applications web sans authentification répétée.
- Garantir l'intégrité et la confidentialité des données échangées.
- Assurer la compatibilité avec les technologies et les frameworks existants.
- Respecter les normes de développement et les bonnes pratiques de sécurité.

1.2.4 Problématique

La problématique du projet réside dans la complexité de l'intégration d'un système d'authentification centralisé avec les applications web existantes. Il est nécessaire de comprendre en détail le fonctionnement de CAS, de configurer correctement les applications pour utiliser CAS, et de mettre en place un mécanisme de synchronisation des utilisateurs et des rôles pour assurer une expérience utilisateur fluide.

1.2.5 Objectifs du stage

Les objectifs du stage sont les suivants :

- Comprendre le fonctionnement du système d'authentification CAS et son intégration avec LDAP.
- Configurer le serveur CAS et le serveur LDAP pour prendre en charge l'authentification CAS avec LDAP.
- Modifier les applications web pour utiliser l'authentification CAS avec LDAP.
- Mettre en place un mécanisme de synchronisation des utilisateurs et des rôles entre CAS, LDAP et les applications.
- Tester et valider l'intégration du système d'authentification centralisé avec connexion LDAP.
- Documenter le processus d'intégration, les configurations réalisées et les bonnes pratiques pour la maintenance future.

1.2.6 Diagramme de Gantt

- Semaine 1 : du 19/05/2023 au 25/05/2023
- Semaine 2 : du 26/05/2023 au 01/06/2023
- Semaine 3 : du 02/06/2023 au 08/06/2023
- Semaine 4 : du 09/06/2023 au 15/06/2023
- Semaine 5 : du 16/06/2023 au 22/06/2023
- Semaine 6 : du 23/06/2023 au 29/06/2023
- Semaine 7 : du 30/06/2023 au 06/07/2023
- Semaine 8 : du 07/07/2023 au 13/07/2023
- Semaine 9 : du 14/07/2023 au 20/07/2023

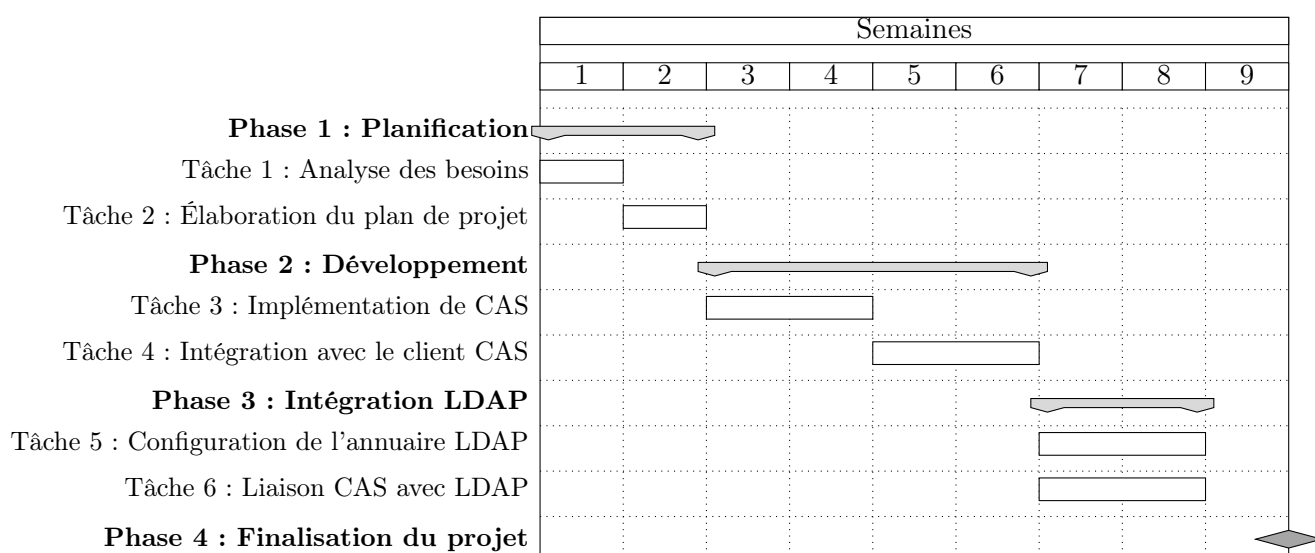


FIGURE 1.2.1 – Diagramme de Gantt - Projet SSO avec CAS

Conclusion

L'intégration d'un système d'authentification centralisé avec CAS et LDAP représente une avancée significative pour l'organisme CNRST en matière de gestion des accès aux applications. Cela permettra d'améliorer la sécurité, de simplifier l'expérience utilisateur et de faciliter la gestion des identités. Avec cette solution en place, l'organisme CNRST sera en mesure de mieux protéger ses systèmes et ses données tout en offrant aux utilisateurs un accès rapide et sécurisé aux ressources dont ils ont besoin.

Chapitre 2

Le Single Sign-On et Le mécanisme CAS

Introduction

L'authentification unique (Single Sign-On ou SSO) est un processus d'authentification qui permet aux utilisateurs ou aux clients de se connecter en une seule paire login/password non seulement à un domaine donné, mais aussi de bénéficier d'une authentification automatique à un autre domaine sans autre interaction de l'utilisateur. Dans ce second chapitre, nous allons présenter, d'abord, les différents concepts de l'authentification unique (SSO). Ensuite, nous poursuivrons en présentant le protocole d'authentification (Central Authentication Service) ainsi que l'annuaire LDAP (Lightweight Directory Access Protocol).

2.1 Introduction au SSO

Dans le domaine de l'informatique, le Single Sign-On (SSO), également connu sous le nom d'authentification unique, est un processus d'authentification qui se déroule de la manière suivante :

- Un utilisateur se connecte une seule fois à son poste de travail.
- Une fois connecté, l'utilisateur a accès à tous les ordinateurs et services pour lesquels il dispose d'autorisations, tant qu'il reste sur le même poste de travail.
- Lorsque l'utilisateur se déconnecte de ce poste, tous les droits d'accès sont annulés. La déconnexion peut se produire après un délai prédéfini ou lorsque l'utilisateur exécute manuellement une déconnexion unique (single sign-out ou single sign-off).

Le SSO permet à un utilisateur d'accéder à plusieurs applications liées les unes aux autres, mais indépendantes, en se connectant une seule fois, au lieu de saisir ses informations d'identification individuellement à chaque connexion à un logiciel. Cette procédure est largement utilisée tant dans un contexte privé (applications Web et Cloud privés) que

professionnel (applications utilisées au sein d'une entreprise et portail Intranet), en raison de sa convivialité.

2.2 Avantages et intérêts du SSO

Les organisations qui mettent en place une solution de SSO (Single Sign-On) bénéficient de nombreux avantages. Ils évitent les risques liés à la réutilisation des mots de passe, offrent une expérience utilisateur fluide et présentent d'autres avantages significatifs. Voici quelques-uns des principaux atouts de cette technologie :

- **Surface d'attaque réduite** : Le SSO élimine les problèmes liés à la gestion des mots de passe, réduisant ainsi la vulnérabilité de votre entreprise aux attaques de phishing. Les utilisateurs n'ont besoin de mémoriser qu'un seul mot de passe fort, ce qui limite les demandes de réinitialisation coûteuses en termes de temps et de ressources.
- **Accès utilisateurs fluides et sécurisés** : Le SSO fournit des informations en temps réel sur les utilisateurs qui accèdent aux applications, ainsi que sur le moment et l'emplacement de la connexion. Cela permet aux entreprises de protéger l'intégrité de leurs systèmes. Les solutions SSO réduisent également les risques de sécurité, par exemple en permettant la désactivation immédiate de l'accès d'un terminal professionnel perdu aux comptes et aux données critiques de l'entreprise.
- **Simplification de l'audit des accès utilisateurs** : La configuration des droits d'accès des utilisateurs en fonction de leur rôle, de leur département et de leur niveau hiérarchique peut être complexe dans un environnement professionnel en évolution constante. Les solutions SSO permettent de simplifier cet audit en offrant une visibilité sur les niveaux d'accès à tout moment, assurant ainsi la transparence et la conformité.
- **Utilisateurs plus autonomes et productifs** : Les utilisateurs attendent désormais un accès fluide aux applications nécessaires à leur travail. La gestion manuelle des demandes est fastidieuse et peut frustrer les utilisateurs. L'authentification SSO élimine le besoin de suivi manuel, permettant un accès immédiat à des milliers d'applications en un seul clic, ce qui favorise l'autonomie et la productivité des utilisateurs.
- **Pérennisation** : L'authentification SSO est la première étape pour protéger votre entreprise et ses utilisateurs. En utilisant cette technologie comme base, votre organisation peut mettre en œuvre d'autres bonnes pratiques telles que l'authentification multifacteur (MFA) et intégrer des outils de vérification d'identité, d'évaluation des risques et de gestion du consentement pour répondre aux exigences de conformité et réduire les fraudes. En commençant par une authentification SSO solide, vous mettez votre entreprise sur la voie d'une sécurité durable.

2.3 Architecture d'un SSO

L'architecture de la plupart des produits de SSO est inspirée de Kerberos. On y utilise largement sa terminologie et ces produits partagent ses concepts de base qui sont les suivants :

Les applications sont déchargées du travail d'authentification des utilisateurs ; Cette tâche est assurée par un serveur d'authentification dédié. Le serveur d'authentification délivre des tickets au client (maintien de la session d'authentification) et aux applications (transmission de l'identité de l'utilisateur)

- **Le serveur d'authentification** : Le serveur d'authentification est l'élément central du système de SSO. Il assure l'authentification de l'utilisateur qui lui fournit ses éléments d'authentification. Si le mode d'authentification est le mot de passe, la phase d'authentification implique la vérification de celui-ci auprès d'une base de référence. La plupart des systèmes de SSO implémentent plusieurs backend1 d'authentification (/etc./Password, NIS, LDAP)
- **L'agent d'authentification** : L'agent vérifie que l'utilisateur est authentifié ; s'il ne l'est pas, il le redirige vers le serveur d'authentification. Si le client est déjà authentifié auprès du serveur d'authentification, le serveur le redirige directement vers l'agent d'authentification demandeur. Les principaux modèles d'architecture SSO sont :
 1. SSO côté client (client-Side SSO).
 2. SSO côté serveur (server-Side SSO).
 3. Des systèmes hybrides.

2.3.1 SSO côté client

Le SSO côté client est présenté comme suite :

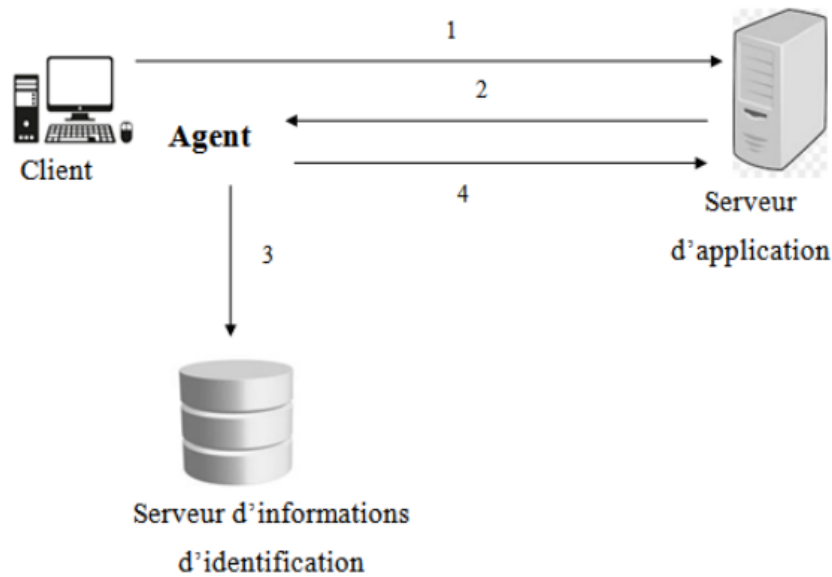


FIGURE 2.3.1 – SSO coté client avec un agent de serveur d'authentification

1. Le client souhaite accéder à une application.
2. L'application demande ses informations d'identification. L'agent présent sur le poste client intercepte la demande.
3. L'agent vérifie les informations d'identification dans le serveur centralisé des identifications.
4. L'agent simule l'utilisateur réel et envoie les informations d'identification à l'application. Ainsi, l'identification est transparente pour l'utilisateur

2.3.2 SSO côté serveur

Deux types de SSO « Server Side » existent : ceux qui utilisent un "Reverse Proxy" et ceux utilisant des agents "Serveurs"

Architecture avec agent serveur

La figure suivante présente l'architecture avec agent serveur

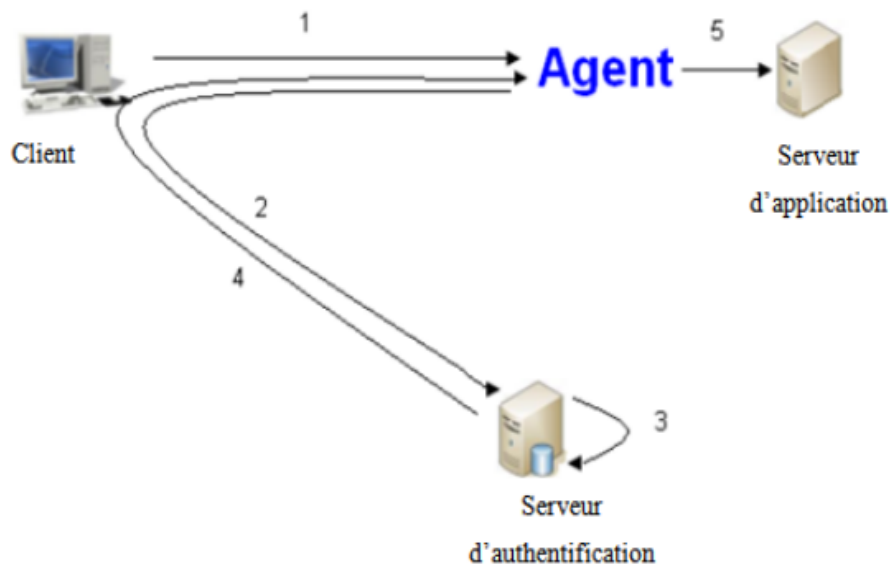


FIGURE 2.3.2 – SSO coté client avec un agent de serveur d'authentification

1. Le client souhaite accéder à une application. L'agent présent sur le serveur d'application intercepte la demande.

2. L'agent vérifie que l'utilisateur est authentifié : s'il ne l'est pas, l'agent se trouvant au niveau du client redirige la requête vers le serveur d'authentification. Cette redirection apparaît sous

forme d'un portail ou d'une fenêtre. L'utilisateur fournit ses informations d'identification pour le serveur d'authentification.

3. Le serveur d'authentification vérifie l'identité de l'utilisateur dans la base de données de référence.

4. Une fois l'utilisateur authentifié, le serveur d'authentification renvoie un cookie2 HTTP sur le poste de l'utilisateur qui permet de maintenir la session de l'utilisateur.

5. L'agent le transfère sur le serveur d'application

Architecture avec Reverse Proxy :

Cette architecture est présentée comme suite :

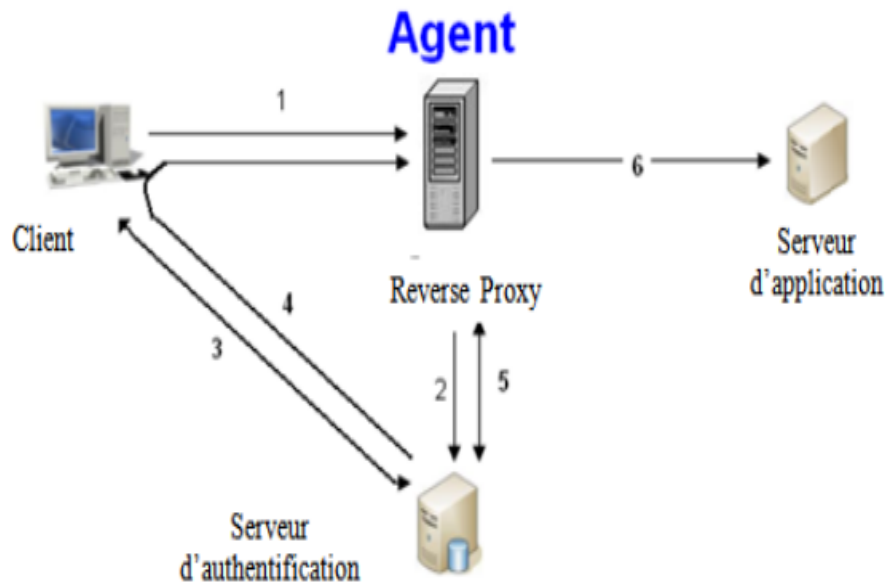


FIGURE 2.3.3 – SSO coté client serveur avec agent de reverse proxy

1. L'utilisateur tente de se connecter à l'application Web. Toute demande de connexion pour une application est redirigée vers le reverse proxy.

2. L'agent sur le reverse proxy intercepte la demande et vérifie l'authentification de l'utilisateur via le serveur d'authentification.

3. Si l'utilisateur n'est pas authentifié, le serveur d'authentification demande à l'utilisateur des informations d'identification. L'utilisateur fournit ses informations d'identification pour le serveur d'authentification.

4. Le serveur d'authentification envoie un jeton jouant le rôle de cookie et redirige le navigateur.

5. L'agent sur le reverse proxy intercepte la demande et vérifie l'authentification de l'utilisateur via le serveur d'authentification avec le jeton. Le serveur d'authentification envoie le login et l'autorisation des informations qui est associée avec le jeton.

6. L'agent permet d'accéder à la demande

2.3.3 SSO hybrides

Les approches dites hybrides combinent le client-side SSO et le server-side SSO et elles sont nombreuses. L'avantage de cette architecture est qu'elle permet de réduire les problèmes du SSO côté client

2.4 Principaux protocoles SSO

L'utilisation de la SSO à des fins d'identification nécessite de connaître différents protocoles et normes, parmi lesquels :

2.4.1 Le Protocole (CAS)

Intégration de CAS (**Central Authentication Service**) : CAS est un système open-source de gestion d'authentification centralisée qui permet la mise en œuvre de la SSO. Il fournit un service d'authentification unique pour plusieurs applications, permettant aux utilisateurs de se connecter une fois et d'accéder à différentes ressources sans avoir à s'authentifier à chaque fois. L'intégration de CAS dans une infrastructure de sécurité permet de simplifier la gestion des identités, d'améliorer l'expérience utilisateur et de renforcer la sécurité des applications.

2.4.2 Autre Protocoles

- **Security Access Markup Language (SAML)** : SAML est une norme ouverte qui encode du texte en langage machine et échange des informations d'identification. C'est devenu l'un des piliers de la SSO et est utilisé pour aider les fournisseurs d'applications à vérifier l'adéquation des demandes d'authentification. SAML 2.0 est spécifiquement optimisé pour une utilisation dans les applications web, permettant la transmission des informations via un navigateur.
- **Open Authorization (OAuth)** : OAuth est un protocole d'autorisation basé sur une norme ouverte qui transfère des informations d'identification entre des applications et les chiffre en code machine. Les utilisateurs peuvent ainsi autoriser une application à accéder à leurs données dans une autre application sans avoir besoin de valider manuellement leur identité, ce qui est particulièrement utile pour les applications natives.
- **OpenID Connect (OIDC)** : OIDC s'appuie sur OAuth 2.0 pour ajouter des informations sur l'utilisateur et faciliter le processus d'authentification SSO. Ce protocole permet d'accéder à plusieurs applications avec une seule session de connexion. Par exemple, un utilisateur peut se connecter à un service en utilisant son compte Facebook ou Google au lieu de saisir des identifiants.
- **Kerberos** : Kerberos est un protocole qui permet une authentification mutuelle, où à la fois l'utilisateur et le serveur vérifient l'identité de l'autre sur des connexions réseau non sécurisées. Il utilise un service de délivrance de tickets pour authentifier les utilisateurs et les applications logicielles, comme les clients de messagerie ou les serveurs wiki.
- **Authentification par carte à puce** : Au-delà de la SSO traditionnelle, il existe des équipements matériels qui permettent de réaliser le même processus, tels que les dispositifs de cartes à puce que les utilisateurs connectent à leur ordinateur.

Le logiciel de l'ordinateur interagit avec les clés cryptographiques de la carte pour authentifier l'utilisateur. Bien que les cartes à puce soient hautement sécurisées et nécessitent un code PIN, elles sont des dispositifs matériels que l'utilisateur doit transporter avec lui, ce qui présente un risque de perte. De plus, leur utilisation peut être coûteuse.

2.5 Présentation de CAS

Développé par l'Université de Yale, CAS (Central Authentication Service) est un serveur d'authentification accessible via le Web, composé de servlets Java, et compatible avec n'importe quel moteur de servlets (comme Tomcat). Les points forts de CAS sont énumérés ci-dessous.

La sécurité est assurée par les dispositifs suivants :

- le mot de passe de l'utilisateur ne circule qu'entre le navigateur client et le serveur d'authentification, nécessairement à travers un canal crypté.
- Les ré-authentifications suivantes sont faites de manière transparente à l'utilisateur, sous réserve de l'acceptation d'un cookie privé et protégé. Seul le serveur d'authentification peut lire et écrire ce cookie, qui ne contient qu'un identifiant de session.
- L'application reçoit du serveur d'authentification un "ticket opaque" qui ne contient pas d'informations personnelles. Ce ticket circule en clair via le navigateur (en tant que paramètre CGI) ; il n'est pas réutilisable, a une durée de vie courte et n'est utilisable que par l'application qui l'a demandé. L'application contacte ensuite directement le serveur CAS (via HTTP) pour valider (et expirer) ce ticket ; le serveur CAS renvoie à l'application l'identifiant de la personne, validé. Ainsi, l'application n'a jamais accès au mot de passe (ce qui est le schéma classique de la plupart des mécanismes de SSO).

Les mécanismes classiques imposent une communication entre le navigateur web l'application, ce qui exclut les configurations n-tiers, où une application doit directement interroger un service nécessitant authentification (c'est le cas par exemple pour un portail accédant à un web service). CAS, dans sa version 2.0, résout ce problème en proposant un mécanisme de mandataires (proxies). Des tickets dédiés permettent à des applications tierces, n'ayant aucune communication avec le navigateur client, d'être assurées de l'authentification de l'utilisateur. Cette fonctionnalité est assurément le point fort de CAS.

Le package proposé implémente tout le protocole de mise en oeuvre du SSO, à l'exception du module d'authentification locale qui est à la charge de l'administrateur du serveur d'authentification. Cela laisse la liberté d'implémenter exactement l'authentification souhaitée (LDAP, Kerberos, certificats X509, NIS, un panachage, ...).

Des bibliothèques clientes en Java, Perl, JSP, ASP, PL/SQL et PHP sont livrées. Cela permet une grande souplesse sur les serveurs d'applications. L'intégration dans des outils utilisés dans le monde universitaire est d'ores et déjà faite, comme celle d'uPortal.

L'utilisation de cookies exclusivement privés dans CAS (passage de tickets entre serveur d'authentification et applications uniquement sous forme de paramètres de GET HTTP) permet à CAS d'être opérationnel sur des serveurs situés dans des domaines DNS différents.

Un module Apache (mod-cas) permet d'utiliser CAS pour protéger l'accès à des documents web statiques, les bibliothèques clientes ne pouvant être utilisées dans ce cas.

Un module PAM (pam-cas) permet de "CAS-ifier" des services non web, tels que FTP, IMAP, ...

Enfin, CAS est en production dans plusieurs Universités américaines, avec des authentifications internes Kerberos ou LDAP, ce qui permet d'être confiant sur sa fiabilité

2.6 Le mécanisme CAS

2.6.1 Architecture du serveur CAS

L'architecture de CAS tourne autour de trois principaux acteurs : le serveur CAS, le client CAS et le navigateur.

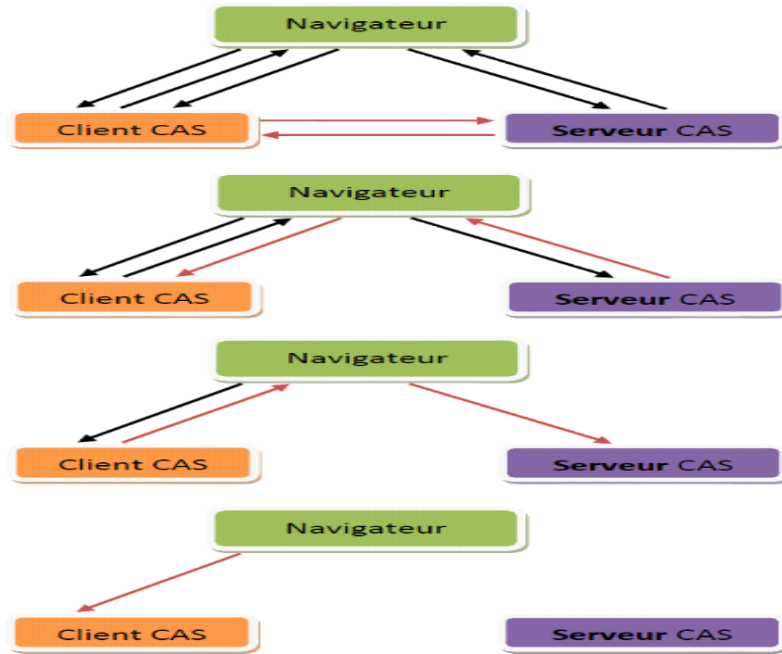


FIGURE 2.6.1 – Architecture du serveur CAS

- **CAS server** : le serveur CAS est l'élément central de l'authentification. Ce serveur est le seul acteur du mécanisme CAS qui relie les mots de passe des utilisateurs vers un annuaire LDAP centralisé. Son rôle est de :
 - (a) Authentifier les utilisateurs ;
 - (b) Certifier l'identité de la personne authentifiée aux CAS par son ticket de service.
- **CAS client** : un client CAS est tout fournisseur de services compatible avec CAS et pouvant communiquer avec le serveur. C'est un progiciel qui peut être intégré à plusieurs plates-formes logicielles et applications afin de communiquer avec serveur CAS .
- **Les navigateurs Web** : Ils doivent satisfaire les contraintes suivantes pour bénéficier de tout le confort de CAS :
 - (a) Permettre le protocole HTTPS ;
 - (b) Savoir stocker des cookies (en particulier, les cookies privés ne devront être retransmis qu'au serveur les ayant émis pour garantir la sécurité du mécanisme CAS).

2.6.2 Fonctionnement de base

Authentification d'un utilisateur

un utilisateur non authentifié, ou dont l'authentification a expiré, qui accède au serveur CAS se voit proposer un formulaire d'authentification , dans lequel il est invité à entrer ses informations d'authentications (login/email et mot de passe)

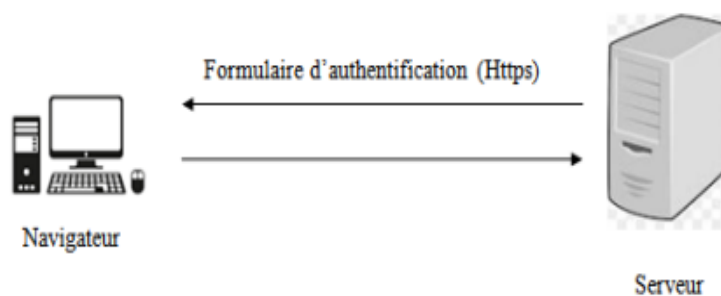


FIGURE 2.6.2 – Premier accès d'un navigateur au serveur CAS

Si les informations sont correctes, le serveur renvoie au navigateur un cookie appelé TGC (Ticket Granting Cookie) :

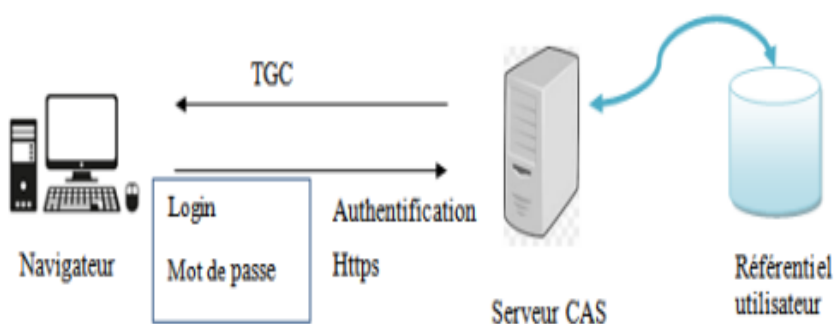


FIGURE 2.6.3 – Authentification d'un navigateur auprès du serveur CAS

Ticket Granting Cookie (TGC) : est le passeport de l'utilisateur auprès du serveur CAS. Ayant une durée de vie limitée, TGC est le moyen pour les navigateurs d'obtenir auprès du serveur CAS des tickets de services sans avoir à se ré authentifier. C'est un cookie privé et protégé. Il est aussi opaque tout comme les autres tickets utilisés dans le mécanisme CAS .

Accès à une ressource protégée après authentification

la figure ci-dessous représente l'ordre d'accès aux ressources protégées par CAS

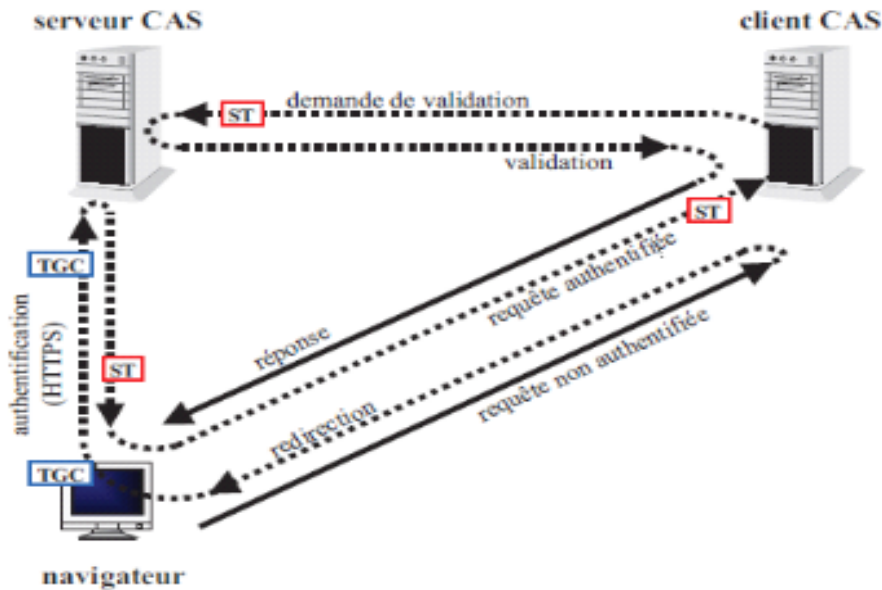


FIGURE 2.6.4 – Redirection par le serveur CAS d'un navigateur vers un client CAS après authentification.

Les étapes sont :

- Le navigateur tente d'accéder à une ressource protégée sur un client CAS
- Le client redirige le navigateur vers le serveur CAS dans le but d'authentifier l'utilisateur
- Le navigateur, précédemment authentifié auprès du serveur CAS, lui présente le TGC
- Dès que le TGC apparaît sur le serveur CAS, celui-ci émet un Service Ticket (Service Ticket ou ST) au navigateur ; il s'agit d'un ticket opaque qui ne comporte aucune information personnelle ; il n'est accessible que par le « service » (L'URL) le demandant
- Dans le même temps, le serveur CAS redirige le navigateur vers le service demandeur en passant le ST en paramètre
- Le ticket de service est ensuite validé par le client CAS directement en http avec le serveur CAS
- La ressource est livrée au navigateur

Accès à une ressource protégée sans authentification préalable

Si l'utilisateur n'est pas déjà authentifié auprès du serveur CAS avant d'accéder à une ressource, son navigateur est, comme précédemment, redirigé vers le serveur CAS qui lui propose alors un formulaire d'authentification . Lors de la soumission du formulaire par le navigateur au serveur CAS, si les informations fournies sont correctes, le serveur CAS :

- Emet un TGC au client, qui lui permettra ultérieurement de ne pas avoir à se reauthentifier ;
- Délivre au client un Service Ticket à destination du client CAS ;
- Redirige le client vers le client CAS

2.6.3 Fonctionnement multi-tiers

Les mandataires (Proxies) CAS

Le fonctionnement n-tiers de CAS consiste en la possibilité, pour un client CAS, de se comporter comme un navigateur. Un tel client CAS est alors appelé mandataire (proxy) CAS.

Les exemples d'utilisation de mandataires sont divers :

- **un portail web**, pour lequel l'utilisateur s'est authentifié, peut avoir besoin d'interroger une application externe sous l'identité de l'utilisateur connecté (un web service par exemple) ;
- **une passerelle web** de courrier électronique (webmail), à laquelle un utilisateur s'est authentifié, a besoin de se connecter à un serveur IMAP pour relever le courrier électronique de l'utilisateur, sous son identité. Dans le fonctionnement de base, le client CAS est toujours en lien direct avec le navigateur. Dans un fonctionnement n-tiers, le navigateur accède à un client CAS à travers un mandataire CAS. Le mécanisme de redirection vu dans le fonctionnement de base n'est alors plus applicable.

Fonctionnement 2-tier

Un mandataire CAS, lorsqu'il valide un Service Ticket pour authentifier un utilisateur, effectue, dans le même temps, une demande de PGT (Proxy Granting Ticket).

Le Proxy Granting Ticket (PGT) est le passeport d'un mandataire CAS, pour un utilisateur, auprès du serveur CAS. Le PGT est opaque, rejouable, et obtenu du serveur CAS par un canal chiffré (et un mécanisme de callback assurant son intégrité). Comme le TGC, il est à durée de vie limitée (typiquement quelques heures).

Le PGT est l'équivalent, pour les mandataires CAS, des TGCs pour les navigateurs. Il permet d'authentifier l'utilisateur auprès du serveur CAS, et d'obtenir ultérieurement du serveur CAS des Proxy Tickets, équivalents pour les mandataires des Service Tickets pour les navigateurs.

Les Proxy Tickets (PT) sont, comme les Service Tickets, validés par le serveur CAS pour donner accès à des ressources protégées

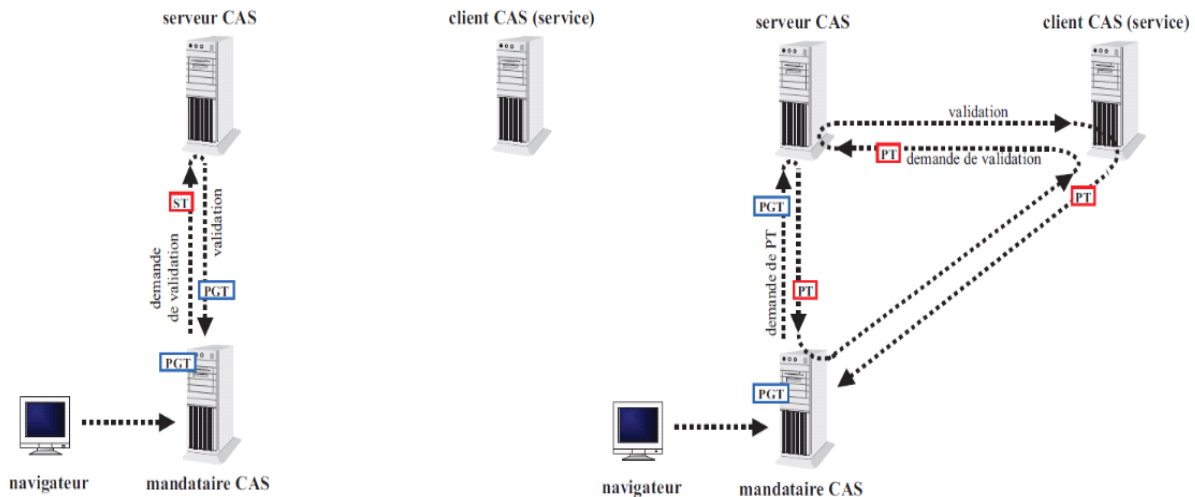


FIGURE 2.6.5 – Validation du PT pour l'accès à une ressource

Le Proxy Ticket (PT) est le passeport des mandataires CAS auprès des clients CAS. Il est opaque, non rejouable, et à durée de vie très limitée (comme le Service Ticket, typiquement quelques secondes).

Fonctionnement n-tier

Il est clairement démontré que le client CAS, accessible par le mandataire CAS dans un système du fonctionnement 2-tiers, peut également agir en tant que mandataire à son tour. Cela permet la création d'une chaîne de mandataires. CAS est actuellement le seul mécanisme de SSO connu offrant cette fonctionnement multi-tiers sans propagation de mot de passe

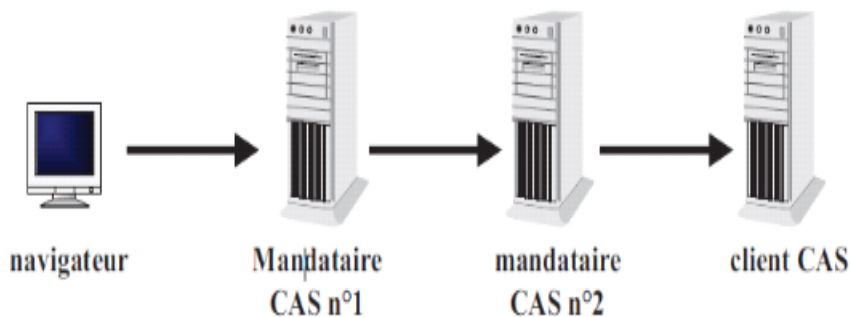


FIGURE 2.6.6 – Fonctionnement de n-tiers.

2.7 Présentation d'OpenLDAP

2.7.1 Introduction d'OpenLdap

LDAP, (Lightweight Directory Access Protocol), est le standard pour accéder à un annuaire. Un annuaire est une base de données qui contient des informations sur des individus, des machines, des groupes, ou toute autre catégorie imaginable.

Les annuaires sont largement utilisés pour stocker des données d'authentification (identifiant et mot de passe) ou pour obtenir des informations sur des personnes (adresse e-mail, numéro de téléphone, etc.) ou des objets (localisation, marque, modèle, etc.). Par exemple, toutes les applications de votre entreprise (site web, e-mails, comptes système des ordinateurs, etc.) peuvent utiliser ce service d'annuaire pour valider les identifiants de connexion.

2.7.2 Le fonctionnement de LDAP

Tout d'abord, un annuaire LDAP est une organisation hiérarchique d'entrées. Cette organisation constitue un arbre appelé DIT (Directory Information Tree) dont une des entrées est la racine.

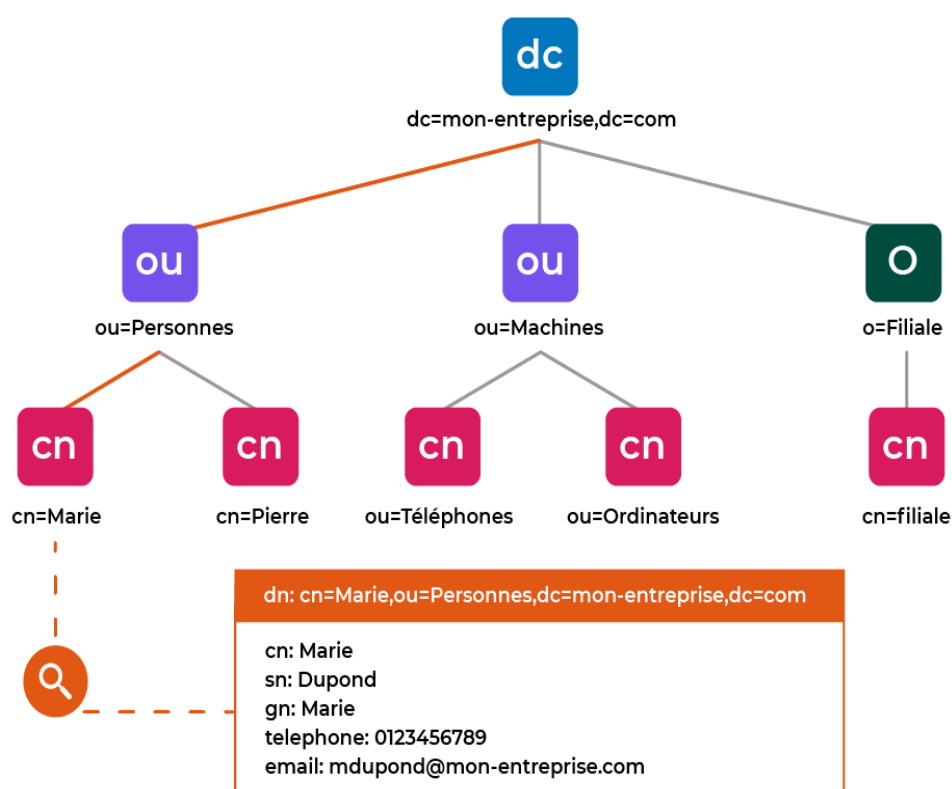


FIGURE 2.7.1 – Exemple d'un DIT de racine "dc=mon-entreprise,dc=com"

Chaque entrée peut contenir des attributs auxquels on assigne des valeurs. Chaque entrée appartient au moins à une classe d'objet qui définit les attributs de l'entrée.

Par exemple, la classe d'objet "Employés" pourrait définir qu'un "élément" appartenant à cette classe doit contenir les attributs obligatoires :

- nom de famille ;
- prénom.

et peut contenir les attributs facultatifs :

- e-mail ;
- téléphone ;
- date de naissance.

Chacun des attributs de cet élément aura une valeur. Par exemple, "nom de famille=Dupond".

De nombreux attributs et classes d'objets sont prédéfinis, mais il est possible de définir les vôtres si besoin. L'ensemble des classes d'objets et attributs utilisés est défini dans le schéma. Certains attributs sont particulièrement courants et intéressants à connaître :

Attributs	Fonction
dc (domain component)	la racine du DIT "dc=mon-entreprise,dc=com"
cn (common name)	le nom commun. c'est en général le prénom + le nom de famille
gn (given name)	le prénom
sn (surname)	le nom de famille
o (organization name)	le nom de l'entreprise
ou (organisational unit)	l'unité d'organisation.(commercial, comptabilité, etc.)

Un attribut particulier est le **dn (distinguished name)**, c'est-à-dire le nom distinct. C'est un attribut qui identifie de manière unique un élément dans le DIT. Il reprend les noms de tous les éléments depuis la racine jusqu'à l'élément, et indique ainsi un "chemin" unique pour trouver l'élément.

Par exemple, le dn de "Marie Dupond" qui travaille chez "mon-entreprise.com" pourrait être "**cn=Marie Dupond,ou=Personnes,dc=mon-entreprise,dc=com**". On appelle RDN, pour Relative Distinguished Name, la partie "finale" propre à Marie. Ici le RDN serait "cn=Marie Dupond". Lui ne garantit pas l'unicité dans le DIT.

Conclusion

Le déploiement d'un système d'authentification unique (SSO) devient de plus en plus crucial dans le contexte actuel des applications web, avec l'augmentation de leur nombre et du nombre d'utilisateurs. La gestion des comptes utilisateurs et des accès devient une tâche de plus en plus complexe. Chacune des solutions d'authentification présentées offre ses propres caractéristiques et avantages uniques. Étant donné les fonctionnalités impressionnantes de CAS et de LDAP, nous avons choisi de les associer pour créer notre système d'authentification SSO.

Chapitre 3

Mise en œuvre

Introduction

Dans ce chapitre dédié à la mise en œuvre et à la réalisation de notre projet, nous allons présenter les différentes étapes et les outils utilisés pour concrétiser notre vision. Nous commencerons par fournir un aperçu général de notre application, en mettant en évidence les choix technologiques et les fonctionnalités clés.

Le chapitre sur le CAS (Central Authentication Service) fournit une vue d'ensemble du CAS et de ses fonctionnalités. Le CAS est un serveur d'authentification central développé par l'Université de Yale. Il met en œuvre un serveur d'authentification sécurisé accessible via le web et offre des mesures de sécurité solides telles que la transmission de mots de passe chiffrée et la ré-authentification transparente. Le CAS permet une fonctionnalité de connexion unique (SSO) et prend en charge les configurations en plusieurs niveaux via des tickets de proxy.

3.1 Installation et configuration de CAS

3.1.1 Présentation du projet

Le projet consiste à intégrer CAS, une solution open source de single sign-on (SSO), avec un annuaire LDAP pour l'authentification et l'autorisation des utilisateurs. CAS agit en tant que serveur d'authentification central, permettant aux utilisateurs de se connecter une seule fois et d'accéder à plusieurs applications sans avoir à se ré-authentifier. LDAP sert de service d'annuaire, stockant les informations d'identification des utilisateurs et fournissant des informations sur les utilisateurs.

3.1.2 Prérequis

Avant d'installer et configurer CAS, On a besoin des éléments suivants :

JDK 17

CAS 7.0.0 requiert JDK 17 (Java Development Kit) pour fonctionner correctement. Assurons nous d'avoir installé JDK 17 sur notre système.

Pour configurer les variables d'environnement pour JDK 17, on suit les étapes suivantes :

- Ouvrons le panneau de configuration de notre système d'exploitation.
- Recherchons l'option "Variables d'environnement" et cliquons dessus.
- Dans la fenêtre des variables d'environnement, on recherche la variable "JAVA_HOME" ou créons-la si elle n'existe pas.
- Modifions la valeur de la variable "JAVA_HOME" et on spécifie le chemin d'installation de JDK 17.
- Ajoutons le chemin d'accès au répertoire "bin" de JDK 17 à la variable "Path".

Une fois les variables d'environnement configurées, nous pouvons vérifier si JDK 17 est correctement installé en ouvrant une nouvelle fenêtre de terminal et en exécutant la commande suivante :

```
java -version
```

Si la version affichée correspond à JDK 17, cela signifie que JDK 17 est correctement configuré.

Gradle 8.1.1

Pour la construction et la gestion des dépendances de CAS, nous avons besoin de Gradle 8.1.1. Assurons-nous d'avoir installé Gradle 8.1.1 sur notre système.

Pour configurer les variables d'environnement pour Gradle 8.1.1, suivez les étapes suivantes :

- Ouvrons le panneau de configuration de notre système d'exploitation.
- Recherchons l'option "Variables d'environnement" et cliquons dessus.
- Dans la fenêtre des variables d'environnement, on recherche la variable "GRADLE_HOME" ou créons-la si elle n'existe pas.
- Modifions la valeur de la variable "GRADLE_HOME" et spécifions le chemin d'installation de Gradle 8.1.1.
- Ajoutons le chemin d'accès au répertoire "bin" de Gradle 8.1.1 à la variable "Path".

Une fois les variables d'environnement configurées, nous pouvons vérifier si Gradle 8.1.1 est correctement installé en ouvrant une nouvelle fenêtre de terminal et en exécutant la commande suivante :

```
gradle -v
```

Si la version affichée correspond à Gradle 8.1.1, cela signifie que Gradle 8.1.1 est correctement configuré.

3.1.3 Installation de CAS

Une fois que nous avons vérifié les prérequis et que nous avons les éléments nécessaires, nous pouvons procéder à l'installation et à la configuration de CAS.

Voici les étapes à suivre :

Clonage du référentiel CAS

Nous commençons par cloner le référentiel CAS à l'aide de la commande suivante :

```
git clone https://github.com/apereo/cas-overlay-template.git
```

Cette commande crée un répertoire contenant les fichiers nécessaires pour l'installation de CAS.

Accès au Répertoire CAS

Une fois le référentiel cloné, nous devons accéder au répertoire CAS à l'aide de la commande suivante :

```
cd cas-overlay-template
```

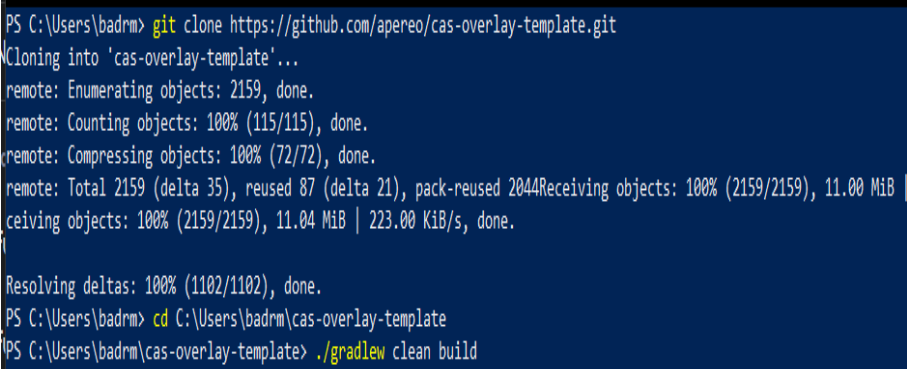
Cela nous place dans le répertoire où nous trouverons les fichiers de configuration et les scripts nécessaires.

Construction de CAS

Maintenant, nous pouvons procéder à la construction de CAS en exécutant la commande suivante :

```
./gradlew clean build
```

Cette commande télécharge les dépendances requises, compile les fichiers source et construit l'application CAS.



```
PS C:\Users\badrm> git clone https://github.com/apereo/cas-overlay-template.git
Cloning into 'cas-overlay-template'...
remote: Enumerating objects: 2159, done.
remote: Counting objects: 100% (115/115), done.
remote: Compressing objects: 100% (72/72), done.
remote: Total 2159 (delta 35), reused 87 (delta 21), pack-reused 2044Receiving objects: 100% (2159/2159), 11.00 MiB |
ceiving objects: 100% (2159/2159), 11.04 MiB | 223.00 KiB/s, done.
Resolving deltas: 100% (1102/1102), done.
PS C:\Users\badrm> cd C:\Users\badrm\cas-overlay-template
PS C:\Users\badrm\cas-overlay-template> ./gradlew clean build
```

```

PS C:\Users\badrm\cas-overlay-template> ./gradlew clean build
Configuration on demand is an incubating feature.

BUILD SUCCESSFUL in 21s
9 actionable tasks: 8 executed, 1 from cache
PS C:\Users\badrm\cas-overlay-template>

```

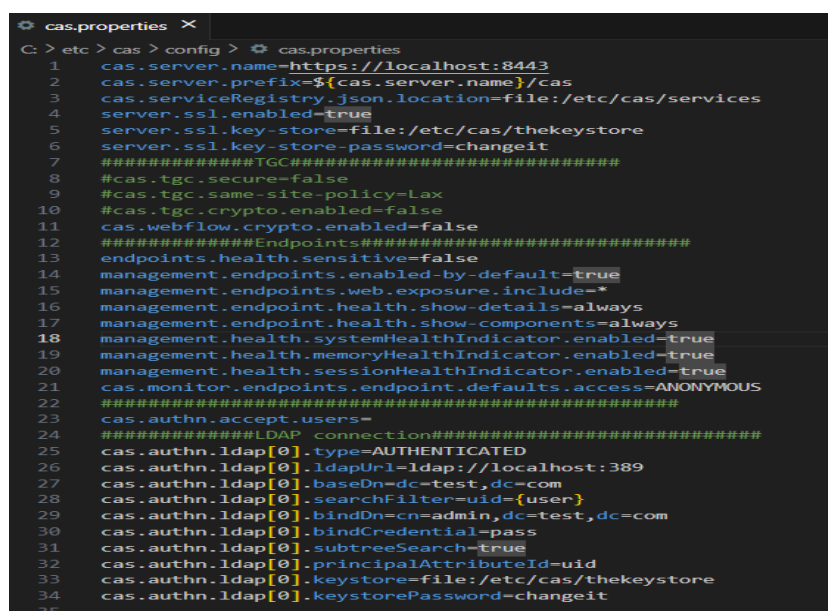
FIGURE 3.1.2 – Clonnage réussie.

3.1.4 Configuration de CAS

Une fois la construction terminée, nous devons configurer CAS en ajustant les fichiers de configuration selon nos besoins spécifiques. Les fichiers de configuration se trouvent dans le répertoire "cas/src/main/resources".

Voici quelques fichiers de configuration clés que nous pouvons modifier :

- **cas.properties** : Ce fichier contient les paramètres de configuration principaux de CAS, tels que l'URL du serveur LDAP, les informations sur la base de données, les stratégies d'authentification, etc.



```

cas.properties
C:\etc>cas>config>cas.properties
1 cas.server.name=https://localhost:8443
2 cas.server.prefix=${cas.server.name}/cas
3 cas.serviceRegistry.json.location=file:/etc/cas/services
4 server.ssl.enabled=true
5 server.ssl.key-store=file:/etc/cas/thekeystore
6 server.ssl.key-store-password=changeit
7 #####TGC#####
8 #cas.tgc.secure=false
9 #cas.tgc.same-site-policy=Lax
10 #cas.tgc.crypto.enabled=false
11 cas.webflow.crypto.enabled=false
12 #####Endpoints#####
13 endpoints.health.sensitive=false
14 management.endpoints.enabled-by-default=true
15 management.endpoints.web.exposure.include=*
16 management.endpoint.health.show-details=always
17 management.endpoint.health.show-components=always
18 management.health.systemHealthIndicator.enabled=true
19 management.health.memoryHealthIndicator.enabled=true
20 management.health.sessionHealthIndicator.enabled=true
21 cas.monitor.endpoints.endpoint.defaults.access=ANONYMOUS
22 #####
23 cas.authn.accept.users=
24 #####LDAP connection#####
25 cas.authn.ldap[0].type=AUTHENTICATED
26 cas.authn.ldap[0].ldapUrl=ldap://localhost:389
27 cas.authn.ldap[0].baseDn=dc=test,dc=com
28 cas.authn.ldap[0].searchFilter=uid={user}
29 cas.authn.ldap[0].bindDn=cn=admin,dc=test,dc=com
30 cas.authn.ldap[0].bindCredential=pass
31 cas.authn.ldap[0].subtreeSearch=true
32 cas.authn.ldap[0].principalAttributeId=uid
33 cas.authn.ldap[0].keystore=file:/etc/cas/thekeystore
34 cas.authn.ldap[0].keystorePassword=changeit
35

```

FIGURE 3.1.3 – Fichier cas.properties

- **log4j2.xml** : Ce fichier définit la configuration des journaux pour CAS.

```

1 <?xml version="1.0" encoding="UTF-8" ?>
2 <!--
3 All loggers are asynchronous because of log4j.component.properties in cas-server-core-logging-api.
4 Set -Dlog4j2.contextSelector=org.apache.logging.log4j.core.selector.BasicContextSelector or override log4j2.component.properties to turn off async
5 -->
6 <!-- Specify the refresh interval in seconds. -->
7 <Configuration monitorInterval="5" packages="org.apereo.cas.logging">
8   <Properties>
9     <Property name="baseDir">/var/log/</Property>
10    <Property name="cas.log.level">info</Property>
11    <Property name="spring.webflow.log.level">warn</Property>
12    <Property name="spring.security.log.level">info</Property>
13    <Property name="spring.cloud.log.level">warn</Property>
14    <Property name="spring.web.log.level">warn</Property>
15    <Property name="spring.boot.log.level">warn</Property>
16    <Property name="ldap.log.level">warn</Property>
17    <Property name="pac4j.log.level">warn</Property>
18    <Property name="opensaml.log.level">warn</Property>
19    <Property name="hazelcast.log.level">warn</Property>
20    <Property name="log.console.stacktraces">true</Property>
21    <Property name="log.file.stacktraces">false</Property>
22    <!-- .Dlog.stacktraceappender=null to disable stacktrace log -->
23    <Property name="log.stacktraceappender">casStackTraceFile</Property>
24    <Property name="log.include.location">false</Property>
25  </Properties>
26  <Appenders>
27    <Null name="null" />
28
29    <Console name="console" target="SYSTEM_OUT">
30      <PatternLayout pattern="%highlight{%d %p [%c] - %lt;%n}>%n" alwaysWriteExceptions="${sys:log.console.stacktraces}" />
31    </Console>
32
33    <RollingFile name="file" fileName="${baseDir}/cas.log" append="true"
34      filePattern="${baseDir}/cas-%d{yyyy-MM-dd-HH}.%i.log.gz"
35      immediateFlush="false">
36      <PatternLayout pattern="%highlight{%d %p [%c] - %lt;%n}>%n"
37        alwaysWriteExceptions="${sys:log.file.stacktraces}" />
38      <Policies>
39        <OnStartupTriggeringPolicy />
40        <SizeBasedTriggeringPolicy size="10 MB" />
41        <TimeBasedTriggeringPolicy />
42      </Policies>
43    </RollingFile>
44  </Appenders>
45
46  <Loggers>
47    <Logger name="org.apereo.cas" level="info" />
48    <Logger name="org.springframework" level="info" />
49    <Logger name="org.springframework.webflow" level="warn" />
50    <Logger name="org.springframework.security" level="info" />
51    <Logger name="org.springframework.cloud" level="warn" />
52    <Logger name="org.springframework.web" level="warn" />
53    <Logger name="org.springframework.boot" level="warn" />
54    <Logger name="org.springframework.ldap" level="warn" />
55    <Logger name="org.springframework.pac4j" level="warn" />
56    <Logger name="org.springframework.opensaml" level="warn" />
57    <Logger name="org.springframework.hazelcast" level="warn" />
58    <Logger name="org.springframework.log" level="info" />
59    <Logger name="org.springframework.log.console" level="info" />
60    <Logger name="org.springframework.log.file" level="info" />
61    <Logger name="org.springframework.log.stacktrace" level="info" />
62    <Logger name="org.springframework.log.location" level="info" />
63  </Loggers>
64 </Configuration>

```

FIGURE 3.1.4 – Extrait du fichier log4j2.xml

Après avoir apporté les modifications nécessaires aux fichiers de configuration, nous sommes prêts à exécuter CAS.

3.1.5 Exécution de CAS

Pour que le serveur s'exécute correctement, nous devons créer un fichier Keystore Cela peut être fait à l'aide de l'utilitaire 'keytool' du JDK :

```
keytool -genkey -keyalg RSA -alias thekeystore -keystore thekeystore -storepass
changeit -validity 360 -keysize 2048
```

ou via la commande suivante :

```
./gradlew createKeystore
```

```

PS C:\Users\badrm\cas-overlay-template> keytool -genkey -keyalg RSA -alias thekeystore -keystore thekeystore -storepass
changeit -validity 360 -keysize 2048
What is your first and last name?
[Unknown]: localhost
What is the name of your organizational unit?
[Unknown]: localhost
What is the name of your organization?
[Unknown]: localhost
What is the name of your City or Locality?
[Unknown]: rabat
What is the name of your State or Province?
[Unknown]: rabat
What is the two-letter country code for this unit?
[Unknown]: ma
Is CN=localhost, OU=localhost, O=localhost, L=ratat, ST=ratat, C=ma correct?
[no]: yes

Generating 2,048 bit RSA key pair and self-signed certificate (SHA256withRSA) with a validity of 360 days
for: CN=localhost, OU=localhost, O=localhost, L=ratat, ST=ratat, C=ma
PS C:\Users\badrm\cas-overlay-template>

```

FIGURE 3.1.5 – Création de fichier keystore

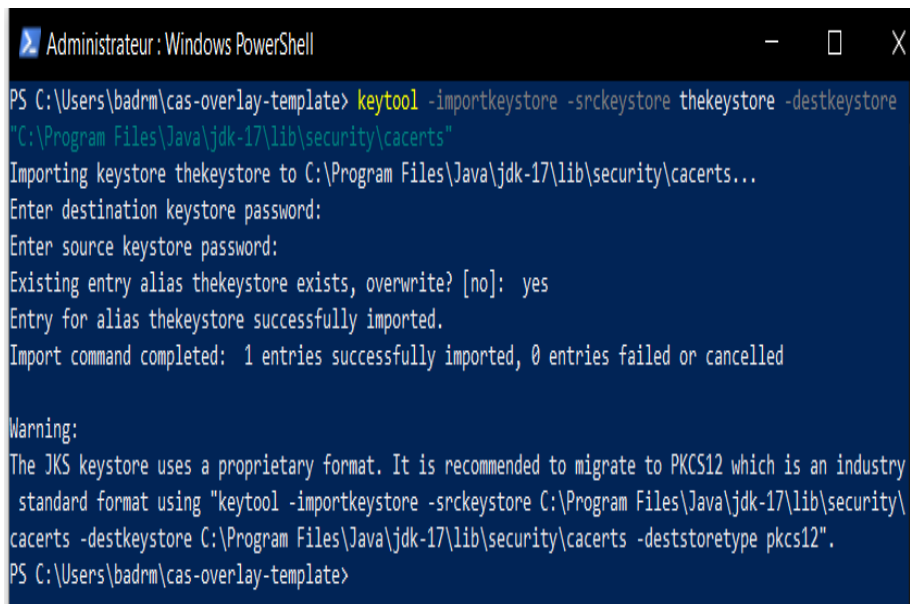
Pour que nous n'ayons pas d'erreur de négociation SSL, nous devons utiliser localhost comme valeur du prénom et du nom. Nous devrions également utiliser la même chose pour le nom de l'organisation et de l'unité.

De plus, nous devons importer le keystore dans le JDK avec la commande suivante :

```

keytool -importkeystore -srckeystore thekeystore -destkeystore
"C:\Program Files\Java\jdk-17\lib\security\cacerts"

```



```

Administrateur : Windows PowerShell
PS C:\Users\badrm\cas-overlay-template> keytool -importkeystore -srckeystore thekeystore -destkeystore
"C:\Program Files\Java\jdk-17\lib\security\cacerts"
Importing keystore thekeystore to C:\Program Files\Java\jdk-17\lib\security\cacerts...
Enter destination keystore password:
Enter source keystore password:
Existing entry alias thekeystore exists, overwrite? [no]: yes
Entry for alias thekeystore successfully imported.
Import command completed: 1 entries successfully imported, 0 entries failed or cancelled

Warning:
The JKS keystore uses a proprietary format. It is recommended to migrate to PKCS12 which is an industry
standard format using "keytool -importkeystore -srckeystore C:\Program Files\Java\jdk-17\lib\security\
cacerts -destkeystore C:\Program Files\Java\jdk-17\lib\security\cacerts -deststoretype pkcs12".
PS C:\Users\badrm\cas-overlay-template>

```

FIGURE 3.1.6 – Importer le keystore dans le JDK cacerts

NB : Le mot de passe source et de destination keystore est **changeit**.
On peut Maintenant exécuter CAS utilisons la commande suivante :

```
./gradlew bootRun
```

[illegible]

FIGURE 3.1.7 – Execution du serveur

```
> [org.apereo.cas.web.CasWebApplication] - <Started CasWebApplication in 18.29 seconds (process running for 11.95)>  
[org.apereo.cas.services.AbstractServiceManager] - <(loaded {} service(s) from {}><br>[org.apereo.cas.web.CasWebApplicationHead] - <>  
[org.apereo.cas.web.CasWebApplicationHead] - <-  
  
┌───┐ ┌───┐      /   \    \| | \| |  
│ _ │ │ _ │      /     \   \| | \| |  
│_||_|_|||       /       \__\ || ||  
│_||_|_|||       /         \| || ||  
└───┘ └───┘      /           \_____\
```

```
[org.apereo.cas.web.CasWebApplicationHead] -->  
[org.apereo.cas.web.CasWebApplicationHead] - <Ready to process requests @ {2022-07-06T12:00:06.755Z}>  
(-----) 8% EXECUTING [2m 7s] io.cas.configuration.AsConfigurationPropertiesValidator - <Validating CAS property sources and configuration for active profiles []standalone[]>. Please wait...  
[org.apereo.cas.configuration.AsConfigurationPropertiesValidator] - <Validated CAS property sources and configuration successfully.>
```

FIGURE 3.1.8 – Serveur est "Ready"

Cela démarre l'application CAS sur notre machine locale.

Une fois que CAS est en cours d'exécution, nous pouvons accéder à l'interface utilisateur en ouvrant un navigateur et en accédant à l'URL suivante :
https ://localhost :8443/login/cas

Cela affiche la page d'accueil de CAS où nous pouvons commencer à configurer et tester l'authentification unique (SSO).

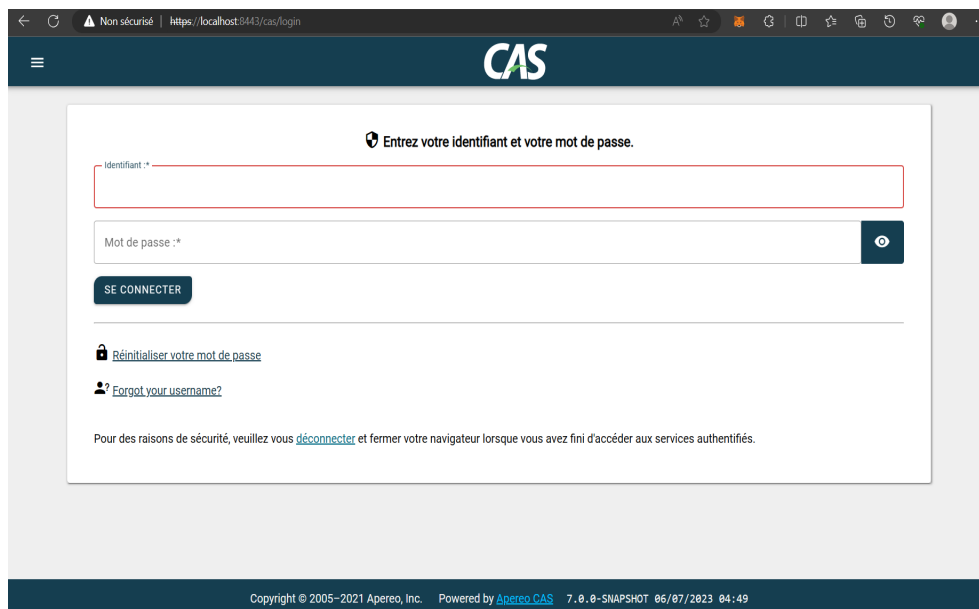


FIGURE 3.1.9 – Interface d'authentification de CAS

Nous avons maintenant installé et configuré CAS avec succès.

3.2 Intégration de l'authentification OpenLDAP

3.2.1 Installation et configuration d'OpenLDAP

Installez OpenLDAP Server sur Ubuntu

OpenLDAP est un des annuaires les plus répandus. Pour l'installer, nous devons installer le paquet slapd . Installons également le paquet ldap-utils qui contient les utilitaires clients pour pouvoir interroger ou modifier notre annuaire.

```
$ sudo apt-get install slapd ldap-utils
```

Pendant l'installation, nous avons invité à définir le mot de passe administrateur LDAP, à fournir le mot de passe souhaité, puis à appuyer sur <OK>

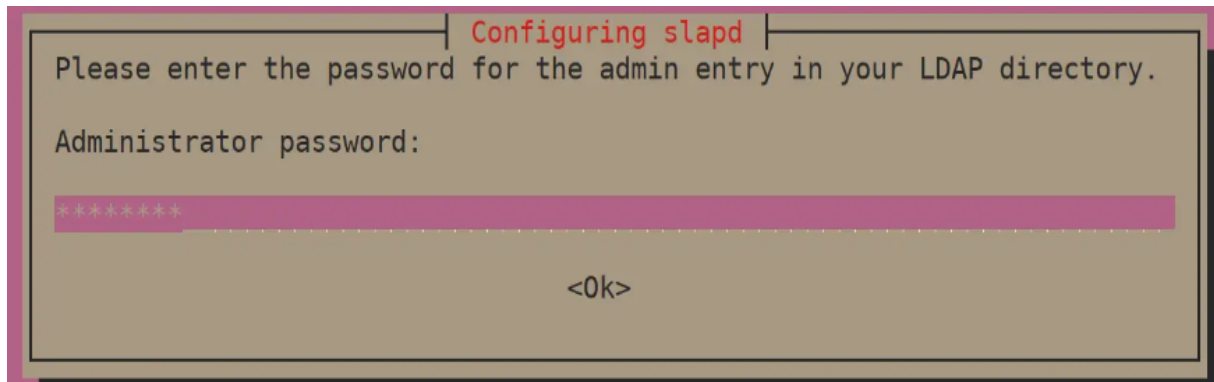


FIGURE 3.2.1 – le mot de passe administrateur

Confirmons le mot de passe et poursuivre l'installation en sélectionnant <ok>

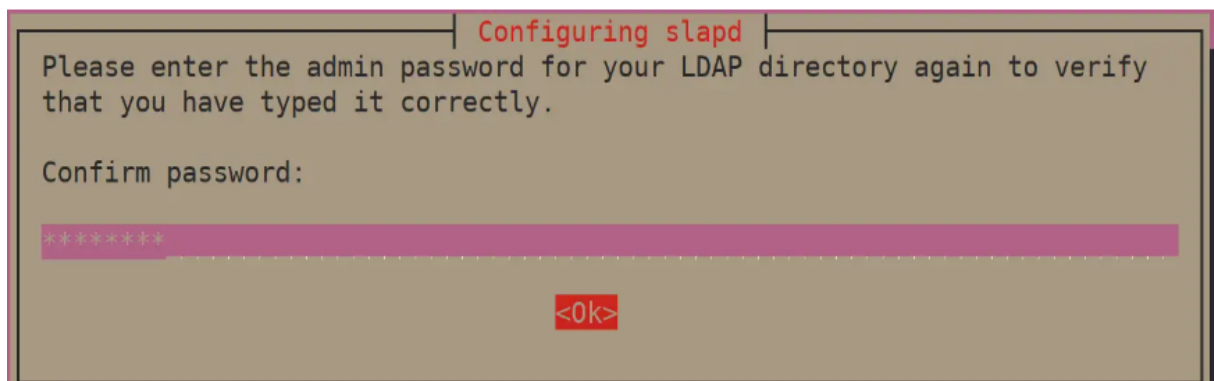


FIGURE 3.2.2 – Confirmez le mot de passe

nous pouvons confirmer que notre installation a réussi en utilisant la commande **slapcat** pour générer le contenu de la base de données SLAPD

```
$ sudo slapcat
dn : dc=example,dc=com
objectClass : top
objectClass : dcObject
objectClass : organization
o : example.com
dc : example
structuralObjectClass : organization
entryUUID : 3096cde2-64b5-103c-836e-1d0b0995a781
creatorsName : cn=admin,dc=example,dc=com
createTimestamp : 20220510135944Z
entryCSN : 20220510135944.468673Z000000000000000000
modifiersName : cn=admin,dc=example,dc=com
modifyTimestamp : 20220510135944Z
```

Ajoutons un dn de base pour les utilisateurs et les groupes

L'étape suivante consiste à ajouter un nom unique de base pour les utilisateurs et les groupes. Créons un fichier nommé basedn.ldif avec le contenu ci-dessous :

```
$ nano basedn.ldif
dn : ou=people,dc=test,dc=com
objectClass : organizationalUnit
ou : people

dn : ou=groups,dc=test,dc=com
objectClass : organizationalUnit
ou : groups
```

J'ai remplacé "example" et "com" par mes composants de domaine corrects. "test" "com"

Maintenant, j'ai ajouté le fichier ldif en exécutant la commande :

```
$ ldapadd -x -D cn=admin,dc=test,dc=com -W -f basedn.ldif
Enter LDAP Password :
adding new entry "ou=people,dc=test,dc=com"
adding new entry "ou=groups,dc=test,dc=com"
```

Ajouter des comptes d'utilisateurs et des groupes

Générer un mot de passe pour le compte d'utilisateur à ajouter avec la commande **slappasswd**

```
$ sudo slappasswd
```

```
New password :
```

```
Re-enter new password :
```

```
SSHAZn4/E5f+Ork7WZF/alrpMuHHGufC3x0k
```

J'ai créé un fichier ldif pour ajouter des utilisateurs.

```
$ nano ldapusers.ldif
```

```
dn : uid=myuser,ou=people,dc=test,dc=com
```

```
objectClass : inetOrgPerson
```

```
objectClass : posixAccount
```

```
objectClass : shadowAccount
```

```
cn : myuser
```

```
sn : myuser
```

```
userPassword : SSHAZn4/E5f+Ork7WZF/alrpMuHHGufC3x0k
```

```
loginShell : /bin/bash
```

```
uidNumber : 2000
```

```
gidNumber : 2000
```

```
homeDirectory : /home/myuser
```

Maintenant, j'ai ajouté le fichier ldif en exécutant la commande :

```
$ ldapadd -x -D cn=admin,dc=test,dc=com -W -f ldapusers.ldif
```

```
Enter LDAP Password :
```

```
adding new entry "uid=myuser,ou=people,dc=test,dc=com"
```

la même chose pour le groupe

```
$ nano ldapgroups.ldif
```

```
dn : cn=myuser,ou=groups,dc=test,dc=com
```

```
objectClass : posixGroup
```

```
cn : computingforgeeks
```

```
gidNumber : 2000
```

```
memberUid : myuser
```

Ajoute de groupe

```
$ ldapadd -x -D cn=admin,dc=test,dc=com -W -f ldapgroups.ldif
```

```
Enter LDAP Password :
```

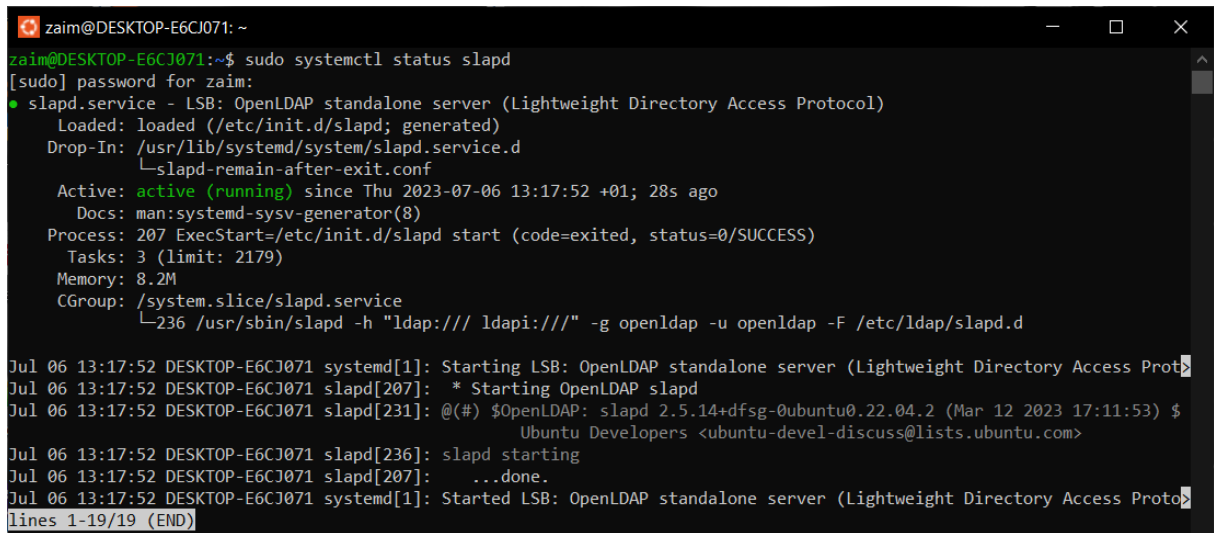
```
adding new entry "cn=myuser,ou=groups,dc=test,dc=com"
```

Une fois l'installation terminée, on peut démarrer le serveur LDAP en utilisant la commande suivante :

```
$ sudo systemctl start slapd
```

Nous pouvons vérifier que le serveur LDAP est en cours d'exécution en exécutant la commande suivante :

```
$ sudo systemctl status slapd
```



```
zaim@DESKTOP-E6CJ071: ~  
zaim@DESKTOP-E6CJ071:~$ sudo systemctl status slapd  
[sudo] password for zaim:  
● slapd.service - LSB: OpenLDAP standalone server (Lightweight Directory Access Protocol)  
   Loaded: loaded (/etc/init.d/slapd; generated)  
   Drop-In: /usr/lib/systemd/system/slapd.service.d  
            └─slapd-remain-after-exit.conf  
   Active: active (running) since Thu 2023-07-06 13:17:52 +01; 28s ago  
     Docs: man:systemd-sysv-generator(8)  
  Process: 207 ExecStart=/etc/init.d/slapd start (code=exited, status=0/SUCCESS)  
    Tasks: 3 (limit: 2179)  
   Memory: 8.2M  
   CGroup: /system.slice/slapd.service  
           └─236 /usr/sbin/slapd -h "ldap:/// ldapi:///" -g openldap -u openldap -F /etc/ldap/slapd.d  
  
Jul 06 13:17:52 DESKTOP-E6CJ071 systemd[1]: Starting LSB: OpenLDAP standalone server (Lightweight Directory Access Prot  
Jul 06 13:17:52 DESKTOP-E6CJ071 slapd[207]: * Starting OpenLDAP slapd  
Jul 06 13:17:52 DESKTOP-E6CJ071 slapd[231]: @(#) $OpenLDAP: slapd 2.5.14+dfsg-0ubuntu0.22.04.2 (Mar 12 2023 17:11:53) $  
           Ubuntu Developers <ubuntu-devel-discuss@lists.ubuntu.com>  
Jul 06 13:17:52 DESKTOP-E6CJ071 slapd[236]: slapd starting  
Jul 06 13:17:52 DESKTOP-E6CJ071 slapd[207]: ...done.  
Jul 06 13:17:52 DESKTOP-E6CJ071 systemd[1]: Started LSB: OpenLDAP standalone server (Lightweight Directory Access Proto  
lines 1-19/19 (END)
```

FIGURE 3.2.3 – Executer Ldap dans ubuntu

3.2.2 Liaison entre CAS et OpenLDAP

Pour intégrer CAS avec LDAP pour l'authentification des utilisateurs, nous devons effectuer les étapes suivantes :

- Nous devons ajouter la dépendance LDAP à notre projet CAS. Pour ce faire, nous devons modifier le fichier **build.gradle** de notre projet. Ouvrez le fichier build.gradle et ajoutez la ligne suivante à l'intérieur du bloc dependencies

```
implementation org.apereo.cas:cas-server-support-ldap:${project.'cas.version'}
```

En ajoutant la dépendance LDAP, nous permettons à CAS d'utiliser les bibliothèques nécessaires pour communiquer avec les serveurs LDAP et effectuer des opérations d'authentification des utilisateurs et autres opérations liées à LDAP.

- Assurons-nous d'avoir configuré et exécuté un serveur LDAP fonctionnel. Nous devons connaître les informations de connexion à notre serveur LDAP, telles que l'URL, le nom d'utilisateur et le mot de passe.

- Modifions le fichier de configuration de CAS pour ajouter les paramètres de connexion à LDAP. Ouvrons le fichier ‘cas.properties’ situé dans le répertoire de configuration de CAS.
- Recherchons la section ‘LDAP Authentication Settings’ et configurons les propriétés suivantes en fonction de notre serveur LDAP :

```
cas.authn.ldap[0].type=AUTHENTICATED
cas.authn.ldap[0].ldapUrl=ldap://localhost:389
cas.authn.ldap[0].baseDn=dc=test,dc=com
cas.authn.ldap[0].userFilter=(&(objectClass=person)(uid={user}))
cas.authn.ldap[0].bindDn=cn=admin,dc=test,dc=com
cas.authn.ldap[0].bindCredential=pass
cas.authn.ldap[0].principalAttributeId=uid
cas.authn.ldap[0].searchFilter=(&(objectClass=person)(uid={user}))
```

- Enregistrons les modifications apportées au fichier ‘cas.properties’.
- Redémarrons le serveur CAS pour appliquer les modifications de configuration.
- Testons l’authentification avec LDAP en utilisant un utilisateur existant dans notre serveur LDAP. Nous devrions pouvoir nous connecter avec les informations d’identification LDAP et accéder aux fonctionnalités protégées de notre application.

En suivant ces étapes, nous avons configuré CAS pour l’authentification des utilisateurs via LDAP.

3.3 Réalisation du client CAS avec une application Spring Boot

3.3.1 Outils de développement de l’application

Dans le cadre de notre projet de développement d’une application Spring Boot avec CAS, nous avons utilisé les outils suivants :

Eclipse IDE

Eclipse IDE est un environnement de développement intégré populaire pour Java. Nous avons utilisé Eclipse IDE pour développer notre application Spring Boot. Voici quelques fonctionnalités offertes par Eclipse IDE qui nous ont été utiles :

- **Édition de code** : Eclipse IDE offre des fonctionnalités d’édition de code avancées, telles que la coloration syntaxique, l’auto-complétion, la navigation dans le code, la refactorisation, etc. Ces fonctionnalités nous ont aidés à écrire du code de manière efficace et sans erreurs.

- **Débogage** : Eclipse IDE propose un débogueur intégré qui nous a permis de mettre en place des points d'arrêt, d'exécuter notre application en mode débogage et d'inspecter les variables et les étapes d'exécution. Cela nous a aidés à identifier et à résoudre les problèmes lors du développement de notre application.

Grâce à Eclipse IDE, nous avons pu développer notre application Spring Boot de manière efficace, en profitant des fonctionnalités avancées et de l'intégration avec Spring Boot.

Application Spring Boot

Nous avons utilisé Spring Boot comme framework de développement pour notre application. Spring Boot simplifie le développement d'applications Java en fournissant des conventions et des fonctionnalités par défaut, ce qui nous a permis de nous concentrer sur la logique métier plutôt que sur la configuration technique.

Les avantages de l'utilisation de Spring Boot pour notre application incluent :

- **Configuration automatique** : Spring Boot offre une configuration automatique basée sur les conventions. Cela signifie que nous n'avons pas besoin de configurer manuellement de nombreux aspects de notre application, tels que la configuration de la base de données, la configuration du serveur web, etc. Spring Boot détecte automatiquement les dépendances et les configure en fonction des meilleures pratiques.
- **Starter Packs** : Spring Boot propose des starter packs (pilotes de démarrage) qui incluent toutes les dépendances nécessaires pour développer des fonctionnalités spécifiques. Par exemple, nous avons utilisé le starter pack 'spring-boot-starter-web' pour développer des fonctionnalités web dans notre application. Cela nous a permis de démarrer rapidement avec les fonctionnalités requises sans avoir à gérer manuellement les dépendances.

En résumé, nous avons utilisé Eclipse IDE comme environnement de développement et Spring Boot comme framework pour développer notre application. Ces outils nous ont fourni des fonctionnalités avancées, une productivité accrue et une facilité de déploiement, ce qui a contribué au succès de notre projet.

3.3.2 Configuration du client CAS

Pour configurer notre application Spring Boot en tant que client CAS, nous devons effectuer les étapes suivantes :

- Ajout des dépendances nécessaires à notre fichier pom.xml , selon notre gestionnaire de dépendances. Nous devons nous assurer d'inclure la dépendance CAS client pour Spring Boot.

```

<dependencies>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
  </dependency>
  <dependency>
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
    <optional>true</optional>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-test</artifactId>
    <scope>test</scope>
  </dependency>
  <dependency>
    <groupId>org.jasig.cas.client</groupId>
    <artifactId>cas-client-core</artifactId>
    <version>3.2.0</version>
    <exclusions>
      <exclusion>
        <groupId>javax.servlet</groupId>
        <artifactId>servlet-api</artifactId>
      </exclusion>
    </exclusions>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-autoconfigure</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-configuration-processor</artifactId>
    <optional>true</optional>
  </dependency>
</dependencies>

```

FIGURE 3.3.1 – Extrait de fichier pom.xml

- Dans notre fichier de configuration (application.yml), nous devons spécifier les propriétés suivantes :

```

spring:
  main:
    allow-bean-definition-overriding: true
server:
  port: 8888
  tomcat:
    uri-encoding: UTF-8
  servlet:
    context-path: /cas-client
cas:
  cas-server-login-url: https://localhost:8443/cas/login
  cas-server-url-prefix: https://localhost:8443/cas
  cas-server-logout-url: https://localhost:8443/cas/logout
  service-url: http://localhost:8888

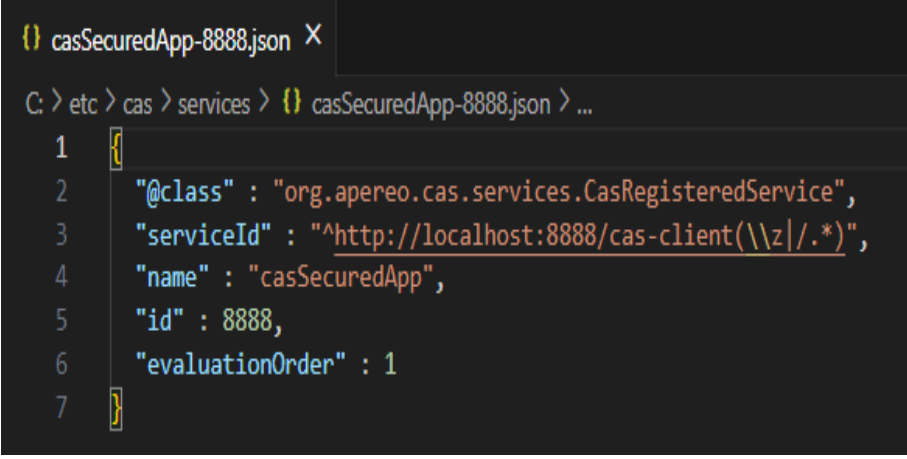
```

FIGURE 3.3.2 – Extrait de fichier application.yml

- **spring.main.allow-bean-definition-overriding** : Cette propriété permet la substitution de définitions de beans. Dans le cas où nous avons besoin de définir des beans spécifiques pour le client CAS, cette propriété doit être

définie sur `true`.

- **server.port** : Nous devons spécifier le port sur lequel notre application Spring Boot sera exécutée. Dans cet exemple, le port est configuré sur 8888.
 - **server.tomcat.uri-encoding** : Cette propriété définit l'encodage URI à utiliser. Ici, nous utilisons UTF-8.
 - **server.servlet.context-path** : Nous devons spécifier le chemin de contexte de notre application. Dans cet exemple, le chemin de contexte est configuré sur `/cas-client`.
 - **cas.cas-server-login-url** : Nous devons spécifier l'URL de connexion du serveur CAS. Dans cet exemple, l'URL est `https://localhost:8443/cas/login`.
 - **cas.cas-server-url-prefix** : Nous devons spécifier le préfixe d'URL du serveur CAS. Dans cet exemple, le préfixe d'URL est `https://localhost:8443/cas`.
 - **cas.cas-server-logout-url** : Nous devons spécifier l'URL de déconnexion du serveur CAS. Dans cet exemple, l'URL est `https://localhost:8443/cas/logout`.
 - **cas.service-url** : Nous devons spécifier l'URL du service de notre application. Dans cet exemple, l'URL est `http://localhost:8888`.
- Enregistrement de l'application en tant que fichier JSON avec le nom `casSecuredApp-8888.json` dans les services du serveur CAS pour l'autorisation.



```
{
  "@class" : "org.apereo.cas.services.CasRegisteredService",
  "serviceId" : "^http://localhost:8888/cas-client(\\z|/.*)",
  "name" : "casSecuredApp",
  "id" : 8888,
  "evaluationOrder" : 1
}
```

FIGURE 3.3.3 – Code JSON dans les services de CAS

Une fois ces étapes effectuées, notre application Spring Boot sera configurée en tant que client CAS et sera capable de s'authentifier auprès du serveur CAS.

3.3.3 Utilisation du client CAS

Une fois que nous avons configuré notre application Spring Boot en tant que client CAS, nous pouvons commencer à l'utiliser pour l'authentification centralisée. Voici les étapes à suivre pour utiliser le client CAS :

- Accès à l'application : Ouvrons notre navigateur web et accédons à l'URL de notre application Spring Boot configurée avec CAS. notre application est accessible à l'adresse `http://localhost:8888/cas-client`, nous saisissons cette URL dans la barre d'adresse du navigateur.
- Redirection vers le serveur CAS : Lorsque nous accédons à l'application, nous serons automatiquement redirigés vers l'URL de connexion du serveur CAS configurée dans notre application. Le serveur CAS nous présentera alors une page de connexion où nous devrons fournir nos identifiants d'authentification.

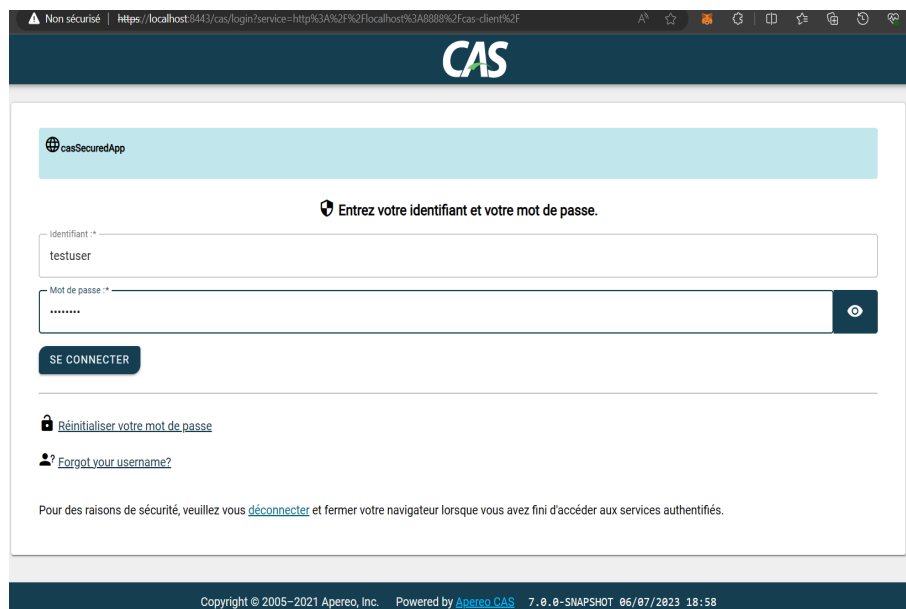


FIGURE 3.3.4 – Interface d'authentification de CAS client

- Authentification CAS : Nous saisissons nos identifiants (nom d'utilisateur et mot de passe) sur la page de connexion du serveur CAS. Une fois les identifiants soumis, le serveur CAS les vérifie par rapport à son annuaire d'authentification (dans notre cas, LDAP). Si les identifiants sont valides, nous serons authentifiés avec succès.
- Retour à l'application : Après une authentification réussie, le serveur CAS génère un ticket de service (ST) qui est inclus dans l'URL de redirection vers notre application Spring Boot. Notre application reçoit ce ticket de service et le valide auprès du serveur CAS pour confirmer l'authentification.

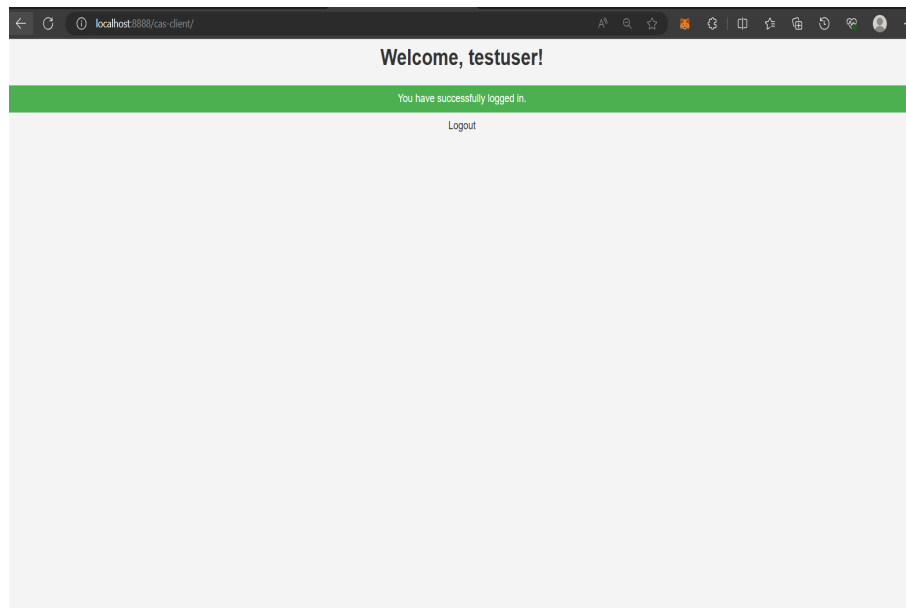


FIGURE 3.3.5 – La Page du CAS apres l’authentification

- Accès autorisé : Une fois que le ticket de service est validé avec succès, notre application Spring Boot considère l’utilisateur comme authentifié et lui donne accès aux ressources protégées. Nous pouvons alors naviguer sur les différentes pages de l’application sans avoir à nous reconnecter à chaque fois.

Il est important de noter que le processus d’authentification CAS n’est pas limité à une seule session. Une fois que nous sommes authentifiés, nous pouvons accéder à d’autres applications configurées avec le même serveur CAS sans avoir à nous reconnecter.

Conclusion

Dans ce chapitre, nous avons élaboré les différentes étapes suivi afin de mettre en œuvre notre projet, nous avons procéder en premier, par présenter notre projet. Par la suite, nous avons expliqué les diverses phases de l’installation et configuration du serveur CAS et l’annuaire LDAP. De plus, nous avons défini le service web sur lequel l’interface client fonctionne.

Conclusion générale

Pour conclure, ce stage m'a offert une expérience précieuse dans le domaine de la sécurité et de l'authentification, en particulier en ce qui concerne la mise en place d'un système d'authentification unique avec CAS. J'ai pu acquérir des connaissances pratiques en configurant le DNS, l'annuaire LDAP et le serveur CAS, et en surmontant les défis liés à l'interdépendance des versions des outils.

Ce projet m'a permis de comprendre l'importance du Single-Sign-On (SSO) pour offrir une meilleure expérience utilisateur et une gestion plus efficace des mots de passe. J'ai pu constater les avantages du SSO en termes de centralisation des systèmes d'authentification et de sécurisation des comptes à différents niveaux.

Malgré les difficultés rencontrées, telles que le manque de documentation sur certains aspects du serveur CAS, j'ai pu développer mes compétences en résolution de problèmes et en recherche de solutions. J'ai également renforcé ma compréhension de l'architecture des systèmes d'authentification et des bonnes pratiques de sécurité.

Ce stage m'a non seulement permis d'appliquer mes connaissances théoriques acquises au cours de mes études, mais aussi de découvrir de nouvelles technologies et de me familiariser avec des outils tels que Eclipse IDE et Spring Boot, Ubuntu et OpenLdap. Ces compétences techniques acquises seront précieuses pour ma future carrière dans le domaine de l'informatique.

En somme, ce stage a été une expérience enrichissante qui m'a permis d'approfondir mes connaissances, de développer de nouvelles compétences et de me familiariser avec des technologies et des méthodologies de développement avancées. Je suis reconnaissant d'avoir eu cette opportunité qui a contribué à ma croissance professionnelle et personnelle.

Bibliographie

https://fr.wikipedia.org/wiki/Authentification_unique

<https://www.baeldung.com/spring-security-cas-sso>

<https://www.apereo.org/projects/cas>

https://en.wikipedia.org/wiki/Central_Authentication_Service

<https://groups.google.com/a/apereo.org/g/cas-user>

https://www.cert-ist.com/pub/files/sso_part1_fr.pdf

<https://openclassrooms.com/fr/courses/1733551-gerez-votre-serveur-linux-et-ses-services/5236036-installez-un-annuaire-ldap>

<https://computingforgeeks.com/install-and-configure-openldap-server-ubuntu/>

https://www.esup-portail.org/consortium/espace/SSO_1B/cas/jres/cas-jres2003-article.pdf

<https://aldian.developpez.com/tutoriels/javaee/authentication-centralisee-sso-cas/>

<https://www.oracle.com/fr/security/qu-est-ce-qu-un-sso.html>

<https://www.ibm.com/docs/en/rpa/23.0?topic=ldap-installing-configuring-openldap>

<https://medium.com/swlh/install-cas-server-with-db-authentication-8ff52234f52>

<http://www-igm.univ-mlv.fr/~dr/XPOSE2006/CLERET/objectifs.html>