Sem vložte zadanie Vašej práce.

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra počítačových systémů

Bakalárska práca

Nasadenie SSO riešenia

Martin Gajdoš

Vedúci práce: Ing. Jiří Stegura

12. mája 2017

Poďakovanie

V prvom rade by som chcel poďakovať mojim rodičom a súrodencom. Bez Vás by som teraz nebol tam, kde som. Špeciálne poďakovanie patrí vedúcemu mojej práce, Jirkovi. Ďakujem za trpezlivosť a všetko, čo si ma naučil. Veľmi si to vážim.

Prehlásenie

Prehlasujem, že som predloženú prácu vypracoval(a) samostatne a že som uviedol(uviedla) všetky informačné zdroje v súlade s Metodickým pokynom o etickej príprave vysokoškolských záverečných prác.

Beriem na vedomie, že sa na moju prácu vzťahujú práva a povinnosti vyplývajúce zo zákona č. 121/2000 Sb., autorského zákona, v znení neskorších predpisov, a skutočnosť, že České vysoké učení technické v Praze má právo na uzavrenie licenčnej zmluvy o použití tejto práce ako školského diela podľa § 60 odst. 1 autorského zákona.

V Prahe 12. mája 2017 .....

Odkaz na túto prácu

Gajdoš, Martin. *Nasadenie SSO riešenia*. Bakalárska práca. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2017.

Abstrakt

Hlavným cieľom mojej práce je zaintegrovať single sign-on riešenie do existujúcej infraštruktúry firmy Ataccama Software, s.r.o. Aktuálne toto prostredie pozostáva z webových aplikácií, kde do každej sa musia užívatelia prihlasovať samostatne. To ale nie je ideálna situácia z pohľadu zákazníkov, ako aj zamestnancov.

Tento problém by malo vyriešiť nasadenie technológie SSO, čo znamená preniesť zodpovednosť za prihlasovanie a overovanie identity z jednotlivých stránok na jedinú webovú aplikáciu, ktorá je na to určená. SSO taktiež zabezpečí, aby sa užívateľ, ktorý je prihlásený do jednej webovej aplikácie, automaticky prihlásil do ďalších aplikácií bez toho, aby musel znovu zadávať svoje prihlasovacie údaje. Toto by malo zaistiť centralizáciu administrácie, zvýšiť bezpečnosť a dosiahnuť homogenitu prostredia. Je zrejmé, že táto úloha nie je úplne triviálna.

V práci sa budem zaoberať analýzou prostredia firmy, aby som určil aplikácie vhodné na zaintegrovanie do SSO systému a s tým súvisiace zmeny v infraštruktúre. Na základe tejto analýzy vyberiem vhodnú SSO implementáciu a navrhnem správu externých užívateľov pomocou LDAPu. Nakoniec nasadím a nakonfigurujem vybrané SSO riešenie a otestujem ho.

Klíčová slova SSO, autentifikácia, LDAP, webová bezpečnosť, CAS

Abstract

The main goal of my thesis is to integrate a single sign-on solution into the existing infrastructure of Ataccama Corporation. Currently, this environment consists of web applications, to which the users login separately, although using the same credentials. This situation is not ideal for customers, nor for employees.

The solution to this problem is an SSO solution deployment. This means transferring logon management and authentication from the web applications to a dedicated one. The SSO application ensures, that once the user is logged on to one application, they will be logged on to the other applications upon accessing them, without the need of reentering credentials. The result of this change will be centralization of administration, better security and homogeneous environment. It is obvious, that this is not a trivial task.

In this work, I will discuss the analysis of the company environment, so I can select applications to integrate with the SSO system and describe related changes in the infrastructure. Based on this analysis, I will select the most suitable SSO implementation and design a LDAP management system of external users. Then, I will deploy, configure and test the selected SSO solution.

Keywords SSO, authentication, LDAP, web security, CAS

Obsah

Zoznam obrázkov

Úvod

Už je to rok, čo pracujem pre firmu Ataccama software s.r.o.[1], ktorá sa zaoberá problematikou veľkých dát (anglicky *big data*[2]). Predávame softvérové nástroje na skvalitnenie, kontrolu, správu a analýzu veľkých dát. Toto odvetvie informatiky je relatívne nové a začína sa tešiť čoraz väčšiemu záujmu, či už zo strany širokej verejnosti, alebo zo strany firiem [2]. Dôsledkom týchto priaznivých okolností sa naša firma rýchlo rozrastá a máme stále viac zákazníkov.

Neustále sa snažíme zlepšovať užívateľskú skúsenosť s našimi produktami. Preto poskytujeme množstvo dokumentácie a podporu pre našich zákazíkov. Keďže však na tieto účely používame softvér tretích strán, nie je toto prostredie jednoliate a úplne intuitívne. Takisto kvôli rýchlemu rastu firmy existujúca infraštruktúra vyžaduje čoraz viac správy, a preto sú nutné jej úpravy.

Z týchto dôvodov sme sa rozhodli toto prostredie zjednotiť a zjednodušiť ako to len pôjde, aby bola užívateľská skúsenosť s našou podporou čo najlepšia. Témou mojej bakalárskej práce je preto nasadenie Single Sign-on (ďalej len „SSO“) riešenia v tomto prostredí a s ním súvisiace úpravy infraštruktúry. Toto zabezpečí jednoduchšie, pohodlnejšie a plynulejšie využívanie našich stránok tak zákazníkmi, ako aj zamestnancami. Malo by to tiež priniesť zjednodušenie správy daného prostredia a efektívnejšiu prácu v ňom.

[1] Viac o spoločnosti na: `https://ataccama.com/company/`

[2] *Veľké dáta je pojem, ktorý popisuje obrovské objemy dát, štrukturovaných a neštrukturovaných, ktoré každodenne zaplavujú biznis. Avšak nie je podstatný objem dát. Záleží na tom, čo organizácie robia s dátami. Veľké dáta sa dajú analyzovať pre náhľad, ktorý vedie k lepším rozhodnutiam a strategickým obchodným ťahom.* [1]

Kapitola 1

Analýza infraštruktúry firmy

1.1 Čo je SSO

Predtým, než začnem so samotnou analýzou, bolo by vhodné aby som definoval ústredný bod tejto práce – pojem SSO. Táto definícia a vysvetlenie všeobecných princípov jeho fungovania mi umožní, aby som správne zanalyzoval prostredie firmy a naplno využil potenciál, ktorý SSO prináša. Taktiež mi pomôže správne zvoliť konkrétnu implementáciu SSO v závislosti na výsledku tejto analýzy. Zároveň by som tu chcel vysvetliť aj niekoľko termínov, s ktorými SSO úzko súvisí, a ktoré budem používať v priebehu celej práce.

Webová aplikácia V tejto práci aj skrátene *aplikácia*. Je *počítačový program, ktorý využíva webový prehliadač a webové technológie na vykonávanie úloh prostredníctvom internetu* [3]. Rozdiel oproti webovej stránke je veľmi subjektívny. Princíp je však ten, že webová stránka má skôr informatívny charakter a webová aplikácia je interaktívna a jej obsah závisí na užívateľovi. Napríklad *Facebook*[3] a *Gmail*[4] sú typické webové aplikácie.

Webová služba Skrátene *služba*. Funkcionalita, ktorá je ponúkaná nejakým počítačovým programom iným počítačovým programom prostredníctvom internetu. Umožňuje automatizovanú výmenu informácií.

HTTP cookie Skrátene *cookies*, je krátky textový súbor, ktorý webový server posiela webovému prehliadaču na relatívne dlhodobé uloženie. Následne, pri dalšej návšteve daného webu, prehliadač tieto *cookies* odošle serveru. Ten ich použije napríklad na identifikáciu užívateľa alebo nastavenie preferovaného jazyka.

[3] `https://www.facebook.com/`
[4] `https://mail.google.com`

Identita Definuje, *kto* je užívateľ. Príkladom môže byť užívateľské meno, ktoré je unikátne.

Autentifikácia Overenie identity užívateľa, či je naozaj ten, za koho sa vydáva. Najbežnejší spôsob autentifikácie je zadanie hesla.

Autorizácia Rozhodnutie, či má daný užívateľ právo na získanie požadovaného obsahu, alebo vykonanie nejakej akcie. Zvyčajne nastáva po autentifikácii.

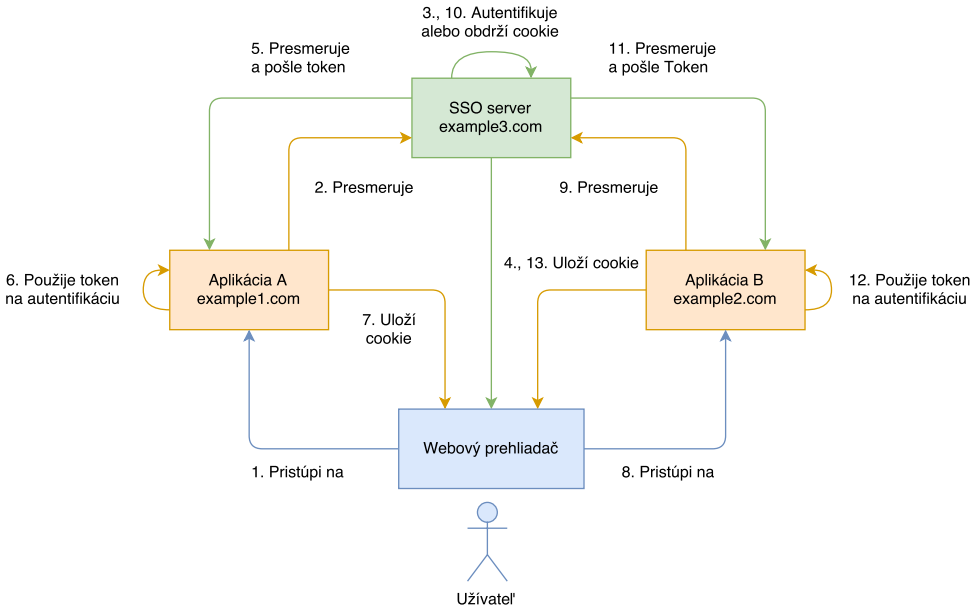
LDAP V dlhom tvare *Lightweight Directory Access Protocol*, je protokol určený na komunikáciu medzi *directory* službou a jej klientom. Aktuálna verzia tohoto protokolu je definovaná tu [4]. LDAP sa stal tiež synonymom pre množstvo pojmov súvisiacich s týmto protokolom, napríklad LDAP server. Directory služba (server) poskytuje informácie uložené v špeciálnom druhu úložiska, ktoré sa nazýva *directory*. Toto pozostáva zo záznamov dát zvaných *entry* a tie vytvárajú štruktúry v podobe stromov. Každý entry má unikátne *Distinguished Name (DN)* v rámci celého directory, čo je vlastne cesta k tomuto entry. Každý entry má atribúty, ku ktorým sú priradené ich hodnoty, čo je vlastne spôsob, akým sú tu uložené informácie. Takéto úložisko je navrhnuté pre veľmi efektívne čítanie a vyhľadávanie, avšak nemali by sa v ňom ukladať často sa meniace dáta, pretože je pomalé pre zápis [5].

SSO Je webová aplikácia, pomocou ktorej sa užívateľ prihlási do jednej webovej aplikácie a pri prístupe na ďalšie je do nich automaticky prihlásený bez toho, aby musel znovu zadávať svoje prihlasovacie údaje.

Táto definícia je veľmi stručná, takže pre úplné pochopenie toho, ako SSO vykonáva svoju funkcionalitu, je potrebné vysvetlenie jeho architektúry. Podotýkam, že popísané princípy sú, s menšími odchylkami, spoločné pre všetky SSO systémy nezávisle na konkrétnej implementácii.

SSO zabezpečuje autentifikáciu užívateľov iným webovým aplikáciám. Ukážem to na príklade. Mám dve aplikácie A a B. A je prístupná na doméne `example1.com` a B na `example2.com`. Aby ich užívateľ mohol plnohodnotne využívať, musí sa prihlásiť. Uvažujem tiež, že do oboch aplikácií má rovnaké prihlasovacie údaje. Otázka znie, je možné aby bol užívateľ, ktorý sa prihlási do aplikácie A automaticky prihlásený pri prístupe na aplikáciu B?

Dalo by sa to spraviť jednoducho pomocou cookies. Aplikácia A vytvorí cookies v prehliadači užívateľa pri prihlásení. Aplikácia B si ich prečíta a na základe nich ho automaticky prihlási. Tento prístup má však podľa bezpečnostného pravidla *same origin policy*, ktoré používajú prehliadače, jeden zásadný problém. *Stránka môže nastaviť cookie len pre vlastnú doménu, alebo*



Obr. 1.1: Architektúra SSO. Prebraté z: [7]

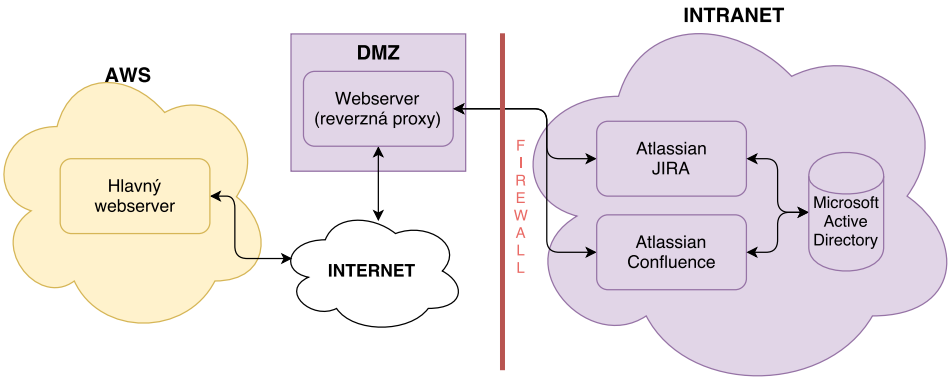
pre ktorúkoľvek rodičovskú doménu v prípade, že táto rodičovská doména nie je verejná prípona[5] [6].

Tento problém rieši SSO v závislosti na konkrétnom protokole, ktorý používa. Predpokladám, že na odoslanie informácie o užívateľovi SSO klientom využije SSO server *JSON Web Token*[6]. Potom užívateľ pristúpi na aplikáciu A a tá ho presmeruje na SSO server, ktorý beží na samostatnej doméne `example3.com`. Tu sa užívateľ prihlási, server si vytvorí cookie a presmeruje ho naspäť na A. Zároveň pošle Token o identite užívateľa klientovi A, na základe ktorého A užívateľa autentifikuje. Pri prístupe na klienta B je užívateľ znovu presmerovaný na SSO server, ktorému však tentoraz pošle cookie a vďaka tomu je presmerovaný naspäť na klienta bez potreby sa znovu prihlasovať [7]. Pre prehľadnosť je celá táto situácia znázornená na obrázku 1.1.

1.2 Aktuálny stav

Prvým logickým krokom pre úspešné nasadenie SSO riešenia v existujúcom prostredí firmy **Ataccama Software, s.r.o.**[7], je analýza jej informačnej infraštruktúry. Zjednodušene povedané to znamená popísať aplikácie a služby, ktoré sa v tomto prostredí nachádzajú, a vysvetliť vzťahy medzi nimi. Vďaka

[5] Napríklad .com alebo .sk
[6] `https://tools.ietf.org/html/rfc7519`
[7] `https://www.ataccama.com`



Obr. 1.2: Aktuálna infraštruktúra firmy.

tomuto rozboru vymedzím aplikácie, ktoré sú vhodné na začlenenie do zamýšľaného SSO systému. Na základe toho definujem očakávaný výsledný stav a určím ďalšie kroky, ktoré sú potrebné na jeho dosiahnutie. Taktiež načrtnem zjednodušené schémy pre lepšie pochopenie uvedených situácií.

Z pochopiteľných dôvodov a v záujme stručnosti sme s vedúcim práce vybrali len relevantnú podmnožinu celkovej infraštruktúry firmy. Preto sa teda jedná hlavne o aplikácie s veľkým množstvom používateľov a s nimi súvisiace prvky infraštruktúry. Celá firemná sieť je rozdelená na dva celky, v závislosti na ich fyzickom umiestnení.

Prvá časť beží v cloudovom prostredí firmy Amazon.com, Inc., ktoré sa nazýva Amazon Web Services, Inc.[8], alebo len skrátene AWS. AWS ponúka široké možnosti informačných riešení, ako napríklad výpočetnú silu, dátové úložiská, databázy, analytické nástroje, sieťovanie a ďalšie. Navyše, všetky tieto služby bežia v cloude, takže si môžem jednoducho zvoliť aj ich požadovanú lokáciu [8]. Toto je aj jeden z dôvodov, prečo sme sa rozhodli časť nášho prostredia hostovať u tejto spoločnosti. Zo serverov, ktoré ma zaujímajú, je tu hlavný webserver.

Druhá časť je naša lokálna firemná infraštruktúra. Tá sa skladá z dvoch častí. Prvá je DMZ[9], čiže oblasť, ktorá je otvorená do internetu, z bezpečnostných dôvodov je vyčlenená z intranetu a nachádza sa pred firewallom. Ten chráni druhú časť – firemný intranet. V DMZ beží webserver, ktorý funguje ako reverzná proxy pre webové aplikácie Atlassian JIRA, Atlassian Confluence a niektoré ďalšie, ktoré nie sú pre účely tejto práce podstatné. Tieto sa nachádzajú v intranete, a tam sú napojené na Microsoft Active Directory. Celú infraštruktúru zachytáva obrázok 1.2.

Microsoft Active Directory[10] je *adresárová služba spoločnosti Micro-*

[8] `https://aws.amazon.com`

[9] Skratka pre anglický výraz „*demilitarized zone*“.

[10] `https://msdn.microsoft.com/en-us/library/bb742424.aspx`

soft. Acitve Directory je centralizovaný a štandardizovaný systém, ktorý automatizuje sieťovú správu užívateľských dát, bezpečnosti, a distribuovaných zdrojov [9]. Pre potreby tejto práce je dôležité, že okrem interných účelov ho používame na správu zamestnaneckých účtov, ktorých už je aktuálne vyše sto.

Atlassian Confluence[11] je webová aplikácia vyvíjaná austrálskou spoločnosťou Atlassian. Je napísaná v jazyku Java a na svoj chod využíva webový server Apache Tomcat, ku ktorému sa ešte dostaneme. V našej firme pomocou nej zdieľame interné dokumenty a takisto v nej zdieľame dokumentáciu k našim produktom pre našich zákazníkov. Zamestnanci sa do nej prihlasujú pomocou užívateľských účtov, ktoré sú uložené v Active Directory. Zákazníci sa sem prihlasujú pomocou lokálne vytvorených účtov, na základe ktorých im je sprístupnená určitá časť dokumentácie.

Atlassian JIRA[12] je taktiež aplikácia vyvíjaná spoločnosťou Atlassian, naprogramovaná v Jave a na svoj chod používa Tomcat. Využívame ju na *bug tracking* našich produktov. Zákazníci tu majú po prihlásení možnosť vytvoriť tickety pre vyriešenie svojich problémov a nahlásenie chýb v našich aplikáciách. Zamestnanecké účty sú opäť dostupné z Active Directory a zákazníci majú účty vytvorené lokálne.

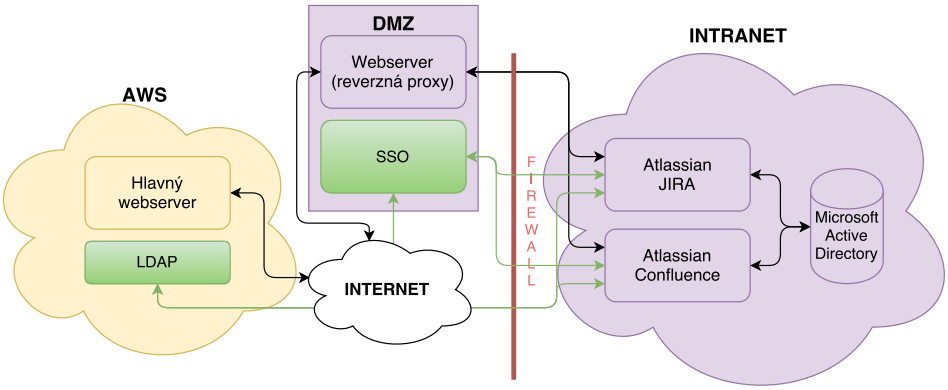
Hlavný webserver je inštancia najrozšírenejšieho webového servera Apache Httpd[13]. Ako som už spomínal, beží na virtuálnom serveri v AWS a obsluhuje požiadavky na adresu `https://www.ataccama.com/`. Užívatelia si môžu z našich stránok zadarmo stiahnuť náš produkt Data Quality Analyzer (DQA) v plnej verzii. Za výmenu sa tu zaregistrujú. Títo užívatelia sú uložení v lokálnej databázi, ako potencionálni budúci zákazníci.

1.3 Nový stav

Teraz, keď už som rozobral stav, v ktorom sa momentálne naša infraštruktúra nachádza, je ďalším krokom navrhnúť jej vylepšenia a samozrejme zaintegrovať SSO. Tu by sa hodilo podotknúť, že zákazníci, ktorých je aktuálne zhruba tisíc a stále pribúdajú, sú rozdelení na tri skupiny. Prvá má účet iba na JIRE, druhá iba na Confluence a tretia má účty na oboch aplikáciách. Preto sa ako prvé vylepšenie ponúka zjednotiť tieto rastúce lokálne užívateľské databázy do jednej a prípadne ju externalizovať. Najlepšie bude, ak sa vytvorí samostatný server, na ktorom bude bežať služba pre správu užívateľov. Na tieto účely by bol vhodný buď spomínaný Active Directory, alebo LDAP server.

Active Directory však využívame výlučne v intranete firmy, a preto nie je vhodné ho použiť na správu zákazníkov. Má to viacero dôvodov. Účty zamestnancov a zákazníkov by sa nachádzali na jedinom serveri. To by znížilo

[11] `https://www.atlassian.com/software/confluence`
[12] `https://www.atlassian.com/software/jira`
[13] `https://httpd.apache.org/`



Obr. 1.3: Návrh novej infraštruktúry firmy. Zelenou farbou sú vyznačené nové časti.

bezpečnosť firmy tým, že by sa tento systém otvoril do internetu, čo nie je vôbec žiaduce. Pre externých užívateľov taktiež nevyužijeme enterprise funkcionalitu, ktorú Active Directory ponúka.

Preto sa ako omnoho lepšia možnosť ponúka vytvoriť samostatný LDAP server, s odľahčenou funkcionalitou a ideálne open source. Vytvorenie oddelených užívateľských databáz je flexibilnejšie a bezpečnejšie. V našej internej sieti sa takáto zmena neprejaví negatívne. Taktiež časom plánujeme všetky aplikácie ktoré sa nevyužívajú výlučne na interné účely presunúť do AWS, a tak môžeme tento LDAP vytvoriť priamo tam. Ako som už spomínal, hlavná zmena bude nasadenie SSO. Keď to zhrniem, tak po dôkladnej analýze možných riešení, a konzultáciách s vedúcim práce som dospel k týmto krokom:

1. Zvoliť vhodnú implementáciu LDAP servera a pomocou neho zjednotiť užívateľské databáze aplikácií JIRA a Confluence do jednej.

2. Vytvoriť samostatný virtuálny server a na ňom nasadiť vhodné SSO riešenie, ktoré preberie zodpovednosť za autentifikáciu užívateľov do aplikácií JIRA a Confluence.

3. Pripojiť JIRU, Confluence a SSO na nový LDAP server. Zároveň pripojiť SSO na existujúci Active Directory.

4. Nakonfigurovať JIRU a Confluence tak, aby používali na autentifikáciu SSO.

Na hlavnom webserveri sa zatiaľ nevyužíva customizovaný obsah, a ani vytvorené užívateľské profily. Tie slúžia iba na interné účely. Z toho dôvodu nie je momentálne nutné tento server začleniť do SSO systému. LDAP server by mal slúžiť na správu *skutočných* zákazníkov. Návrh možnej infraštruktúry

firmy po splnení týchto bodov ukazuje obrázok 1.3. Tieto zmeny so sebou prinesú okrem výhod, ktoré sú zrejmé z prvej kapitoly, tiež nasledujúce:

- Zjednotenie užívateľských účtov zákazníkov pod jednu správu LDAP servera. To zjednoduší ich administráciu, zvýši bezpečnosť, rozšíri možnosti správy a zabezpečí trvalé riešenie problému s ich pribúdajúcim počtom. Umožní to aj jednoduchú rozšíriteľnosť tohoto prostredia o nové aplikácie.

- Centralizácia autentifikácie poskytne zjednotenie prostredia aj z pohľadu užívateľov. Vďaka nej sa zefektívni ich práca a zlepší *user experience*. Taktiež to napomôže rozšíriteľnosti.

- Z najväčšej výhody vyplýva tiež jedna nevýhoda a to tá, že ak útočník získa nejaké prihlasovacie údaje, má tak prístup ku všetkým aplikáciám, ktoré sú na SSO napojené. Má to ale jednoduché riešenie. Takýto účet sa dá rýchlo zablokovať jediným nastavením v LDAP serveri alebo Active Directory.

S týmito zmenami úzko súvisí aj niekoľko bezpečnostných rizík, ktoré musím zobrať do úvahy. Keďže sa jedná o veľkú centralizáciu, vzniká problém *single point of failure* [10]. To znamená, že ak zlyhá SSO alebo LDAP, zlyhá celé riešenie. Preto treba správne nastaviť logovanie, zálohovanie a monitoring týchto serverov a aplikácií. Aby bolo možné v prípade zlyhania rýchlo zistiť jeho príčinu a vykonať potrebné opatrenia na odstránenie závad. Taktiež je nutné zaistiť šifrovanie komunikácie a dostatočne silné hešovanie uložených hesiel.

Kapitola 2

Výber SSO implementácie

V predchádzajúcej kapitole som popísal a vysvetlil zmeny v infraštruktúre. Jednou z nich je aj nasadenie SSO systému. Na to, aby som to mohol urobiť, musím vybrať konkrétnu implementáciu. Podrobný popis každej implementácie nie je pri ich veľkom množstve prakticky možný a nie je pre účely tejto práce ani žiadúci. Preto bude najlepšie spísať zoznam požiadaviek, ktoré by mala daná implementácia spĺňať, a na ktoré sa zameriam. Z nich potom budem vychádzať pri výbere najvhodnejšieho kandidáta. Na základe výsledkov predchádzajúcej analýzy som zhrnul tieto kritériá výberu:

Cena Peniaze sú najdôležitejší faktor. Komerčné riešenie stojí kľudne aj tisíce dolárov. Napríklad *Crowd* od firmy Atlassian: 8 000 $ jednorázovo[11], alebo *Auth0* od rovnomennej firmy: 2 035 $ ročne[12][14]. Toto vôbec nie je lacné. Našťastie je tu svet slobodného softvéru, ktorý neustále rastie. Podľa môjho predbežného prieskumu existuje aj viacero kvalitných *open source* SSO riešení, ktoré si nižšie porovnáme.

Integrácia Na základe predchádzajúcej analýzy je pre nás kľúčová integrácia s našim konkrétnym prostredím. Teda s aplikáciami JIRA, Confluence a LDAP protokolom (zahŕňa Active Directory). V ideálnom prípade by mala existovať dokumentácia alebo pluginy pre integráciu týchto aplikácií a protokolov.

Dokumentácia Pre úspešné nasadenie SSO implementácie je potrebná kvalitná dokumentácia, ktorá popisuje všetky vlastnosti a možnosti daného softvéru. Čím obsiahlejšia a podrobnejšia dokumentácia, tým ľahšia je modifikácia a riešenie prípadných problémov.

Rozšíriteľnosť Ide v podstate o podporu rôznych protokolov, pluginov a vo všeobecnosti vlastnosti, ktoré sa môžu využiť pri budúcom rozširovaní našej infraštruktúry.

[14] Uvažujeme tisíc a viac užívateľov, a sto zamestnancov. U Auth0 produkt *Developer Pro*, kvôli pripojeniu na Active Directory a LDAP.

Jednoduchosť Na základe predchádzajúcej analýzy môžem povedať, že naše prostredie je dosť špecifické a vyžaduje „čistú“ SSO funkcionalitu. Preto by mala byť daná implementácia jednoduchá, jednoducho sa nasadzovať a nemala by obsahovať štandardné, rôzne nechcené či nevyužité vlastnosti. To ale nutne neznamená, aby bola minimalistická. To znamená, že v ideálnom prípade by som mal mať možnosť každú vlastnosť „zapnúť“ alebo „vypnúť“ podľa potreby.

Atlassian na svojich stránkach uvádza zoznam SSO [13], u ktorých sa podarila úspešná integrácia s ich produktami. Sú tam uvdené aj konkrétne pluginy pre tieto aplikácie. Teraz spravím ich porovnanie na základe vyššie uvedených kritérií. Začnem implementáciami, ktoré sú dostupné zadarmo, keďže cena je pre nás najpodstatnejšia. Ak žiadna z nich nebude vyhovovať mojim požiadavkám, preskúmam aj komerčné riešenia.

CAS[15] je open source enterprise SSO vyvíjané spoločnosťou Apereo Foundation. Architektúra CASu je veľmi jednoduchá a skladá sa len z dvoch častí. CAS Server sa stará o autentifikáciu užívateľov, a na základe toho im umožňuje prístup k CAS Klientom. To je softvér integrovaný priamo do aplikácií, ktorý umožňuje komunikáciu s CAS Serverom [14]. Tento CAS Klient implementuje integráciu pre aplikácie JIRA a Confluence [15]. Dokumentácia je veľmi prehľadná a výborne napísaná, čo dokazujú predchádzajúce odkazy. CAS taktiež podporuje množstvo protokolov (vrátane LDAP), z ktorých si môžem podľa potreby vybrať iba tie, ktoré potrebujem. Obsahuje tiež predpripravené implementácie klientov v rôznych jazykoch [14]. Vďaka tomu má rozšíriteľnosť veľký potenciál.

Shibboleth[16] *je štandardizované open source webové SSO riešenie, ktoré funguje či už v rámci organizácie, alebo mimo jej hraníc* [16]. Architektúra je už o niečo zložitejšia oproti CASu, ale tiež sa v podstate zakladá na dvoch ústredných bodoch [17]. V tomto prípade sa server nazýva Identity provider a klient Service provider. Medzi kľúčové vlastnosti servera mimo iné patrí *natívna podpora LDAP, podpora čítania užívateľských dát z LDAP adresárov a vykonávanie jednoduchých alebo komplexných transformácií týchto dát*, a bezpečný spôsob ich poskytovania klientom tretích strán [17]. Rozšíriteľnosť tohoto riešenia je obrovská vďaka veľkému množstvu pokročilých vlastností. Shibboleth poskytuje viacero vlastností *out-of-the-box* [17], čo považujem v mojom prípade skôr za mínus. Dokumentácia je veľmi zrozumiteľná a kvalitná, avšak rozsiahla s kopou možností, čo sa odrazilo na jej prehľadnosti. Ako príklad dávam dokumentáciu Identity providera [18]. Integrácia pre Confluence je poskytnutá zdarma pluginom v Atlassian obchode [19]. JIRA je podporovaná portovaným Shibboleth autentifikátorom s podrobnou dokumentáciou [20].

[15] `https://www.apereo.org/projects/cas/about-cas`
[16] `https://shibboleth.net/`

JOSSO[17] má podľa oficiálnej dokumentácie, ktorá je veľmi kvalitná a vyčerpávajúca, dve generácie [21]. Všetky ďalšie tvrdenia sú postavené na tomto dokumente [21] ak nie je uvedené inak. V podkapitole 1.2 sa píše, že by sa malo *zvážiť použitie JOSSO1 pre riešenie jednoduchších SSO scenárov v rozsahu jedinej administratívnej jednotky, s minimálnymi alebo so žiadnymi požiadavkami na spoluprácu s externými entitami(partneri, dodávatelia, vetvy...)* [21], čo je presne môj prípad. Preto ďalej uvažujem použitie JOSSO1. Integrácia s našim prostredím je riešená pomocou komponenty rozšírenia JOSSO1 aplikácie pre *Atlassian Seraph* autentifikátor[18], chýba tam však dokumentácia, ktorá sa ale dá nájsť na internete, avšak len pre JIRU [22]. Nepodarilo sa mi nájsť návod ku Confluence. Rozšíriteľnosť je tu vynikajúca. Samozrejmé sú široké možnosti konfigurácie a podpora rôznych protokolov, ktoré zahŕňajú LDAP. JOSSO1 je podobne ako CAS veľmi modulárny a umožňuje vďaka Apache Maven systému vytvoriť aplikáciu „na mieru“ na základe požadovaných vlastností, ktoré si užívateľ sám zvolí.

Open Fusion[19] je *odľahčený SSO autentifikačný modul pre Apache Httpd. Používa bezpečné tickety založené na cookies k tomu, aby implementoval SSO rozhranie, ktoré pracuje medzi viacerými Httpd inštanciami a servermi* [23]. Taktiež sa tam píše, že *samotná autentifikácia je vykonávaná užívateľsky dodaným CGI alebo skriptom v ľubovoľnom jazyku.* Nepodarilo sa mi však nájsť žiadne pluginy, ani dokumentáciu, ako toto riešenie prepojiť s JIRA a Confluence. To by znamenalo, že by som musel autentifikátor napísať sám. Z tohto dôvodu nemá zmysel sa ním ďalej zaoberať, pretože existujú lepšie riešenia.

Keď zhodnotím výsledky jednotlivých vyššie popísaných SSO implementácií s ohľadom na definované kritériá, nebude potrebné analyzovať ďalšie možnosti. Keďže Open Fusion nesplnil základné kritérium, stručne zhrniem porovnanie zvyšných troch kandidátov.

Ako najlepšie riešenie spĺňajúce všetky zadané body sa javí CAS. Shibboleth je veľmi komplexný, rieši množstvo pokročilých scenárov, a preto si myslím, že by zostal kvôli nášmu jednoduchému účelu nevyužitý, a jeho široké možnosti by boli skôr na obtiaž. JOSSO vyzerá ako veľmi dobrá voľba až do momentu, keď treba zaintegrovať Confluence. Po prediskutovaní všetkých riešení s vedúcim práce sme sa obaja zhodli, že najlepšou možnosťou je CAS. Preto nasadím práve CAS SSO implementáciu.

[17] `http://www.josso.org/`

[18] Nachádza sa tu: `https://github.com/atricore/josso1/tree/master/components/josso-seraph-extension/src/main/java/org/josso/atlassian`

[19] `http://www.openfusion.com.au/labs/mod_auth_tkt/`

Kapitola 3

Návrh LDAPu

3.1 Výber LDAP implementácie

Predtým, než začnem so samotným CASom, navrhnem správu externých užívateľov pomocou LDAP servera, vďaka ktorému bude CAS autentifikovať zákazníkov. Tu je vhodné poznamenať, že nasadenie a konfigurácia tohoto servera, takisto ako migrácia existujúcich účtov z JIRY a Confluence nie sú zahrnuté v tejto práci.

Keďže na tento server nemáme žiadne špeciálne požiadavky, okrem podpory protokolu LDAP a štandardných požiadaviek ako sú bezpečnosť, dokumentácia, stabilita a podobne, preskúmam štyri najznámejšie open source riešenia podľa tohoto zdroja [24], a z nich jedno podľa vlastných preferencií s ohľadom na naše prostredie vyberiem.

OpenLDAP[20] je pravdepodobne najznámejší open source projekt v tejto oblasti. Jeho softvérový balíček sa nachádza predpripravený v repozitároch každej hlavnej linuxovej distribúcie. Má kvalitnú dokumentáciu a dobrú bezpečnosť. Samotný OpenLDAP je program, ktorý sa ovláda z príkazového riadka, a preto je pre prehľadnejšiu inštaláciu a konfiguráciu potrebné nainštalovať dodatočný softvér. Z tohoto dôvodu sa pre začiatočníkov v tejto oblasti odporúča skôr ApacheDS alebo OpenDJ, ktoré predstavím ďalej.

ApacheDS[21] je server naprogramovaný v jazyku Java a je postavený na najnovšej verzii 3 LDAP protokolu. Je to projekt známej opensourcovej softvérovej nadácie Apache Software Foundation[22]. Štandardne sa dodáva s grafickou aplikáciou **Apache Directory Studio**[23], ktorá funguje ako klient a umožňuje jednoduchšiu konfiguráciu servera. Vďaka Jave beží prakticky na všetkých platformách.

[20] `http://www.openldap.org/`
[21] `http://directory.apache.org/apacheds/`
[22] `https://www.apache.org/`
[23] `http://directory.apache.org/studio/`

OpenDJ[24] je pokračovanie predchádzajúceho projektu s názvom OpenDS, ktorý bol vytvorený spoločnosťou Sun Microsystems a od roku 2010 už pod názvom OpenDJ vyvíjaný spoločnosťou Oracle Corporation[25]. Je napísaný v Jave a vďaka tomu beží na všetkých platformách, podobne ako ApacheDS. Taktiež obsahuje dobrú dokumentáciu a klientskú aplikáciu pre prehľadnejšiu konfiguráciu.

389 Directory Server[26] je LDAP server s pokročilými rysmi a funkcionalitou, ktorý je vyvíjaný spoločnosťou Red Hat[27]. Tak ako predchádzajúce riešenia, aj toto má dobrú dokumentáciu a grafického klienta pre uľahčenie konfigurácie.

Všetky tieto možnosti vyzerajú veľmi podobne, ale keďže sa musím rozhodnúť pre jednu, vybral som ApacheDS. Je to hlavne z toho dôvodu, že vo firme dlhodobo používame viacero produktov nadácie Apache Software Foundation, ako napríklad Httpd, Tomcat a Hadoop. Na základe mojich skúseností môžem povedať, že ich produkty majú výbornú dokumentáciu, množstvo možností konfigurácie, neustály vývoj, stabilitu, podporu najnovších štandardov a v neposlednom rade dbajú na bezpečnosť.

3.2 Definícia LDAP štruktúry

V tejto podkapitole zadefinujem štruktúru, podľa ktorej bude LDAP server nakonfigurovaný a podrobne ju popíšem. Každý prvok vysvetlím slovne a uvediem zápis v typickom formáte pre LDAP. Zároveň rozpíšem dôvody a účely, ktoré viedli k jednotlivým rozhodnutiam.

Ako som už naznačil v prvej kapitole, LDAP directory má štruktúru stromu. Element root (koreň) obsahuje partície, ktoré sú unikátne, nezávislé a definujú príponu všetkých podradených entries (záznamov) v danej partícii [25]. Najprv teda definujem partíciu. Jej prípona sa zvyčajne nazýva podľa doménového mena, napríklad `dc=clients,dc=com`, kde `dc` je skratka pre domain component. Vo vnútri tejto partície sa budú nachádzať všetky logické jednotky, ktoré budú obsahovať samotné entries.

Prvou logickou jednotkou sú skupiny, ktoré definujem ako `ou=Groups`, kde `ou` je skratka pre organizational unit. Ako som už spomenul v prvej kapitole, užívatelia sú imaginárne rozdelení do skupín, na základe prístupu do JIRY, Confluence alebo do obidvoch. Preto sa tu budú nachádzať dve skupiny JIRA `cn=JIRA` a Confluence `cn=Confluence`, kde `cn` je skratka pre common name. Každá skupina musí obsahovať atribút `cn` pre unikátny názov a atribút `uniquemember` pre množinu užívateľov, ktorí do nej patria. K doku-

[24] `https://backstage.forgerock.com/docs/dj`
[25] `https://www.oracle.com/index.html`
[26] `http://directory.fedoraproject.org/`
[27] `https://www.redhat.com/en`

mentácii má štandardne prístup každý užívateľ, a preto bude skupina Confluence priradená každému automaticky pri registrácii. V prípade pridania novej aplikácie do nášho prostredia, môžem veľmi efektívne filtrovať zákazníkov, ktorí majú do nej prístup.

Ďalšou časťou sú produkty, ktoré definujem ako `ou=Products`. Každý produkt bude mať vlastný entry s povinnými atribútmi `cn` pre názov a `uniquemember` pre užívateľov, ktorí majú zakúpený daný produkt. Vďaka tomuto bude môcť Confluence autorizovať užívateľa a ďalej filtrovať obsah dokumentácie na základe jeho produktov. To v praxi znamená, že každému užívateľovi sa po prihlásení sprístupní iba relevantná dokumentácia ku produktom, ktoré má kúpené. Zefektívni to napríklad vyhľadávanie v dokumentácii.

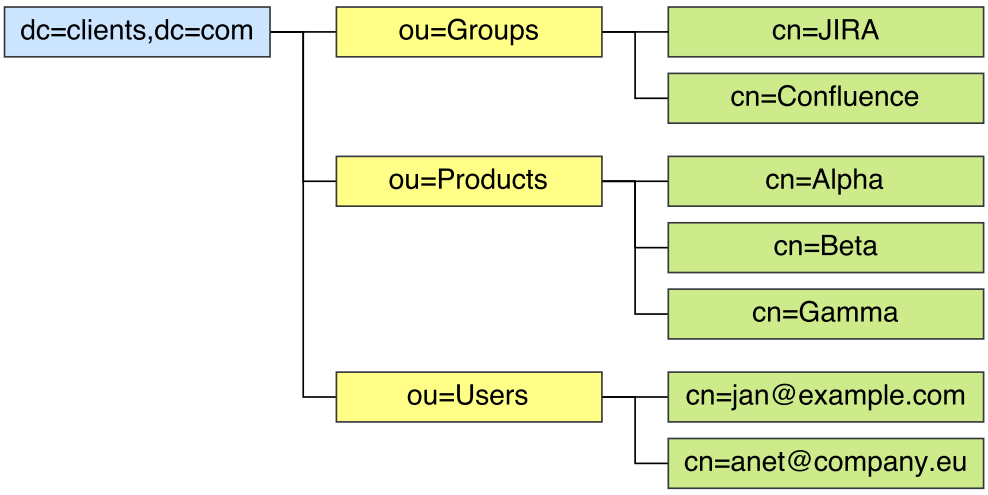
Potom tu sú samozrejme užívatelia, ktorých definujem ako `ou=Users`. Každý entry reprezentuje zaregistrovaného zákazníka a obsahuje tieto povinné atribúty:

- `cn` je unikátne meno, zvyčajne sa zhoduje s emailom užívateľa.
- `email` pri zmene emailu sa modifikuje tento atribút a nie predchádzajúci.
- `gn` krstné meno (given name).
- `sn` priezvisko (surname).
- `password` prihlasovacie heslo.

Ďalej obsahuje tieto voliteľné atribúty:

- `policy` užívateľ súhlasil so spracovaním osobných údajov.
- `belongsTo` môže obsahovať viacero hodnôt. Určuje príslušnosť užívateľa ku skupinám a produktom.
- `isActive` má vždy hodnotu true alebo false, podľa toho, či užívateľ aktivoval svoj účet kliknutím na odkaz v emaili pri registrácii.
- `emailUpdates` booleovská hodnota, ktorá určuje či si zákazník praje doručovanie našich emailov.

Celkovú výslednú štruktúru LDAPu znázorňuje obrázok 3.1. V ďalších kapitolách už budem predpokladať existenciu úspešne nainštalovaného a nakonfigurovaného ApacheDS servera podľa definovanej schémy.



Obr. 3.1: Ukážková štruktúra LDAPu.

Kapitola 4

Nasadenie a konfigurácia CAS servera

4.1 Architektúra a CAS protokol

Túto kapitolu začnem teoretickým úvodom do Central Authentication Service, ktorý som vybral ako najvhodnejšie riešenie pre naše prostredie v kapitole 2. Teraz podrobne rozpíšem architektúru tejto implementácie na základe dokumentácie [14]. Architektúra CASu je primárne zložená z dvoch hlavných častí – CAS servera a CAS klientov, ktorí spolu komunikujú pomocou podporovaných protokolov. Z nich je implicitný CAS protokol [26], preto ho vysvetlím a použijem v tejto práci.

CAS server je webová aplikácia napísaná v jazyku Java. Ako celok je rozdelená na tri súčasti:

Web Časť, ktorá je postavená na Spring frameworku[28]. Umožňuje komunikáciu s ostatnými účastníkmi systému, či už s užívateľmi, alebo CAS klientami.

Ticketing Stará sa o vytváranie a správu vydaných *Ticket Granting Ticketov* (TGT) a *Service Ticketov* (ST), ktorých funkciu vysvetlím ďalej.

Autentifikácia Zvyčajne je vykonaná iba na začiatku danej session užívateľa, kde overí jeho identitu napríklad voči LDAPu.

Hlavnou úlohou CAS servera je autentifikovať užívateľov a na základe toho im umožňovať prístup ku chráneným webovým aplikáciám, zvaným tiež CAS klienti. Túto úlohu plní vďaka ticketovaciemu systému, ktorý úzko súvisí s CAS protokolom a funguje nasledovne.

[28] `https://spring.io/`

Keď sa užívateľ úspešne prihlási, server vystaví TGT, ktorý reprezentuje užívateľovu SSO session. Tento TGT užívateľ obdrží v CASTGC cookie. Následne, na základe tohto cookie je užívateľ presmerovaný na CAS klienta, ktorému server vystaví ST. ST je odoslaný ako HTTP GET parameter v URL presmerovania. Pri prvom takomto prístupe si klient komunikáciou so serverom overí, že tento ST je platný. Keď je toto overenie úspešné, klient pošle užívateľovi cookie, ktoré mu umožňuje prístup k danému klientovi – aplikácii.

Pri prístupe na druhú aplikáciu je užívateľ presmerovaný na SSO server, ktorému zároveň odošle CASTGC cookie. Ak je overenie serverom tohoto TGT cookie úspešné, užívateľ sa nemusí znovu prihlasovať a je priamo presmerovaný na druhú aplikáciu – klienta. Druhý klient si zas overí ST, ktorý obdržal v HTTP požiadavke voči serveru. V prípade úspechu je užívateľovi odoslané cookie umožňujúce prístup na druhého klienta. Pre lepšiu prehľadnosť a pochopenie toho, ako CAS protokol, server a klienti fungujú, prikladám, obrázok, ktorý som musel kvôli veľkosti rozdeliť na dve časti 4.1 4.2. Podrobná špecifikácia najnovšieho CAS protokolu 3.0.2 sa nachádza tu [27].

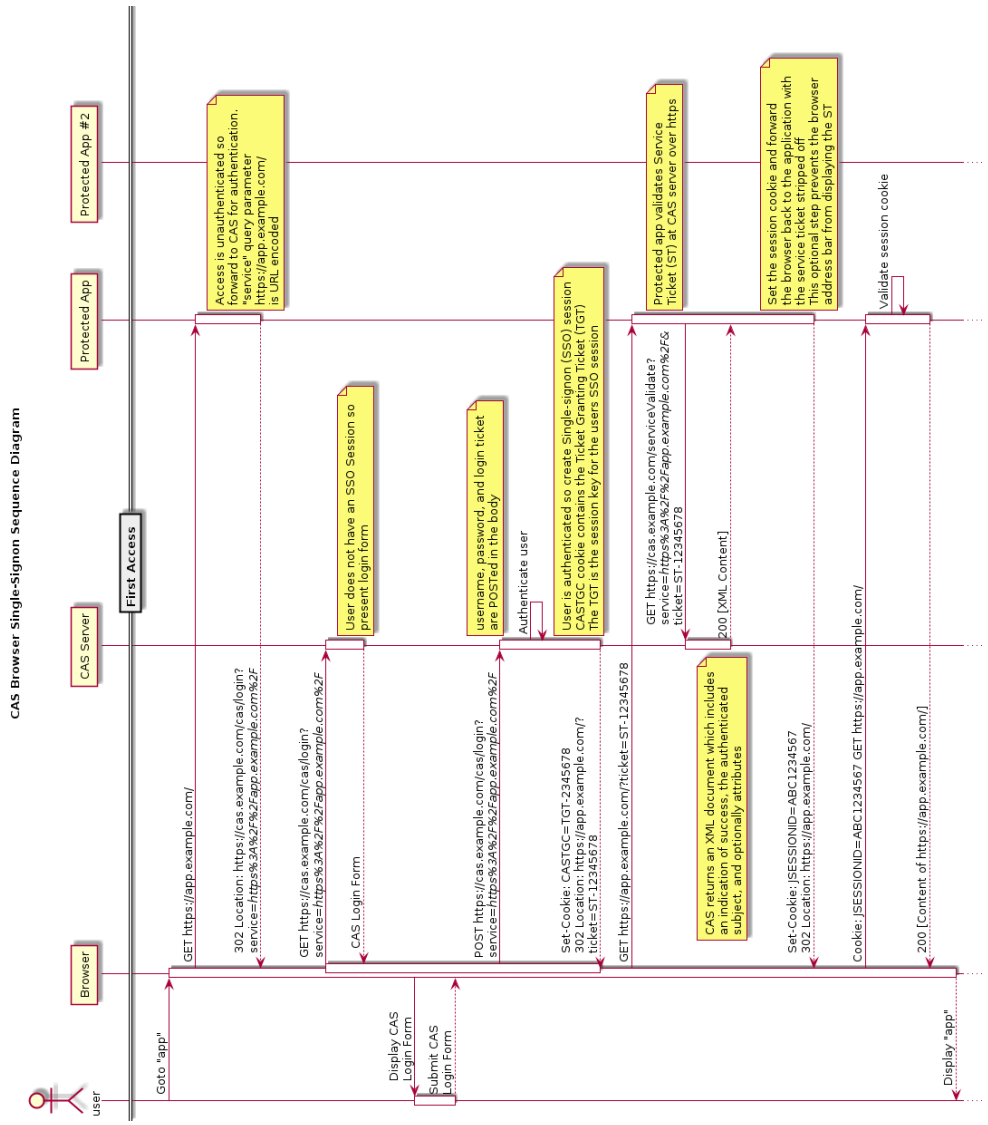
CAS klient má dva významy. Môže to byť klient, ktorý používa pre komunikáciu so serverom CAS protokol. Alebo to môže byť samostatný klient, či integrácia do aplikácie, ktorá je súčasťou CAS SSO systému (nezáleží na použitom protokole komunikácie). Tento klient, ako som už spomínal, má za úlohu overiť ST a vytvoriť cookies pre užívateľa. Tým zabezpečuje autorizovaný prístup na základe identity užívateľa do aplikácie. Existuje viacero hotových naprogramovaných CAS klientov v rôznych jazykoch ako Java, .NET, PHP, Perl a ďalšie. Takisto existuje klient pre rôzne aplikácie, z ktorých ma zaujíma hlavne Atlassian JIRA a Confluence [14].

Teraz vysvetlím architektúru CASu, ktorú použijem pre nasadenie do nášho prostredia. CAS klienti budú zaintegrovaní do JIRY a Confluence. Komunikácia so severom bude prebiehať pomocou CAS protokolu. Pre autentifikáciu bude CAS server komunikovať so servermi ApacheDS a Active Directory prostredníctvom LDAP protokolu. Celú situáciu zachytáva obrázok 4.3.

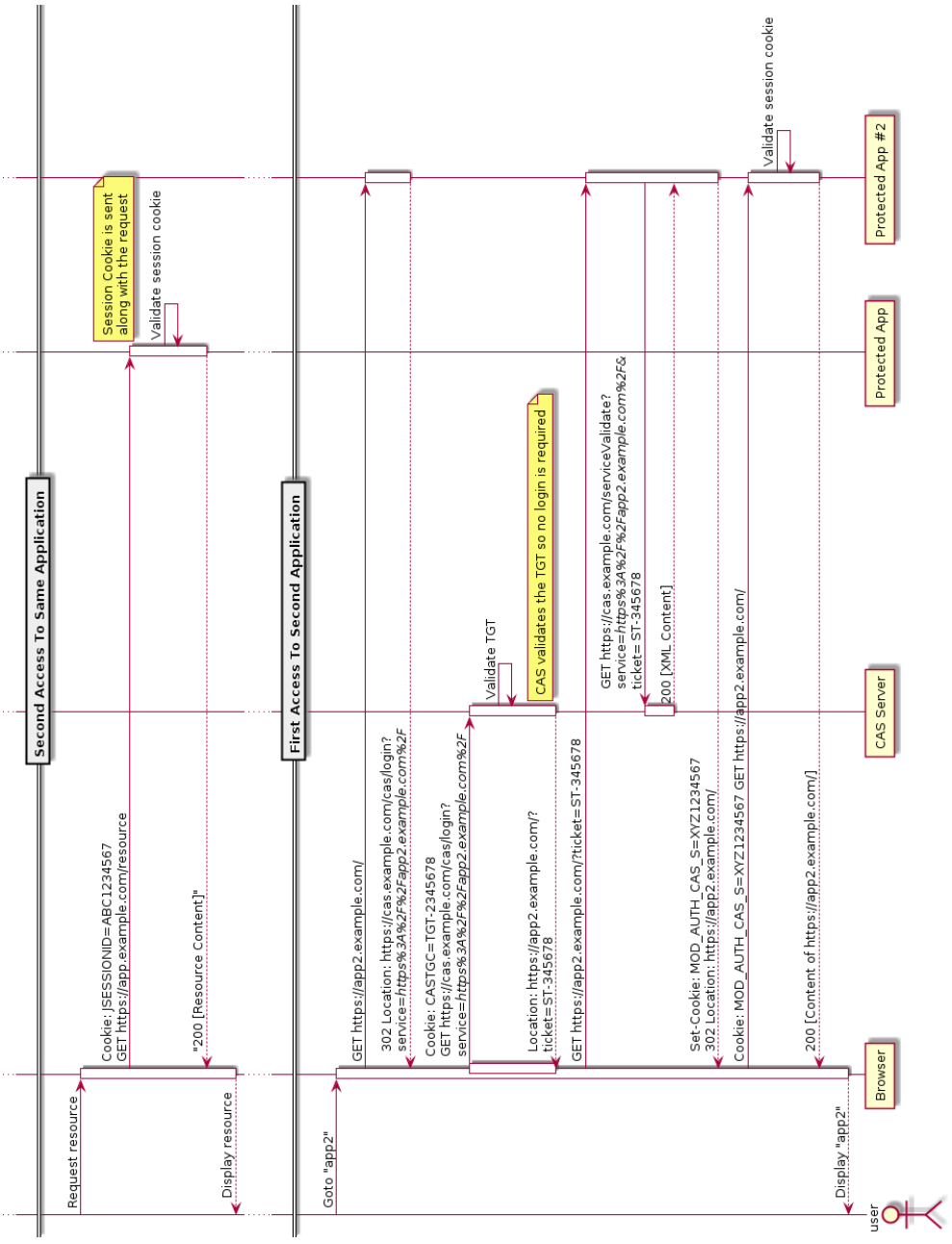
Z architektúry CASu a samotného protokolu jasne vyplýva, že bezpečnosť celého riešenia je postavená na šifrovaní každej komunikácie. Uvádza sa to aj v dokumentácii k bezpečnosti [28]. Preto je potrebné nakonfigurovať, aby komunikácia, ktorá prebieha prostredníctvom protokolu HTTP používala jeho šifrovanú variantu HTTPS. To isté platí pre komunikáciu protokolom LDAP.

4.2 Popis prostredia

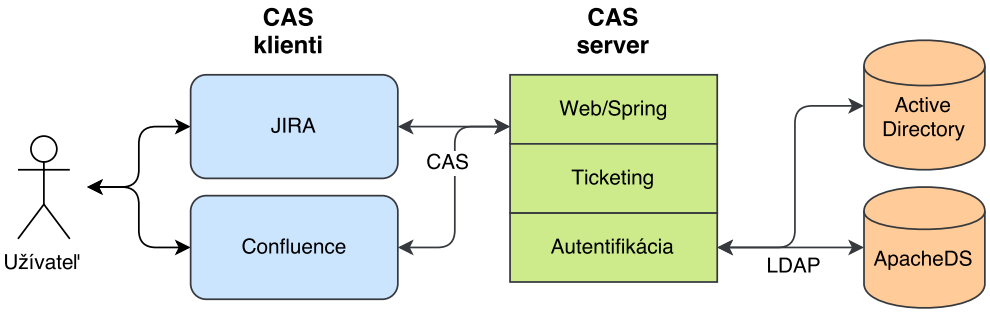
Pre potreby tejto práce som si vytvoril samostatné prostredie v intranete firmy, v ktorom budem nasadzovať, konfigurovať a testovať CAS. Toto prostredie sa skladá z troch virtuálnych serverov. Na prvom je spustená kópia produkčnej JIRY, ktorá je dostupná na adrese `https://testjira.ataccama.com`. Na ďal-



Obr. 4.1: Diagram fungovania CAS protokolu, časť prvá. Zdroj: [26].



Obr. 4.2: Diagram fungovania CAS protokolu, časť druhá. Zdroj: [26].



Obr. 4.3: Architektúra nasadenia CASu.

šom je kópia Confluence, ktorá sa nachádza na adrese `https://testconfluence.ataccama.com`.

Posledný server som pripravil pre CAS. Nainštaloval som tam najnovšiu verziu operačného systému CentOS 7[29], ktorú štandardne používame vo firme. Taktiež som tam pripravil softvér, ktorý vyžaduje CAS. Konkrétne sa jedná o poslednú verziu Oracle Java SE Development Kit[30] `1.8.0_131`. Jej inštalácia je veľmi jednoduchá. Stačí stiahnuť inštalačný balík vo formáte `.rpm` určený pre Red Hat operačné systémy, kvôli bezpečnosti overiť jeho integritu (porovnať hashe) pomocou programu `md5sum` a nainštalovať príkazom `sudo yum install <meno_balika>`.

Ďalej tam mám pripravený Apache Tomcat[31] v poslednej stabilnej verzii `8.5.14`. Ten stačí stiahnuť v distribúcii core, overiť jeho integritu a rozbaliť. Inštalačné pokyny sú v súbore `RUNNING.txt`. Tomcat potrebuje mať na svoj chod nastavenú premennú prostredia `JAVA_HOME`. To zariadia tieto príkazy:

```
sudo echo "export JAVA_HOME=/usr/java/jdk1.8.0_131" \
> /etc/profile.d/java.sh
sudo source /etc/profile.d/java.sh
```

Tu sa chvíľu zastavím pri Apache Tomcat. Je to open source implementácia webového servera, v ktorom sa spúšťajú javovské webové aplikácie. Štandardne počúva na porte `8080` a umiestnil som ho do adresára `/opt/tomcat`, kde je taktiež uložená jeho konfigurácia, logovanie a jednotilvé aplikácie. Na základe skúseností, ktoré s ním mám vďaka Atlasianím produktom, ho zvyčajne používam v reverse proxy[32] nastavení, čo mi umožňuje jeho jednoduchšiu konfiguráciu a v prípade, že budem chcieť tento SSO server otvoriť do internetu bude veľmi jednoduché nastaviť ako proxy napríklad spomínaný webserver v DMZ. Pre tento účel som si pripravil Apache Httpd s modulom `mod_ssl`

[29] `https://www.centos.org/`

[30] `http://www.oracle.com/technetwork/java/javase/downloads/index.html`

[31] `http://tomcat.apache.org/download-80.cgi`

[32] reverse proxy je také nastavenie serverov A a B, kedy server A, na ktorý klient pošle požiadavku neodpovie priamo, ale prepošle túto požiadavku na server B. Ten odpovie serveru A a server A odpoveď prepošle klientovi.

a nakonfigurovaným certifikátom pre HTTPS. Taktiež som nastavil DNS záznam a nakonfiguroval Httpd, takže tento server je dostupný pod adresou `https://testsso.ataccama.com`.

4.3 Inštalácia a nasadenie

V priebehu nasadenia a konfigurácie CASu budem implicitne postupovať a odkazovať na jeho oficiálnu dokumentáciu [29], ak nebude uvedené inak. Inštalácia je v podstate zbildovanie projektu zo zdrojového kódu pomocou systému závislostí Apache Maven[33], ktorý mi tento proces uľahčí. Stiahnem si pripravenú šablonu CAS overlay template[34] pre poslednú stabilnú verziu (master branch), v mojom prípade CAS `5.0.5`. Podľa dokumentácie a na základe mojich cieľov potrebujem doplniť závislosť pre LDAP a JSON Service registry. To dosiahnem tak, že do súboru `pom.xml` pridám medzi `dependencies` tieto riadky:

```
<dependency>
  <groupId>org.apereo.cas</groupId>
  <artifactId>cas-server-support-ldap</artifactId>
  <version>${cas.version}</version>
</dependency>
<dependency>
  <groupId>org.apereo.cas</groupId>
  <artifactId>cas-server-support-json-service-registry
  </artifactId>
  <version>${cas.version}</version>
</dependency>
```

JSON Service registry spravuje pomocou konfiguračných súborov vo formáte JSON[35] zoznam klientov, ktorí môžu využívať služby CASu. Je to z dôvodu bezpečnosti, keďže server môže posielať CAS klientom citlivé informácie o užívateľovi po úspešnom prihlásení. Teraz spustím bildovanie príkazom:

```
sudo ./build.sh package
```

Tento proces stiahne z internetu potrebné súbory a z nich na základe dependencies vytvorí výslednú Java aplikáciu v kompresovanom formáte `.war`, ktorá sa nasadzuje do Tomcatu. Nachádza sa v adresári `target`. Zároveň sa vytvorí adresár pre konfiguráciu `/etc/cas/config` a skopírujú sa tam konfiguračné súbory `application.yml`, `cas.properties` a `log4j2.xml`. V samotnom Tomcate vymažem obsah adresára `webapps`, v ktorom sa implicitne nachádzajú demo aplikácie, ktoré nepotrebujem, a vytvorím tam adresár `ROOT`. Tam rozbalím súbor `cas.war` pomocou programu `unzip`. Samotné rozbalenie dokáže Tomcat robiť automaticky, ale zaberá to viac času. Vďaka umiestneniu

[33] `https://maven.apache.org/`
[34] `https://github.com/apereo/cas-overlay-template`
[35] `http://www.json.org/`

aplikácie do adresára `ROOT` bude CAS server prístupný na adrese `http://testsso.ataccama.com:8080/`. Týmto krokom som nasadil SSO server.

4.4 Konfigurácia

4.4.1 Základné nastavenia

Najprv nakonfigurujem Tomcat a Httpd, aby fungovali v spomínanom reverse proxy nastavení. V serverovej konfigurácii Tomcatu upravím štandardný element `Connector` nasledujúcim spôsobom:

`/opt/tomcat/conf/server.xml`

```
<Connector port="8080" protocol="HTTP/1.1"
  connectionTimeout="20000"
  redirectPort="8443"
  proxyName="testsso.ataccama.com"
  proxyPort="443" scheme="https" secure="true" />
```

Toto nastavenie mi zabezpečí, aby sa Tomcat spustil v proxy móde a všetku komunikáciu smeroval v našom prípade na proxy `testsso.ataccama.com` použitím HTTPS schémy so štandardným portom `443`. Teraz nastavím Httpd, aby všetky požiadavky z `https://testsso.ataccama.com` presmeroval na Tomcat pridaním týchto riadkov do default virtuálneho hosta:

`/etc/httpd/conf.d/ssl.conf`

```
ProxyRequests Off
ProxyPreserveHost On

ProxyPass "/" "http://testsso.ataccama.com:8080/"
ProxyPassReverse "/" "http://testsso.ataccama.com:8080/"

<Proxy *>
        Order deny,allow
        Allow from all
</Proxy>
```

Pre konfiguráciu samotného CASu použijem spomínaný `cas.properties` súbor, ktorý je implicitne načítaný pri každom štarte. Je to ekvivalent k `application.yml` súboru, ktorý tým pádom môžem vymazať. Tu musím zdôrazniť, že tento súbor obsahuje veľmi citlivé informácie ako sú kryptografické kľúče a heslá, takže je nutné nastaviť mu minimálne možné práva, čo je v mojom prípade unixových práv nastavenie `600`, a vlastníkom je `root`. Možností pre konfiguráciu CASu je veľmi veľké množstvo a kompletný zoznam sa nachádza tu [30]. Ja ukážem nastavenia, ktoré sú nutné pre bezpečný chod v našom prostredí. Vysvetlím hlavne kľúčové a netriviálne nastavenia. Funkcia ostatných by mala byť zrejmá z ich názvu. Kompletné konfiguračné súbory však budú súčasťou CD prílohy.

Najprv nastavím cestu ku konfiguračným súborom pre logovanie a service registry pomocou parametra `logging.config` a `cas.serviceRegistry.config.location` respektíve. Z bezpečnostných nastavení je nutné zapnúť šifrovanie (a nastaviť kľúče) ticketov používaných v CAS protokole, Spring Webflowu a TG cookie. Ak CAS pri svojom štarte zistí, že tieto kľúče chýbajú, vygeneruje ich automaticky. Vďaka tomu ich potom stačí skopírovať z logu Tomcatu[36] do konfiguračného súboru. Samozrejme je potom vhodné vymazať tieto logy. Z pochopiteľných dôvodov tieto kľúče neuvediem a nahradím ich znakmi „XXX“.

```
/etc/cas/config/cas.properties

# Configuration files
logging.config=file:/etc/cas/config/log4j2.xml
cas.serviceRegistry.config.location=
        file:/etc/cas/config/services

# Ticket encryption
cas.ticket.security.cipherEnabled=true
cas.ticket.security.encryptionKey=XXX
cas.ticket.security.signingKey=XXX

# Spring webflow encryption
cas.webflow.signing.keySize=512
cas.webflow.signing.key=XXX
cas.webflow.encryption.keySize=16
cas.webflow.encryption.key=XXX

# TG cookie encryption
cas.tgc.signingKey=XXX
cas.tgc.encryptionKey=XXX
cas.tgc.secure=true
```

4.4.2 LDAP

Teraz nakonfigurujem LDAP autentifikáciu v rovnakom súbore. Najprv zadefinujem typ pripojenia. Typ `AD` je pre Active Directory a typ `DIRECT` umožňuje overiť identitu užívateľa priamo jeho prihlasovacími údajmi. To uľahčuje situáciu oproti `AUTHENTICATED`, kde je potrebné prihlásenie technického účtu, prostredníctvom ktorého sa vyhľadávanie v directory vykoná. To však vyžaduje, aby sa všetci užívatelia nachádzali v jednej vetve, čo moja schéma spĺňa, keďže užívatelia sú uložení v `ou=Users,dc=clients,dc=com`. Druhá podmienka je, aby prihlasovacie meno užívateľa bolo súčasťou DN, čo je tiež splnené, pretože DN užívateľa je vo formáte `cn=<mail>,ou=Users,dc=clients,dc=com`.

[36] `/opt/tomcat/logs/catalina.out`

Ďalej je potrebné zadefinovať adresu, kde sa nachádza server. Nastavenie ssl vynucuje šifrované spojenie. BaseDN určuje koreňový uzol vyhľadávania. LDAP filter[37] vyberá entries, ktoré spĺňajú definovanú podmienku. Táto je zapísaná v prefixovej notácii s podobnými logickými operátormi, ako ich poznáme napríklad z jazyka C alebo Java (& je AND, | je OR...) a zátvorky určujú ich prioritu. Vo filtri pre ApacheDS sa nachádza overenie aktivácie účtu užívateľa, aby sa neaktívny účet nemohol prihlásiť. `{user}` ako aj `%s` predstavuje textový reťazec, ktorý užívateľ zadá ako prihlasovacie meno do login formulára CASu.

Potom je tam formát DN, ktorý predstavuje entry užívateľa. V Active Directory konfigurácii sa môže zdať, že je trochu nelogický. To je z toho dôvodu, že AD funguje trochu inak, než bežný LDAP, pretože nevyhľadáva na základe DN, ale sAMAccountName, čo je vlastne unikátny identifikátor užívateľa v celom AD. BindDN a credential sú vlastne prihlasovacie údaje technického užívateľa, pomocou ktorého bude vykonané hľadanie v LDAPe. PrincipalAttributeId a principalAttributePassword sú atribúty v LDAPe, s ktorými sa majú porovnať prihlasovacie údaje zadané užívateľom do prihlasovacieho formulára CASu. Ak je principalAttributePassword prázdny, použije sa štandardná metóda autentifikácie, čo je operácia Bind LDAP protokolu [4]. principalAttributeList je zoznam atribútov, ktoré LDAP server pošle CASu. Niektoré nastavenia som schválne pozmenil z bezpečnostných dôvodov.

```
# Active Directory configuration
cas.authn.ldap[0].type=AD
cas.authn.ldap[0].ldapUrl=
        ldaps://active.directory.server.com:1111
cas.authn.ldap[0].useSsl=true
cas.authn.ldap[0].baseDn=OU=Users,DC=server,DC=com
cas.authn.ldap[0].dnFormat=%s@server.com
cas.authn.ldap[0].userFilter=
        (&(objectclass=user)(sAMAccountName={user}))
cas.authn.ldap[0].bindDn=technicalUser@server.com
cas.authn.ldap[0].bindCredential=password
cas.authn.ldap[0].principalAttributeId=sAMAccountName
cas.authn.ldap[0].principalAttributePassword=
cas.authn.ldap[0].principalAttributeList=sAMAccountName

# User LDAP configuration
cas.authn.ldap[1].type=DIRECT
cas.authn.ldap[1].ldapUrl=ldaps://apacheds.server.com:1111
cas.authn.ldap[1].useSsl=true
cas.authn.ldap[1].baseDn=ou=Users,dc=clients,dc=com
cas.authn.ldap[1].userFilter=
        (&(objectClass=person)(isActive=TRUE)(cn={user}))
cas.authn.ldap[1].dnFormat=cn=%s,ou=Users,dc=clients,dc=com
```

[37] Oficiálna definícia LDAP filtra je prístupná tu: `https://tools.ietf.org/search/rfc4515`

```
cas.authn.ldap[1].principalAttributeId=cn
cas.authn.ldap[1].principalAttributePassword=
cas.authn.ldap[1].principalAttributeList=cn
```

4.4.3 Logovanie

Štandardne je logovanie rozdelené do troch súborov `cas.log`, `cas_audit.log`, `perfStats.log`. Tieto súbory sa za istých podmienok rolujú, čo znamená, že svoj obsah presunú do súborov s definovanou príponou. Vďaka tomu sa najaktuálnejšie informácie nachádzajú v týchto súboroch a archív je tvorený súbormi s príponou podľa dátumu.

Konfiguračný súbor logovania, `log4j2.xml`, implicitne definuje rolovanie na hodinovej bázi s podmienkou rolovania veľkosti 10 MB. To znamená, že logy sa rolujú každú hodinu, (logické) alebo ak dosiahnu veľkosť 10 MB. To mi však nevyhovuje, pretože takéto nastavenie vytvára veľmi veľa malých súborov, ktoré pri veľkom množstve zaťažujú disk. Preto zmením nastavenie rolovania na každý deň o polnoci. Dosiahnem to týmito úpravami[38]:

```
filePattern="/opt/tomcat/logs/cas/cas-%d{yyyy-MM-dd}.log"

<SizeBasedTriggeringPolicy />
<TimeBasedTriggeringPolicy interval="1"/>
```

Z nejakého dôvodu, ktorý sa mi nepodarilo zistiť, prebieha logovanie dvakrát podľa dvoch rôznych konfiguračných súborov. Riešenie tohoto problému je však veľmi jednoduché. Stačí premenovať alebo vymazať súbor:
`/opt/tomcat/webapps/ROOT/WEB-INF/classes/log4j2.xml`

4.4.4 Ostatné

Teraz nastavím klientov, ktorým CAS server umožní autentifikáciu. Vytvorím adresár `/etc/cas/config/services` a v ňom dva JSON súbory `testconfluence-100.json`, a `testjira-200.json`. Do nich vložím konfiguráciu na základe dokumentácie, ktorá nepotrebuje explicitné vysvetlenie, pretože je veľmi jednoduchá. Pre ukážku prikladám konfiguráciu JIRY (oba súbory sa nachádzajú na priloženom CD):

```
/etc/cas/config/services/testjira-200.json

{
  "@class":"org.apereo.cas.services.RegexRegisteredService",
  "serviceId":"^https://testjira.ataccama.com/.*",
  "name":"JIRA",
  "id":200,
  "evaluationOrder":200,
}
```

[38] Uvádzam iba zmenené riadky pre prvý súbor, celá konfigurácia je na priloženom CD.

Ako poslednú vec nakonfigurujem súbor `application.properties`, z ktorého je konfigurácia načítaná ako posledná. To znamená, že prepisuje tú ktorá je definovaná v súbore `cas.properties`. Vymažem implicitného používateľa, ktorý je vhodný na testovacie účely, ale nebezpečný pre produkciu[39]:

```
/opt/tomcat/webapps/ROOT/WEB-INF/classes/application.proper-
ties

cas.authn.accept.users=
```

Teraz, keď už je všetko správne nakonfigurované, môžem server spustiť. Pár užitočných príkazov na ovládanie servera je:

```
# Start CAS server
sudo /opt/tomcat/bin/startup.sh
# Stop CAS server
sudo /opt/tomcat/bin/shutdown.sh
# Main CAS and Tomcat logfile
sudo tail -f /opt/tomcat/logs/catalina.out
```

[39]Umožňuje prihlásenie užívateľa „casuser“ s heslom „Mellon“ do CASu.

Kapitola 5

Konfigurácia CAS klientov

Predtým, než nakonfigurujem JIRU a Confluence tak, aby používali CAS na autentifikáciu užívateľov, nahradím ich interné užívateľské databázy LDAP serverom ApacheDS. Tým dosiahnem, aby mali atlasianie aplikácie po úspešnom prihlasení užívateľa prístup k jeho dátam v LDAPe. Táto konfigurácia nie je vôbec zložitá a je rovnaká pre obe aplikácie s malými rozdielmi. Preto ju stručne zhrniem, pričom zdôrazním kľúčové a rozdielne nastavenia. Podotknem, že túto zmenu, ako aj konfiguráciu CAS klientov na produkčných serveroch je nutné ohlásiť vopred a vykonať ich mimo pracovných hodín, keďže môžu spôsobiť dočasné výpadky. Taktiež je veľmi vhodné zálohovanie týchto systémov. V mojom testovacom prostredí to však nie je potrebné.

Samotnú konfiguráciu začnem prihlásením do administrátorského účtu v JIRE, respektíve Confluence. Atlassian poskytuje podrobnú dokumentáciu k popisovaným krokom aj s obrázkami, pre JIRU napríklad tu [31]. Kliknem na ozubené koliesko vpravo hore, ktoré označuje administračné nastavenia. Z menu vyberiem User Management, vľavo User Directories, a potom dole Add Directory. Vyskočí mi okno pre vyplnenie konfigurácie, ktorá je takmer zhodná s nastavením LDAPu v súbore `cas.properties` v CASe. Jediná vec na ktorú si tu treba dať pozor, je nastavenie užívateľského filtra, kde som pridal podmienku `(belongsTo=cn=JIRA,ou=Groups,dc=clients,dc=com)` a `(belongsTo=cn=Confluence,ou=Groups,dc=clients,dc=com)` respektíve. Potom stačí otestovať spojenie a uložiť nastavenia. Posledným krokom je vypnutie internej databázy tlačítkom Disable.

Pri konfigurácii CAS klientov budem postupovať podľa tohoto [32] návodu. Ich inštalácia je tvorená sadou konfigurácií a pridaním skompilovaných javovských `.jar` súborov, ktoré umožňujú delegáciu autentifikácie na CAS server použitím napríklad CAS protokolu. Najprv nakonfigurujem JIRU. Začnem tým, že upravím jej `web.xml` súbor. Táto úprava je však veľmi veľká a preto som sa rozhodol, že ju tu iba slovne popíšem. Kompletné konfigurácie JIRY aj Confluence sa však nachádzajú na priloženom CD.

Konfigurácia `web.xml` pozostáva z definovania troch filtrov. `CasSingle-`

`SignOutFilter` zabezpečuje odhlasovanie z CASu a tým pádom z celého SSO systému. `CasAuthenticationFilter` zabezpečuje autentifikáciu, teda prihlásenie, užívateľa do SSO systému. `CasValidationFilter` umožňuje užívateľovi prístup na JIRU bez prihlásenia v rámci trvania jeho SSO session. Potom sa tam nachádzajú definície `filter-mapping`, ktoré mapujú tieto filtre na webspace (cesta v linku). Nakoniec je tam `listener`, ktorý by mal zabezpečovať ukončenie lokálnej session užívateľa na JIRE v prípade odhlásenia z CASu. Konfigurácia Confluence je v tomto prípade identická až na zmenené linky.

Ako ďalší krok upravím súbor `seraph-config.xml`. Pre obe aplikácie zakomentujem (alebo vymažem) *value* pre `login.url` a `link.login.url` parametre, a zakomentujem element `authenticator`. Pre JIRU navyše zakomentujem aj *value* `logout.url` parametra. Následne ich nahradím nasledujúcou konfiguráciou[40]. JIRA:

```
<!-- /opt/atlassian/jira/atlassian-jira/WEB-INF/classes/se-
raph-config.xml -->

<param-name>login.url</param-name>
<param-value>
https://testsso.ataccama.com/login?service=${originalurl}
</param-value>

<param-name>link.login.url</param-name>
<param-value>
https://testsso.ataccama.com/login?service=${originalurl}
</param-value>

<param-name>logout.url</param-name>
<param-value>
https://testsso.ataccama.com/logout
</param-value>

<authenticator class="org.jasig.cas.client.integration.at-
lassian.Jira44CasAuthenticator"/>
```

Konfiguračné zmeny pre Confluence:

```
<!-- /opt/atlassian/confluence/confluence/WEB-INF/classes
/seraph-config.xml -->

<param-name>login.url</param-name>
<param-value>
https://testsso.ataccama.com/login?service=${originalurl}
</param-value>

<param-name>link.login.url</param-name>
<param-value>
```

[40] V záujme stručnosti schválne neuvádzam kompletné konfiguračné súbory, pretože z ich kontextu je jasné, kde treba tieto zmeny vykonať.

```
https://testsso.ataccama.com/login?service=${originalurl}
</param-value>

<authenticator class="org.jasig.cas.client.integration.at-
lassian.ConfluenceCasAuthenticator"/>
```

V Confluence je navyše potrebné pre fungovanie single sign-out rozbaliť súbor `WEB-INF/lib/confluence-x.x.x.jar` napríklad pomocou programu `unzip` do nejakého dočasného adresára. Odtiaľ skopírujem súbor `xwork.xml` do `WEB-INF/classes`. V ňom odkomentujem element `result name="success"` a nahradím ho týmto:

```
<result name="success" type="redirect">https://testsso.at-
accama.com/logout</result>
```

Ako poslednú vec je nutné stiahnuť doplňujúce `.jar` súbory, ktoré som už spomínal z Maven repozitára[41] a skopírovať ich do `WEB-INF/lib` adresára oboch aplikácií. Nakoniec ich stačí zreštartovať, aby sa načítala nová konfigurácia.

[41] Core: `https://mvnrepository.com/artifact/org.jasig.cas.client/cas-client-core/3.4.1`
Atlassian Integration: `https://mvnrepository.com/artifact/org.jasig.cas.client/cas-client-integration-atlassian/3.4.1`

Kapitola 6

Testovanie

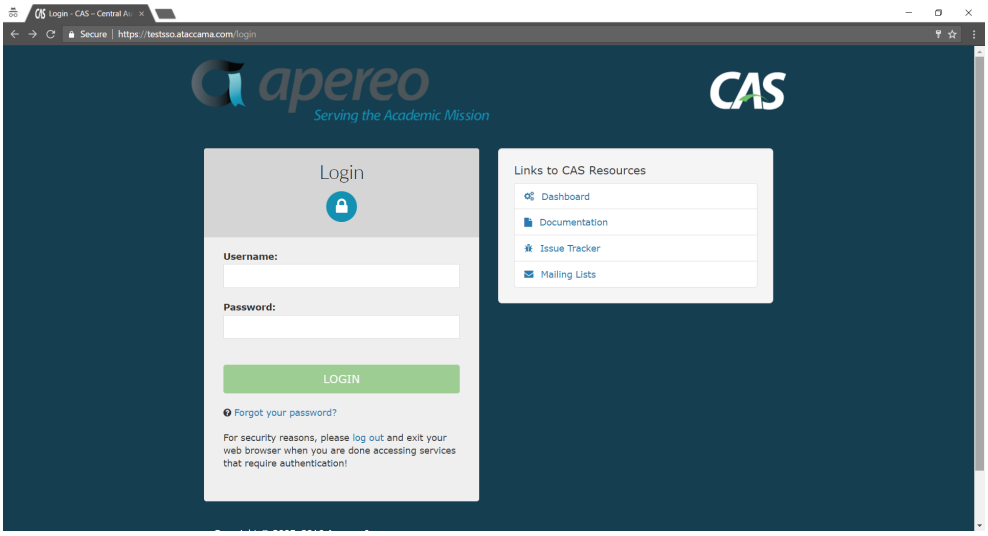
V tejto záverečnej kapitole otestujem nasadený SSO server a jeho klientov, aby som zistil, či všetko funguje podľa očakávania. Mojim cieľom je otestovať základnú funkcionalitu vytvoreného systému z pohľadu užívateľa, ako aj otestovanie bežných administrátorských úkonov. Pozriem sa teda aj na logovanie a to, ako správa LDAPu ovpyvňuje SSO. Predtým však začnem so základnými scenármi, akými sú prihlásenie, odhlásenie a presmerovanie.

Najprv sa skúsim prihlásiť priamo na stránke CAS servera `https://test-sso.ataccama.com`. Pri prístupe na ňu som bol presmerovaný na cestu `/login`, kde sa nachádza formulár na prihlásenie, ktorý je možné vidieť na obrázku 6.1. Tu sa mi úspešne podarilo prihlásiť účtom z Active Directory a takisto klientským testovacím účtom z ApacheDS. V prehliadači sa mi vytvoril cookie s názvom `TGC` pre danú doménu, ktorý obsahuje zašifrovanú hodnotu TGT. Stránku po prihlásení zachytáva obrázok 6.2.

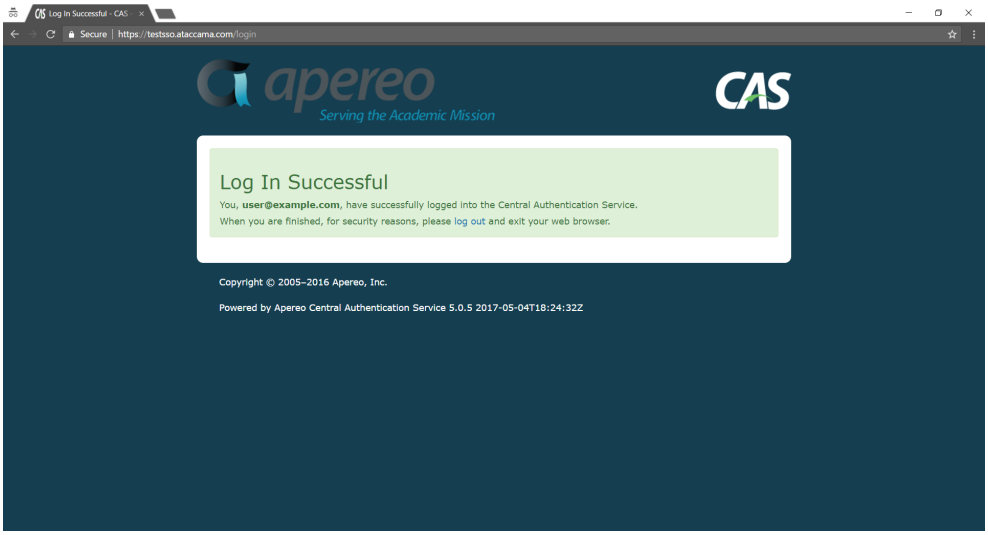
Teraz sa odhlásim a pokúsim sa prihlásiť do JIRY. Kliknutím na prihlasovacie tlačítko som automaticky presmerovaný na CAS server s cestou `/login` a query stringom[42] `service=https%3A%2F%2Ftestjira.ataccama.com%2Fdefault.jsp`. V tomto query stringu je uložená URL, kam CAS server po úspešnej autentifikácii presmeruje užívateľa. Na obrázku 6.3 je zachytená séria týchto presmerovaní, vrátane presmerovania z CASu na JIRU s ticketom v URL. Zároveň je táto URL overená voči nakonfigurovanému zoznamu CAS klientov. V prípade, že sa tam nenachádza, CAS presmeruje užívateľa na chybovú stránku. Po prihlásení som presmerovaný na JIRU a po otvorení Confluence som do nej automaticky prihlásený.

Pri testovaní odhlasovania som však narazil na jeden problém. CAS by sa mal pri odhlásení užívateľa z jednej aplikácie automaticky postarať o ukončenie jeho session vo všetkých ostatných. Avšak v prípade, že sa odhlásim z Confluence, session v JIRE je stále aktívna. Naopak to však funguje správne. Nedokázal som prísť na to, kde je problém, ani ho nijakým rozumným spôsobom

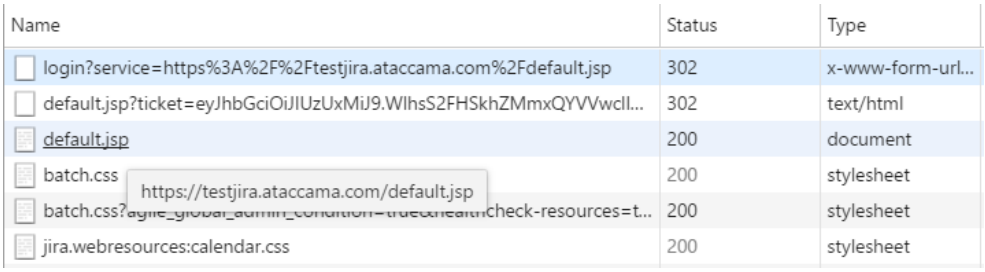
[42] Dáta posielané serveru pomocou URL.



Obr. 6.1: CAS prihlasovací formulár.



Obr. 6.2: CAS úspešné prihlásenie.



Obr. 6.3: Presmerovania pri prihlásení do JIRY.

vyriešiť. Pri odhlasovaní taktiež CAS implicitne nevykonáva presmerovanie naspäť na daného klienta. Toto správanie sa však dá zapnúť pridaním tejto konfigurácie do `cas.properties`:

```
cas.logout.followServiceRedirects=true
```

Teraz sa pozriem na logovanie. Logy sa zapisujú do nakonfigurovaného adresára a rolovanie taktiež funguje podľa očakávania. Logy CASu sú rozdelené do troch súborov. Do prvého `cas.log` sa zapisujú všetky informácie o CASe, ako napríklad počet načítaných konfigurácií klientov, informácie spojené s LDAPom, prihlásenia a odhlásenia užívateľov, vytváranie ST a TGT, a znevalidňovanie týchto ticketov. V druhom súbore `cas_audit.log` sa logujú čisto informácie ohľadne ticketov. V poslednom súbore `perfStats.log` sa nachádzajú informácie o výkone servera a štatistické údaje, ktoré môžu slúžiť pre lepšie vyladenie servera a monitoring.

Ako poslednú vec otestujem správu LDAPu. Užívateľ, ktorý bol pridaný do LDAPu sa môže bez problémov prihlásiť. Pri nastavení atribútu `isActive` v ApacheDS na hodnotu `FALSE` je účet deaktivovaný a nemôže sa autentifikovať. To umožňuje efektívne zablokovanie účtu. Rovnako spoľahlivo fungujú organizačné jednotky Groups pre prístup do aplikácií JIRA a Confluence, a Products filtruje obsah zobrazovanej dokumentácie.

Teraz už ostáva iba dlhodobejšie testovanie s viacerými užívateľmi tohoto riešenia a finálne vylaďovanie detailov ako sú dĺžka trvania session, expirácia ticketov, dlhodobé prihlásenie a customizácia stránok CASu. Ako najvhodnejší ďalší krok k nasadeniu do produkcie sa mi javí interné nasadenie tohoto riešenia pre zamestnancov firmy a odstránenie nedostatkov na základe ich pripomienok.

Záver

Nakoniec by som chcel zhodnotiť celkový výsledok tejto práce. Myslím si, že cieľ sa mi podarilo úspešne splniť a nasadil som SSO riešenie v prostredí firmy Ataccama Software, s.r.o.

Vďaka dôkladnej analýze som vyšetril kľúčové body, ktorých som sa držal v priebehu celej práce. Vybral som riešenie pre správu externých užívateľov použitím protokolu LDAP a definoval som jeho štruktúru. Týmto krokom som dopomohol zjednotiť užívateľské databáze JIRY a Confluence, čím som zabezpečil oveľa jednoduchšiu, a efektívnejšiu správu užívateľov.

Na základe analýzy som potom vybral najvhodnejšiu SSO implementáciu, ktorú som úspešne nasadil a nakonfiguroval. Pomocou skvelej dokumentácie CASu sa mi podarilo veľmi ľahko prepojiť aplikácie JIRA a Confluence s CAS serverom. Vďaka tomu sa zjednotilo prostredie našej firmy a vznikol systém jednotného prihlásenia pre užívateľov. Toto taktiež zabezpečí jednoduché pripojenie nových aplikácii do tohoto prostredia.

Nasadené riešenie vyzerá veľmi sľubne a s vedúcim práce sme sa zhodli, že ho nasadíme do produkcie. Predtým je však potrebné ho ešte dôkladne otestovať. Do budúcna by sme chceli nakonfigurovať automatickú expiráciu užívateľských účtov, zlepšiť single log out a využiť pokročilejšie rysy CASu.

Literatúra

[1] SAS Institute Inc.: *What Is Big Data? | SAS US*. [online] stav z dňa 15.4.2017. Dostupné z: https://www.sas.com/en_us/insights/big-data/what-is-big-data.html

[2] Halevi, G.; Moed, H. F.: *The Evolution of Big Data as a Research and Scientific Topic: Overview of the Literature - Research Trends*. Elsevier B.V., [online] stav z dňa 16.4.2017. Dostupné z: https://www.researchtrends.com/issue-30-september-2012/the-evolution-of-big-data-as-a-research-and-scientific-topic-overview-of-the-literature/

[3] Ndegwa, A.: *What is a Web Application?* MaxCDN.com, [online] stav z dňa 17.4.2017. Dostupné z: https://www.maxcdn.com/one/visual-glossary/web-application/

[4] Zeilenga, K.: *RFC 4510 - Lightweight Directory Access Protocol (LDAP): Technical Specification Road Map*. The Internet Society, [online] stav z dňa 18.4.2017. Dostupné z: https://tools.ietf.org/html/rfc4510

[5] The Apache Software Foundation: *1.2 - Some Background. Directories, directory services and LDAP - Apache Directory*. [online] stav z dňa 18.4.2017. Dostupné z: http://directory.apache.org/apacheds/basic-ug/1.2-some-background.html#directories-and-directory-services

[6] Ruderman, J.: *Same-origin policy - Web security | MDN*. Mozilla Developer Network and individual contributors, [online] stav z dňa 18.4.2017. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/Security/Same-origin_policy#Cross-origin_data_storage_access

[7] Peyrott, S.: *What is and how does Single Sign On Authentication work?* Auth0 Inc., [online] stav z dňa 18.4.2017. Dostupné z: https://auth0.com/blog/what-is-and-how-does-single-sign-on-work/

[8] Amazon Web Services, Inc.: *Cloud Products & Services - Amazon Web Services (AWS)*. [online] stav z dňa 20.4.2017. Dostupné z: https://aws.amazon.com/products

[9] Rouse, M.: *What is Active Directory? - Definition from WhatIs.com*. TechTarget, [online] stav z dňa 20.4.2017. Dostupné z: http://searchwindowsserver.techtarget.com/definition/Active-Directory

[10] Kushi, Y.: *What are the advantages/disadvantages of using single sign-on? - Quora*. Imprivata, Inc., [online] stav z dňa 21.4.2017. Dostupné z: https://www.quora.com/What-are-the-advantages-disadvantages-of-using-single-sign-on

[11] Atlassian: *Crowd - Pricing | Atlassian*. [online] stav z dňa 22.4.2017. Dostupné z: https://www.atlassian.com/software/crowd/pricing

[12] Auth0 Inc.: *Pricing - Auth0*. [online] stav z dňa 22.4.2017. Dostupné z: https://auth0.com/pricing

[13] Atlassian: *Single Sign-on Integration with the Atlassian stack*. [online] stav z dňa 23.4.2017. Dostupné z: https://confluence.atlassian.com/kb/single-sign-on-integration-with-the-atlassian-stack-794495126.html

[14] Apereo, Inc.: *CAS - Architecture*. [online] stav z dňa 24.4.2017. Dostupné z: https://apereo.github.io/cas/5.0.x/planning/Architecture.html

[15] GitHub, Inc.: *GitHub - apereo/java-cas-client: Apereo Java CAS Client*. [online] stav z dňa 24.4.2017. Dostupné z: https://github.com/apereo/java-cas-client#atlassian-integration

[16] Shibboleth: *Shibboleth Consortium - What's Shibboleth*. [online] stav z dňa 24.4.2017. Dostupné z: https://shibboleth.net/about/

[17] Shibboleth: *Shibboleth Consortium - Identity Provider*. [online] stav z dňa 24.4.2017. Dostupné z: https://shibboleth.net/products/identity-provider.html

[18] Shibboleth: *Home - Identity Provider 3 - Shibboleth Wiki*. [online] stav z dňa 24.4.2017. Dostupné z: https://wiki.shibboleth.net/confluence/display/IDP30/Home

[19] Atlassian: *HTTP Authenticator for Confluence | Atlassian Marketplace*. [online] stav z dňa 24.4.2017. Dostupné z: https://marketplace.atlassian.com/plugins/shibauth.confluence.authentication.shibboleth/server/overview

[20] GitHub, Inc.: *GitHub - UW-Madison-DoIT/jiraRemoteUserAuth*. [online] stav z dňa 24.4.2017. Dostupné z: https://github.com/UW-Madison-DoIT/jiraRemoteUserAuth

[21] Atricore Inc.: *JOSSO 2.4*. [online] stav z dňa 24.4.2017. Dostupné z: http://docs.atricore.com/josso2/2.4.0/josso-reference-guide/html/en-US/JOSSO_Reference.html

[22] Atricore Inc.: *JOSSO 2.4 - JIRA Tutorial*. [online] stav z dňa 24.4.2017. Dostupné z: http://docs.atricore.com/josso2/2.4/tutorials/josso-jira-tutorial/docbook/publish/en-US/pdf/josso-jira-tutorial.pdf

[23] Open Fusion Pty. Ltd.: *Open Fusion*. [online] stav z dňa 24.4.2017. Dostupné z: http://www.openfusion.com.au/labs/mod_auth_tkt/

[24] Stani, E.: *Top 4 open source LDAP implementations | Opensource.com*. Red Hat, Inc., [online] stav z dňa 25.4.2017. Dostupné z: https://opensource.com/business/14/5/top-4-open-source-ldap-implementations

[25] The Apache Software Foundation: *1.4.3 - Adding your own partition - Apache Directory*. [online] stav z dňa 1.5.2017. Dostupné z: http://directory.apache.org/apacheds/basic-ug/1.4.3-adding-partition.html#what-are-partitions

[26] Apereo, Inc.: *CAS - CAS Protocol*. [online] stav z dňa 2.5.2017. Dostupné z: https://apereo.github.io/cas/5.0.x/protocol/CAS-Protocol.html

[27] Mazurek, D.: *CAS - CAS Protocol Specification*. Apereo, Inc., [online] stav z dňa 2.5.2017. Dostupné z: https://apereo.github.io/cas/5.0.x/protocol/CAS-Protocol-Specification.html

[28] Apereo, Inc.: *CAS - Security Guide*. [online] stav z dňa 3.5.2017. Dostupné z: https://apereo.github.io/cas/5.0.x/planning/Security-Guide.html

[29] Apereo, Inc.: *CAS - Home*. [online] stav z dňa 3.5.2017. Dostupné z: https://apereo.github.io/cas/5.0.x/index.html

[30] Apereo, Inc.: *CAS Properties*. [online] stav z dňa 7.5.2017. Dostupné z: https://apereo.github.io/cas/5.0.x/installation/Configuration-Properties.html

[31] Atlassian: *Configuring user directories - Atlassian Documentation*. [online] stav z dňa 8.5.2017. Dostupné z: https:

//confluence.atlassian.com/adminjiraserver073/configuring-user-directories-861253197.html

[32] GitHub, Inc.: *GitHub - apereo/java-cas-client: Apereo Java CAS Client.* [online] stav z dňa 8.5.2017. Dostupné z: https://github.com/apereo/java-cas-client#atlassian-integration

Dodatok **A**

Zoznam použitých skratiek

SSO Single sign-on

DMZ Demilitarized zone

AWS Amazon Web Services

LDAP Lightweight directory access protocol

CGI Common gateway interface

CAS Central Authentication Service

JOSSO Java Open Single Sign On

JDK Java development kit

HTTP Hypertext Transfer Protocol

HTTPS Hypertext Transfer Protocol Secure

DNS Domain Name System

DN Distinguished Name

AD Active Directory

URL Uniform Resource Locator

Dodatok **B**

Obsah priloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `BP_Gajdoš_Martin_2017.pdf` ..... Súbor s .pdf verziou bakalárskej práce
- `etc` ..... Adresár s konfiguračnými súbormi pre CAS server a klientov
- `src` ..... adresár so zdrojovými kódmi práce vo formáte LaTeX