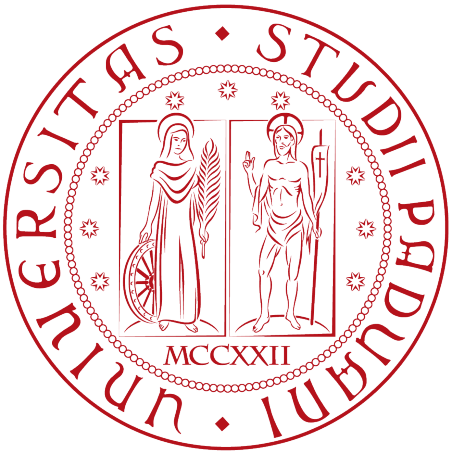
Università degli Studi di Padova

Dipartimento di Matematica "Tullio Levi-Civita"

Corso di Laurea in Informatica



Autenticazione per Zimbra Collaboration Suite (ZCS) tramite protocollo SAML

Tesi di laurea triennale

Relatore

Prof.Tullio Vardanega

Laureando

Francesco De Filippis

Anno Accademico 2019-2020

Ringraziamenti

Innanzitutto, vorrei esprimere la mia gratitudine al Prof. Tullio Vardanega, relatore della mia tesi, per l'aiuto e il sostegno fornitomi durante la stesura del lavoro.

Desidero ringraziare con affetto la mia famiglia per il sostegno, il grande aiuto e per essermi stati vicini in ogni momento durante gli anni di studio.

Ringrazio i miei amici per le esperienze vissute durante questi anni. In particolar modo Andrea, Michele e Riccardo. Ringrazio inoltre le persone con cui ho avuto modo di condividere il percorso universitario.

Vorrei infine ringraziare Zextras per l'opportunità e per la professionalità dimostrata. Ringrazio il team The Lucky Gunslingers per avermi guidato durante tutto il corso del progetto.

Padova, Febbraio 2020 Francesco De Filippis

Sommario

Il presente documento descrive il lavoro svolto durante il periodo di stage, della durata di 304 ore, dal laureando Francesco De Filippis presso l'azienda Zextras S.r.l di Torri di Quartesolo (VI). Gli obiettivi da raggiungere erano molteplici.
La prima funzionalità richiesta dall'azienda era l'autenticazione di un utente presente su Zimbra attraverso l'identity provider Okta, il quale supporta il protocollo SAML. Oltre all'autenticazione per gli utenti già esistenti su Zimbra era richiesto anche il Provisioning. In particolare si trattava di creare un nuovo account su Zimbra al primo tentativo di login dell'utente con conseguente autenticazione. I dati per la creazione dell'utente venivano forniti da Okta attraverso una SAML Assertion. Utilizzando questi dati era richiesta la configurazione dell'account creato.
Il documento è così suddiviso:

- Il primo capitolo descrive l'azienda presso cui ho svolto lo stage. In particolare viene illustrata la sua storia, i suoi prodotti e il modo in cui opera;
- Il secondo capitolo descrive gli obiettivi dello stage in relazione alle aspettative aziendali e personali;
- Il terzo capitolo descrive la scelte progettuali che ho compiuto al fine di proporre una soluzione per soddisfare gli obiettivi prefissati dallo stage;
- Il quarto capitolo presenta una valutazione dello stage in relazione agli obiettivi dell'azienda e all'esperienza da me acquisita nel corso del suo svolgimento.

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

L'azienda

1.1 Profilo aziendale

Zextras s.r.l. nasce a Torri di Quartesolo (VI) nel 2011 come estensione di **Studio Storti s.r.l.** che opera, dal 1997, nel campo delle soluzioni open source. Sin dall'inizio, l'obiettivo principale di questa società era quello di estendere Zimbra *Open Source Edition*, uno dei più diffusi strumenti collaborativi per aziende e pubbliche amministrazioni, aggiungendo nuove funzionalità.
Nel corso degli anni è nata e cresciuta **Zextras Suite**, una raccolta di estensioni che permettono di arricchire Zimbra con nuove funzionalità utili nel suo utilizzo in ambito professionale. Le soluzioni proposte da **Zextras** con i sui prodotti vengono da subito apprezzate da **Synacor**, l'azienda che sviluppa e mantiene Zimbra, la quale decide di includere parte del suo codice nella versione open source. Attualmente i prodotti sviluppati dall'azienda vengono utilizzati da più di 100 milioni di utenti in tutto il mondo.

Figura 1.1: *Open source*

Fonte:medium.com

1.2 Dominio applicativo

1.2.1 Zimbra Open Source Edition

Zextras, come già accennato, è nata con l'obiettivo di creare nuovi contenuti per Zimbra facendone quindi il suo *core business*. Zimbra è un software collaborativo di gruppo adatto a coordinare e supportare l'attività lavorativa di aziende, pubbliche amministrazioni e altri enti. I principali servizi offerti da questo *software* sono i seguenti:

- posta elettronica;
- gestione calendari condivisi e organizzazione eventi;
- interfaccia amministratore;
- supporto dei servizi su dispositivi mobili.

Per estendere l'applicativo con ulteriori funzionalità, sviluppate anche da terze parti, è possibile installare un plug-in che in ambiente Zimbra viene chiamato **Zimlet**.
Esistono due versioni di Zimbra:

- **Zimbra Open Source Edition**: è la versione su cui lavora **Zextras** e offre i servizi elecanti in precedenza;
- **Zimbra Network Edition**: è una versione a pagamento che offre alcune funzionalità closed source tra cui un protocollo per la sincronizzazione di calendario e contatti e maggiori funzionalità per gli amministratori.

1.2.2 Zextras Suite

Zextras Suite è un'insieme di funzionalità che permettono di aggiungere delle funzionalità a Zimbra *Open Source Edition* in modo indipendente da quest'ultimo. Ciò permette una configurazione altamente modulare e personalizzabile in base alle necessità dell'utilizzatore.
Questa *suite* offre i seguenti prodotti:

- **Powerstore**: sistema di ottimizzazione dei dati che permette il risparmio di memoria sui server Zimbra;
- **Backup**: motore di backup in real-time;
- **Admin**: strumenti dedicati agli amministratori per la gestione e il monitoraggio dei servizi attivi sull'istanza di Zimbra;
- **Mobile**: gestione e sincronizzazione di posta elettronica, contatti, eventi e calendario su dispositivi mobili tramite protocolli *Exchange* e *EAS 16.0 (ActiveSync)*;
- **Chat**: piattaforma di messaggistica istantanea nativamente integrata in Zimbra, che permette scambio di messaggi e videochiamate;
- **Drive**: piattaforma per la condivisione di file e l'utilizzo di fogli di lavoro condivisi.

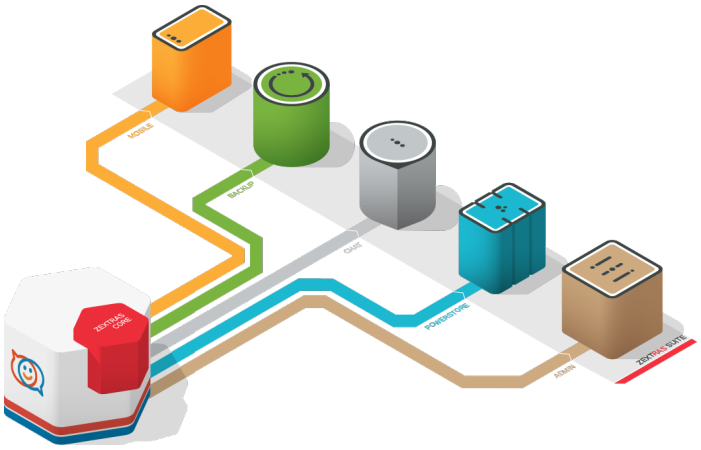


Figura 1.2: Zextras *suite*

Fonte:zimbra-zextras.pl

1.3 Struttura interna

L'azienda è suddivisa in diversi settori specializzati, ciò permette di avere un organico fornito di tutte le competenze necessarie per raggiungere gli obiettivi prefissati. Di seguito un elenco che illustra i diversi reparti:

- **Commercio**: questo reparto, facente parte di **Studio Storti**, si occupa di gestire tutto ciò che riguarda la parte commerciale e *marketing* dei prodotti proposti dall'azienda;
- **System administration**: questo team svolge l'attività di gestione e manutenzione dell'infrastruttura interna all'azienda che ospita numerosi *server*, i quali erogano i servizi offerti da Zimbra ai clienti;
- **Team di sviluppo**: il team di sviluppo è suddiviso in più aree, ognuna delle quali lavora su un aspetto speficifico dei prodotti di **Zextras**:
  - **Front-end**: questa divisione progetta e sviluppa tutto ciò che riguarda l'interfaccia grafica delle applicazioni web;
  - **Back-end**: questo team si occupa di progettare e implementare l'architettura di tutti i servizi presenti nei prodotti dell'azienda. É ulteriormente suddiviso in due team, ciascuno responsabile di un insieme di prodotti diverso;
  - **UI/UX Design**: si occupa di progettare l'interfaccia grafica e di studiare l'user experience delle applicazioni web. Lavora a stretto contatto con la divisione front-end;
  - **Mobile**: progettazione e sviluppo delle applicazioni per dispositivi mobili.

1.4 Processi aziendali

I processi sono un tassello di estrema importanza all'interno di un'azienda che si pone degli obiettivi ben definiti, soprattutto quando questi sono particolarmente ambiziosi e raggiungibili tramite lo sviluppo di prodotti complessi e di grandi dimensioni. Sono inoltre fondamentali qualóra l'azienda avesse al suo interno dei reparti che svolgono mansioni molto differenti tra loro.
Al fine di orchestrare e mettere in funzione tutti i settori dell'azienda affinché questi lavorino in modo coeso, è necessario ricorrere alla definizione e istanziazione di processi, i quali permettono di stabilire con precisione quali sono le attività da svolgere e il modo più strategico per portarle a termine, al fine di raggiungere gli obiettivi prefissati.

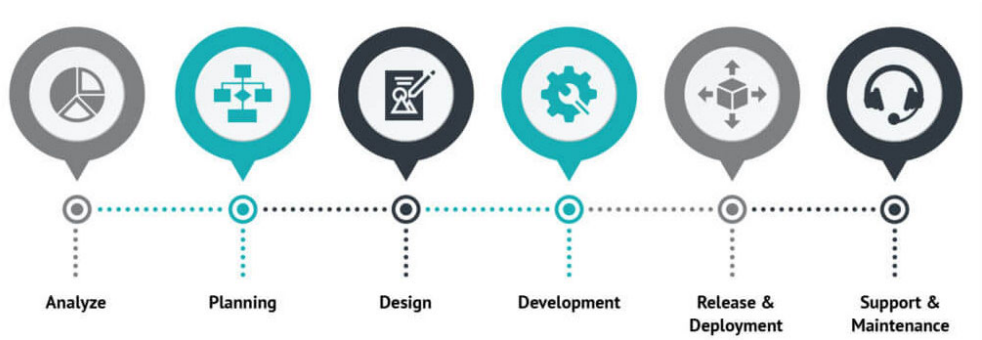


Figura 1.3: Processi

Fonte: keyideasinfotech.com

1.4.1 Fornitura

La grande diffusione di Zimbra in molteplici ambiti, contribuisce alla necessità di nuove funzionalità e al miglioramento di quelle già esistenti. Per questo motivo **Zextras** ottiene spesso nuovi incarichi e nuovi progetti da sviluppare.
La ricerca di nuovi progetti avviene tramite il *CEO* dell'azienda e il *Project Manager*. Oltre all'esigenze dei clienti, nascono dei progetti anche interni dell'azienda stessa che derivano, ad esempio, dalla necessità di migliorare i processi interni con dei nuovi strumenti.
Una volta individuato un possibile progetto, viene fissata una riunione alla quale partecipano i responsabili e il team di sviluppo (formato da gruppi appartenenti a più reparti) che dovrà poi occuparsi del ciclo di vita del nuovo prodotto. Per progetti di grandi dimensioni o particolarmente complessi, la fase di studio di fattibilità dura più tempo e prevede un numero maggiore di riunioni, durante le quali si cerca fin da subito di individuare possibili soluzioni ad alto livello per valutarne la fattibilità.
Dalle riunioni effettuate vengono generati dei verbali, dai quali si estraggono le informazioni più rilevanti che andranno a contribure ai primi contenuti della documentazione. Quando la soluzione viene approvata da azienda e cliente, si procede con le fasi successive.

1.4.2 Comunicazione

Una comunicazione efficiente all'interno del nucleo aziendale è fondamentale per poter essere tutti allineati e coordinati, soprattutto quando è necessario interagire con membri di altri team collocati in altre aree dell'azienda.

Strumenti utilizzati

- **Posta elettronica**: servizio offerto tramite la posta elettronica di Zimbra;
- **Team**: piattaforma di messaggistica istantanea integrata in Zimbra sviluppata da **Zextras**;
- **Drive**: piattaforma per la condivisione di documenti su *server* sviluppata dall'azienda;
- **Riunioni**: per discussioni con tutti i membri (o una parte) del team, è possibile indire delle brevi riunioni.

1.4.3 Metodologia di sviluppo

Per sostenere lo sviluppo di software complessi e dalle dimensioni sostanziose, è necessario avere a disposizione una certa flessibilità ed essere pronti a possibili cambiamenti in termini di requisiti, data anche la tiplogia di servizi che l'azienda sviluppa. Per questo motivo **Zextras** adotta una filosofia *agile*, preferita rispetto a modelli più rigorosi con l'istanziazione di processi molto forti che portano ad una minore capacità di reagire a cambiamenti imminenti. In particolare, la metodologia di sviluppo adottata in azienda è *Scrum*. In seguito descriverò *Scrum* in relazione a come l'azienda lo attua e da quello che concerne la mia esperienza di stage.
Lo sviluppo di un progetto viene suddiviso in fasi, chiamati *Sprint*, la cui durata può variare da una a quattro settimane. In questo caso, la durata dello *Sprint* era di due settimane. Ciascuno *Sprint* ha uno o più obiettivi da portare a termine entro la fine di esso. Il *framework Scrum* prevede lo svolgimento di alcuni eventi e di seguito saranno descritti quelli effettivamente svolti:

- ***Sprint Planning***: si tratta di una riunione che viene indetta all'inizio di ogni *Sprint*, durante la quale vegono discussi gli obiettivi da raggiungere (*Sprint Goal*). Nello specifico, lo *Scrum Master* ovvero colui che coordina il team sceglie, insieme ai colleghi, i *task* da svolgere nel corso dello *Sprint* e li inserisce nello *Sprint Backlog*. I *task* vengono scelti dal *Product Backlog*, ovvero dalla lista completa di funzionalità da implementare nel prodotto. Vengono inoltre stabilite le ore che ciascun membro del team potrà e dovrà dedicare allo sviluppo, alle riunioni e ad altre attività durante il prossimo *Sprint*;
- ***Daily Scrum***: è una riunione della durata di circa 15 minuti che viene svolta ogni inizio giornata con il team di sviluppo (ed eventuali altre persone coinvolte) chiamato anche *Stand-up meeting*. Duranta questa riunione ciascun membro del team parla di ciò che ha svolto il giorno precedente, se ha incontrato ostacoli nello svolgere i suoi *task* e su quali lavorerà durante la giornata. Per più della metà dello stage, i *daily scrum* del mio team si sono svolti in lingua inglese al fine di migliorare le abilità linguistiche;

- ***Sprint Retrospective***: questo evento prevede una riunione in cui il team esegue una valutazione retrospettiva sull'andamento dello *Sprint* appena concluso. In particolare vengono analizzate le metriche riguardanti il numero di *task* portati a termine e gli obiettivi raggiunti. Durante questa riunione viene inoltre svolta un'altra importante attività, ovvero la discussione delle abitudini del team e la ricerca di nuove *best practice*. Per fare ciò, ricorre allo *Starfish Retrospective*, uno schema in cui è possibile individuare i seguenti punti:
  - ***More of***: elenco delle attività che svolte con più frequenza gioverebbero alla crescita del team e allo sviluppo del prodotto;
  - ***Less of***: elenco delle attività da ridurre;
  - ***Start doing***: nuove attività individuate di recente che potrebbero migliorare il team e il prodotto;
  - ***Keep doing***: attività che è utile continuare a svolgere;
  - ***Stop doing***: attività il cui svolgimento va interrotto poiché non più necessario oppure perché si tratta di *bad practice*.

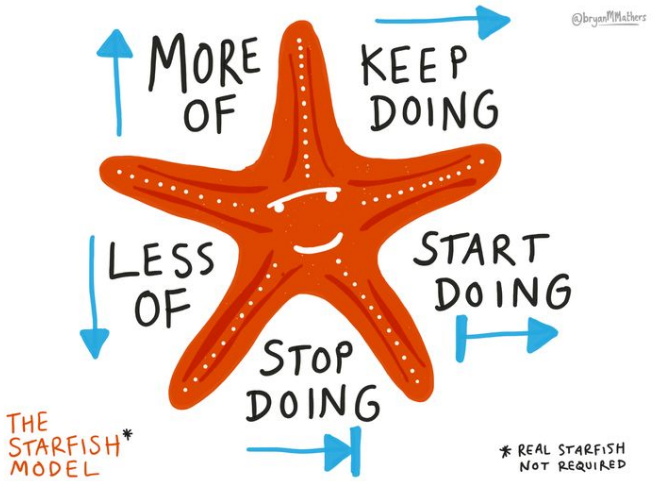


Figura 1.4: *Starfish retrospective*

Fonte: bryanmathers.com

- ***Backlog Refinement***: questa attività, svolta durante lo *Sprint Planning*, consiste nel controllare che tutti gli *item* presenti nel *Product Backlog* siano pronti per essere selezionati e inseriti nello *Sprint Backlog*. Ne vengono rivisti i contenuti, le priorità, le persone assegnate per il loro svolgimento e se necessario, vengono suddivisi in *task* più piccoli o inclusi in altri già esistenti.

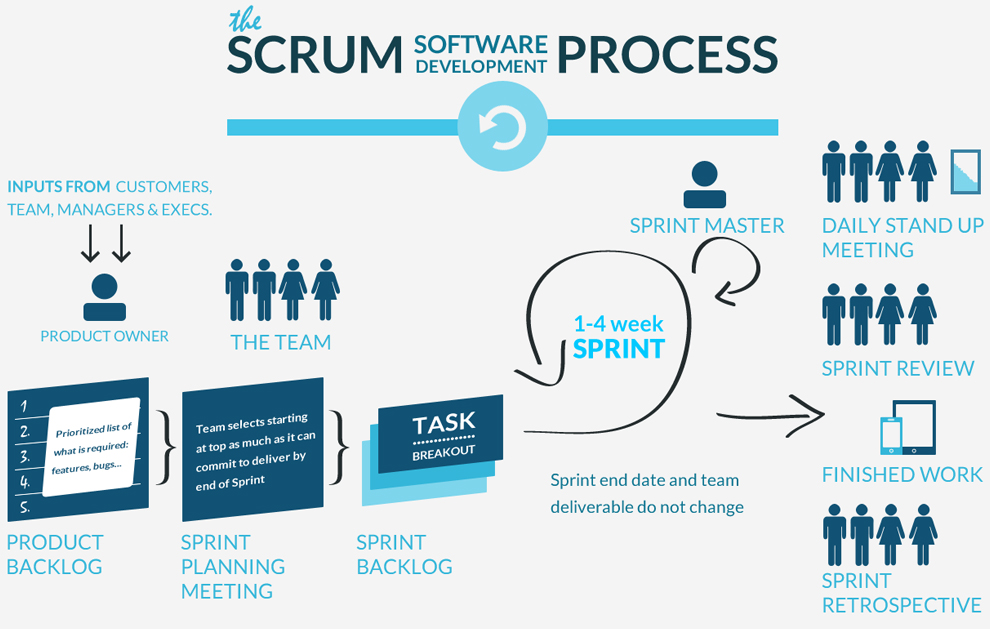


Figura 1.5: Scrum *flow*

Fonte: ICS

1.4.4 Gestione di progetto

La gestione del progetto viene supportata principalmente da due strumenti:

- ***Atlassian Jira***: è un *issue tracking system* molto avanzato che offre la possibilità di gestire tutto ciò che riguarda il *framework Scrum*, quindi gestione degli *Sprint*, dei *task* e molto altro. Da **Jira** è inoltre possibile tracciare le ore impiegate per portare a termine ciascun *task* fornendo, al termine dello *Sprint*, delle serie storiche con i tempi impiegati dai membri del team e il numero di *task* portati a termine rispetto a quelli previsti nello *Sprint Backlog*;
- **Zimbra**: offre, tra i tanti servizi, il calendario per la gestione degli eventi interni all'azienda, per esempio le riunioni e la possibilità di lavorare su documenti condivisi, per esempio i fogli di calcolo utilizzati per il tracciamento delle ore durante lo *Sprint Planning* come descritto nella sezione precedente.

1.4.5 Documentazione

La documentazione di tutti i team di sviluppo è raccolta in un unico punto accessibile a tutti. Per ottenere ciò viene utilizzato il *software* di collaborazione ***Confluence*** sviluppato da ***Atlassian***, il quale offre un editor di tipo *WYSIWYG* che permette di scrivere documenti completi e ben formattati oltre a fornire un sistema di catalogazione molto flessibile e personalizzabile.

1.4.6 Configurazione

Gli strumenti per il versionamento del codice utilizzati sono i seguenti:

- **Git**: sistema di versionamento;
- **Bitbucket**: servizio che ospita i *repository* di tutti i team di sviluppo. Il vantaggio di usare questo strumento è la sua completa integrazione con tutti gli altri strumenti di **Atlassian** impiegati dall'azienda;
- **GitHub**: servizio che ospita i repository *open source* dell'azienda.

Per lo sviluppo di nuove funzionalità viene utilizzato un *workflow* molto simile a ***Gitflow***[1]. Il funzionamento è il seguente:

1. Creazione di un nuovo *branch* sul quale sviluppare una nuova funzionalità;
2. Implementazione nuova funzionalità;
3. *Pull request* per effettuare il *merge* del codice sul *branch master*, nella quale vengono inseriti alcuni membri del team come revisori, al fine di effettuare una *code review* la quale permette di verificare che il codice sia in linea con gli standard di qualità.
4. Dopo aver messo a punto le correzioni segnalate nella *code review* e aver ricevuto l'approvazione dai revisori, avviene il *merge* sul *branch master*.

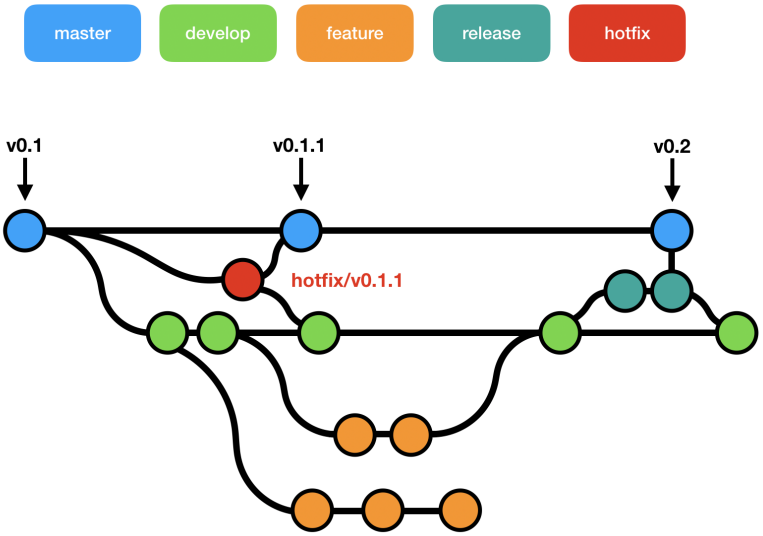


Figura 1.6: *Gitflow*

Fonte: codewall.co.uk

[1] https://danielkummer.github.io/git-flow-cheatsheet/

1.4.7 Verifica

La verifiche che permettono di perseguire la qualità di prodotto avvengono tramite **Jenkins**, un *automation server* che supporta la *continuous integration* e la *continuous delivery*. In questo modo è possibile automatizzare il processo di build ed esecuzione delle varie tipologie di test ad ogni commit sul *repository* e rilasciare quest'ultima in ambiente di produzione. **Jenkins** permette di aggiungere altre funzionalità al processo di build, per esempio l'analisi statica del codice, la rilevazione di alcune tipologie di *bug* e il *code coverage*, rendendolo quindi uno strumento molto flessibile e personalizzabile. Prima della verifica automatizzata vengono effettuate le *code review* come descritto nella sezione precedente.

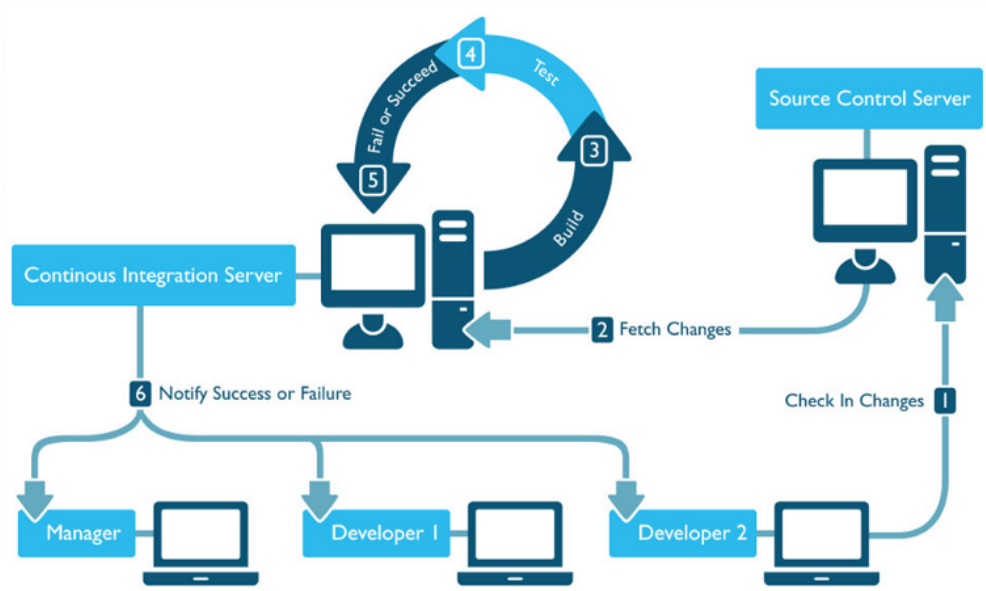


Figura 1.7: *Continuous Integration*

Fonte: developers.redhat.com

Capitolo 2

Obiettivi dello stage

2.1 Vantaggi aziendali

Zextras ospita stage di questa tipologia da ormai diversi anni nonostante sia un'attività che richiede un quantità di tempo e impegno non indifferente poiché, costringe l'azienda a sottrarre risorse dai progetti e dalle attività in corso che, molto spesso, sono vincolate da scadenze. Tuttavia i vantaggi portati da uno strumento come lo stage sono molteplici. Per prima cosa l'azienda ha la possibilità di condurre dei progetti di ricerca che molto spesso coinvolgono l'utilizzo di nuove tecnologie, senza essere costretta a rallentare lo sviluppo dei prodotti ordinari e soprattutto riducendo il rischio di investire troppe risorse in progetti che potenzialmente potrebbero non proseguire.
Alternativamente ha la possibilità affidare ad uno studente lo sviluppo di un prodotto non particolarmente complesso al fine di inserirlo gradualmente nel mondo del lavoro (o nell'organico aziendale) e, allo stesso tempo, far risparmiare tempo al resto del *team*. Un altro aspetto interessante dal punto di vista aziendale è il poter entrare in contatto con studenti che, seppur privi di esperienza lavorativa, sono spesso in grado di proporre soluzioni alternative e creative a problemi comuni. I vantaggi elencati si sono effettivamente concretizzati nel tempo, infatti negli utlimi anni il *team* di sviluppo ha integrato nel suo organico alcuni studenti che, una volta completato lo stage, sono rimasti a contribuire alle sfide tecnologiche che l'azienda affronta giornalmente.
Ciò significa che uno stage ben organizzato e seguito si rivela essere un vero e proprio investimento, non solo per l'azienda che lo ospita, la quale beneficerà degli *output* di questa attività, ma anche per la crescita degli studenti, i quali diventeranno le figure professionali che nel futuro faranno parte dell'intera industria la quale, soprattutto negli ultimi anni, necessita di molte risorse vista la sua dinamicità.

2.2 Presentazione del progetto

Questo progetto nasce dall'esigenza di rendere disponibile un nuovo sistema di autenticazione per la *webmail* di Zimbra, in grado di fornire il giusto connubio tra sicurezza e facilità d'uso. **Zextras** utilizza molti strumenti e servizi al suo interno, sia per lo sviluppo sia per l'amministrazione. Questi utilizzano Okta per la gestione dell'autenticazione, ciò significa che è sufficiente effettuare una sola autenticazione con il proprio account Okta per poter poi accedere a tutti i servizi ad esso collegati.
Questa tipologia di autenticazione si chiama Single Sign-On (SSO) e consiste nell'utilizzo di una singola credenziale che permette di accedere a più servizi.
Zimbra era l'unico serivizio, uno dei più utilizzati in azienda, che non beneficiava di questa tecnica di autenticazione. Per questo motivo **Zextras** prende la decisione di esplorare la possibilità di sviluppare un sistema personalizzato che, oltre all'autenticazione base, avesse le seguenti caratteristiche:

- quando un utente che possiede un *account* su Okta ma non su Zimbra, accede per la prima volta su quest'ultima, viene creato un nuovo account su Zimbra, associato a quello di Okta;
- importazione su Zimbra delle informazioni utente presenti su Okta;
- sincronizzazione dei gruppi di Okta ai quali un utente appartiene, con liste di distribuzione e classi di servizio di Zimbra.

Il fine di questo progetto era quello di uniformare il metodo di autenticazione di Zimbra rispetto a tutti gli servizi utilizzati dall'azienda e soprattutto avere un sistema personalizzato e configurabile che permette di automatizzare alcune operazioni ripetitive e dispendiose in termini di tempo. Per poter proporre una soluzione conforme alle esigenze emerse, ho dovuto per prima cosa condurre uno studio e un'analisi dei protocolli di autenticazione presenti sul mercato. La seconda parte invece consisteva nella progettazione e implementazione del sistema, utilizzando il protocollo ritenuto più idoneo.

Figura 2.1: *Single Sign-On*

Fonte: developer.fourth.com

2.2.1 Analisi stato dell'arte protocolli di autenticazione

La prima attività che ho dovuto svolgere era la ricerca e lo studio dei protocolli di autenticazione più diffusi e utilizzati sul mercato. L'azienda conosceva già alcuni di questi, ma voleva avere un'analisi approfondita e degli esempi del loro utilizzo. Inoltre poteva rivelarsi una valida occasione per scoprire nuovi possibili protocolli adatti all'implementazione del sistema di autenticazione.
L'azienda ha quindi richiesto di redigere un documento che riportasse tutti i risultati delle mie ricerche, in particolare l'analisi dei pro e dei contro di ciascun protocollo e la sua applicabilità al problema da risolvere.

2.2.2 Progettazione di un sistema di autenticazione personalizzato

In seguito all'analisi svolta, il secondo obiettivo dello stage era la progettazione, seguita dall'implementazione, di un sistema di autenticazione per Zimbra utilizzando il protocollo scelto da me, guidato dal *team* di sviluppo. Il sistema di autenticazione doveva essere in grado di:

- utilizzare Okta come identity provider;
- supportare altri identity provider che utilizzato lo stesso protocollo;
- supportare Zimbra tramite il Single Sign-On di Okta rendendo opzionale l'accesso tramite *email* e *password*;
- garantire sicurezza.

Figura 2.2: *Research & Development*

Fonte: youteam.io

2.3 Vincoli

2.3.1 Vincoli metodologici

Per lo svolgimento dell'attività di stage è stato deciso, in comune accordo con il *tutor*, che il modo più efficace per portarla a termine, fosse lavorando presso la sede aziendale. La prima motivazione deriva dalla metodologia di sviluppo adottata, descritta nella sezione §1.4.3, la quale è fortemente incentrata sulla comunicazione e sul confronto frequente con tutti i membri nel *team*. Inoltre, poiché sarei stato affiancato da alcuni *senior developer* si trattava di un'ottima occasione per apprendere il più possibile da professionisti nel settore. Oltre agli incontri di allineamento giornalieri, il *Project Manager* ha stabilito che ci sarebbero stati degli incontri formali, insieme al *tutor* aziendale, per fare il punto della situazione. Nella fase conclusiva del progetto era inoltre prevista una *demo* a cui avrebbero presenziato altre figure aziendali appartententi a diversi reparti. La presenza di figure appartenti a diversi settori e *team* sarebbe state utile a discutere il prodotto sviluppato, sia dal punto di vista funzionale sia da quello infrastrutturale. Inoltre, ricevere un *feedback* da punti di vista diversi è certamente utile per il miglioramento dell'applicazione nel suo insieme.

Figura 2.3: *Teamwork*

Fonte: sandler.com

2.3.2 Vincoli temporali

L'attività di stage prevista aveva una durata di 304 ore, da svolgere nell'arco di due mesi, suddivise in 8 settimane della durata di circa 40 ore ciascuna. L'orario di lavoro accordato con l'azienda era dal Lunedì al Venerdì dalle ore 9.00 alle 18.00.
Prima dell'inizio dello stage ho pianificato, insieme al *tutor*, le attività per ciascuna settimana di lavoro, facendone una stima orario per il loro completamento. Sin dal primo giorno io e il *tutor* aziendale abbiamo rispettato il piano di lavoro stabilito. Tuttavia, in seguito all'attività di analisi svolta, come descritto nella sezione §2.2.1, abbiamo dovuto dedicare più tempo per ottenere un prototipo base funzionante, poiché non era ancora chiaro come utilizzare il protocollo di autenticazione scelto. Nonostante un rallentamento a metà percorso, il resto dello stage ha avuto un andamento lineare che mi ha permesso di terminare il lavoro senza fretta.

Figura 2.4: *Planning*

Fonte: sysaid.com

2.3.3 Vincoli tecnologici

Le tecnologie e i linguaggi utilizzati duarante questo progetto fanno parte dello *stack* tecnologico dell'azienda e sono le seguenti:

- **Java**: essendo il linguaggio con cui è scritto Zimbra, ne consegue che tutta la tecnologia di **Zextras** si sia adeguata, compreso il mio progetto;
- **Git**: sistema di versionamento, come descritto nella sezione §1.4.6;
- **Docker**: è una tecnologia che permette di creare, rilasciare ed eseguire delle applicazioni utilizzando un container. In questo modo l'ambiente racchiuso nel container potrà essere eseguito, tramite *docker*, su macchine diverse che solitamente hanno diverse configurazioni. Questo comportamente è simile alla virtualizzazione e permette di rendere gli ambienti di esecuzione solidi e deterministici. In particolare l'azienda lo utilizza per creare delle istanze di Zimbra utilizzate come ambienti di test in fase di sviluppo. La gestione dei container sono gestiti tramite un servizio di nome ***Portainer***[1].

[1] https://www.portainer.io/

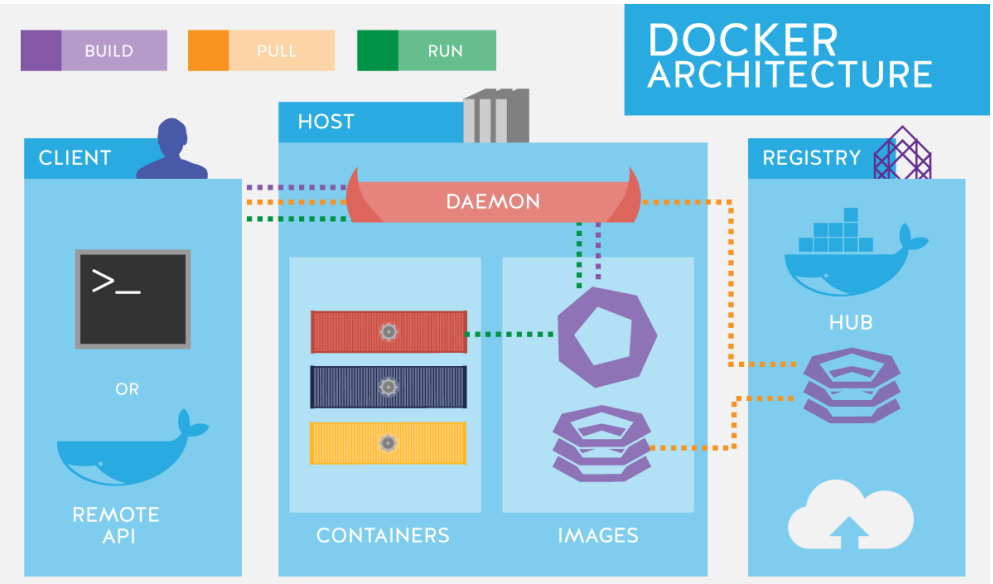


Figura 2.5: *Docker workflow*

Fonte: nordicapis.com

2.4 Aspettative aziendali

Al termine delle 304 ore previste per il completamento dello stage, l'azienda si aspettava di avere almeno l'autenticazione base funzionante che facesse uso del protocollo scelto, in modo da autenticare gli utenti su Zimbra tramite l'identity provider Okta. Come già descritto in precedenza però, la parte interessante di questo progetto era l'aggiunta di alcune funzionalità personalizzate ad un sistema di autenticazione standard. Per questo motivo dopo aver concluso l'attività di ricerca sui protocolli, descritta nel paragrafo §2.2.1, ho discusso, insieme al *team* e al *Project Manager*, i requisiti specifici da implementare. Ciò è stato utile perché al momento della stesura del piano di lavoro, alcuni di essi non potevano essermi chiari a causa della mancanza di contesto riguardante l'ambiente Zimbra. Gli obiettivi stabiliti erano suddivisi in due gruppi. In ordine di priorità: **obbligatori** e **desiderabili**.

Obiettivi obbligatori

- Analisi stato dell'arte dei protocolli di autenticazione più diffusi;
- Implemenazione di un sistema di autenticazione per Zimbra tramite il protocollo scelto;

Obiettivi desiderabili

- *Provisioning*;
- Importazione dei dati dell'utente di Okta su Zimbra;
- Flusso di autenticazione a partire sia dall'identity provider sia da Zimbra;

- Controllo della classe di servizio di Zimbra tramite l'identity provider;
- Gestione delle liste di distribuzione di Zimbra tramite l'identity provider;
- Autenticazione a due fattori;
- Integrazione con *WebAuthn*[2]

2.5 Aspettative personali

Nel corso della laurea triennale ho sempre avuto idea di ciò che volessi fare del punto di vista professionale una volta finito gli studi. Tuttavia, prima di specializzarmi in un determinato ambito, ho deciso di esplorare altre realtà sfruttando l'opportunità di svolgere un'attività stage. Ciò che mi interessava quindi, era osservare in prima persona il modo di lavorare di professionisti, l'utilizzo di tecnologie (sia conosciute in ambito accademico sia per me nuove) a livello professionale e apprendere il più possibile sul mondo del lavoro. Inoltre avevo intezione di trovare un progetto di stage che potesse portare del valore concreto all'azienda e soprattutto avere la possibilità di completarlo nell'arco dei due mesi disponibili, invece di lavorare su prototipi usa e getta. Per poter avere un'idea dell'offerta dell'industria informatica attuale, ho partecipato all'evento *Stage-IT* [3] organizzato dell'Università degli studi di Padova, pensato per mettere in contatto gli studenti con la realtà lavorativa. Durante l'evento ho sostenuto diversi colloqui con aziende che spesso proponevano dei progetti incentrati sull'esplorare una nuova tecnologia ed eventualmente sviluppare un prototipo. Poiché, come già accennato, il mio obiettivo era quello di portare a termine un prodotto finito e utilizzabile, ho continuato la mia ricerca anche dopo lo svolgersi dell'evento. Ho contattato altre aziende e sostenuto altri colloqui finché non ho trovato **Zextras**, che mi ha convinto fin da subito con il progetto proposto e con l'ambiente lavorativo presentato. Quindi i miei obiettivi da raggiungere con un progetto di stage erano:

- lavorare ad un progetto per tutto il suo ciclo di sviluppo. In particolare, partire dalla fase iniziale di ricerca, passare alla fase di progettazione, implementare la soluzione proposta, collaudarlo e vederlo in funzione;
- osservare e provare l'utilizzo di linguaggi a me noti in ambito professionale;
- esplorare una realtà aziendale che proponesse un metodo di lavoro interessante e significativo anche in altri ambiti dello sviluppo *software*;
- lavorare ad un progetto open source.

[2] https://www.w3.org/TR/webauthn-2/
[3] http://informatica.math.unipd.it/laurea/stageit.html

Capitolo 3

Resoconto dello stage

3.1 Descrizione del progetto

Come descritto nella sezione §2.2 il progetto da portare a termine durante questo stage era un sistema di autenticazione personalizzato per Zimbra, che seguisse la tecnica del Single Sign-On.

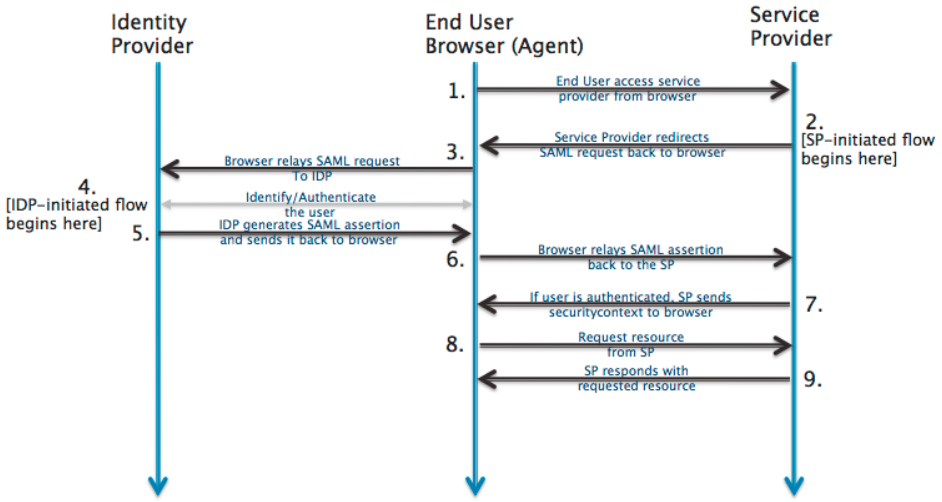


Figura 3.1: *Single Sign-On flow*

Fonte: developer.okta.com

Nel dettaglio, la attività che dovevo svolgere per questo progetto erano diverse. Per prima cosa dovevo trovare un protocollo di autenticazione che fosse adeguato alle esigenze dell'azienda e al problema da risolvere. Successivamente, dovevo passare alla progettazione di un sistema di autenticazione di base, che permettesse di effettuare il *login* su Zimbra tramite Okta. Dal punto di vista pratico, all'utente era sufficiente cliccare su un'applicazione Okta presente nel pannello di controllo Okta, oppure disponibile tramite estensione per *browser*.

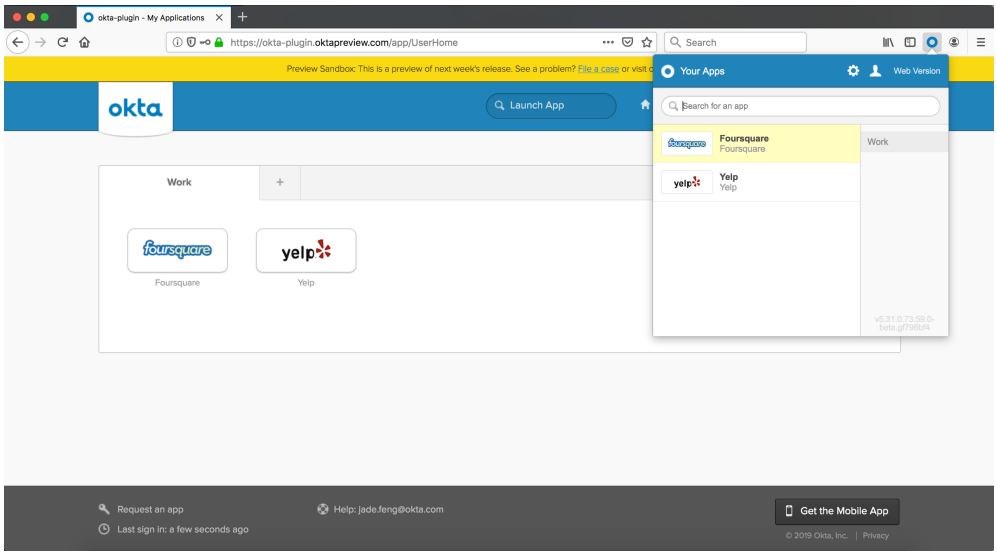


Figura 3.2: *Okta plug-in*

Fonte: addons.mozilla.org

Una volta effettuata l'autenticazione su Okta e premuto il pulsante relativo al *login* per Zimbra, l'utente viene reindirizzato sulla *webmail* di Zimbra. Oltre a questa funzionalità era richiesto di creare un nuovo *account* su Zimbra qualóra l'utente Okta che stesse effettuando un tentativo di accesso ne fosse provvisto e, allo stesso tempo, fare una mappatura tra i gruppi ai quali l'utente appartiene su Okta con le liste di distribuzione e una classe di servizio di Zimbra. Per fare ciò, dovevo progettare un sistema apposito per stabilire come la mappatura dovesse avvenire, dato che le informazioni tra i due sistemi potevano spesso risultare incompatibili.
L'architettura dell'intero sistema doveva essere un'estensione di **Zextras** che sfruttasse l'utilizzo di un *handler http* ovvero un servizio che riceve delle richieste HTTP nelle stesse modalità di una REST API. Questo *handler* doveva gestire lo scambio di informazioni tra Zimbra e Okta, processarle e infine eseguire le operazioni descritte in precedenza.

3.2 Pianificazione

La prima pianificazione delle attività da svolgere è stata fatta da me, insieme al *tutor*, prima dell'inizio dello stage, per poter avere un'idea generale delle tempistiche necessarie al completamento degli obiettivi. Come descritto nella sezione §2.3.2, la pianificazione originaria ha subito delle leggere modifiche, causa necessità di più tempo da dedicare ad alcune attività. Inoltre, pur seguendo il piano di lavoro originale, la pianificazione di ciò che andava fatta ogni settimana veniva confermato una volta completato il lavoro attualmente in corso.
Infatti dopo gli incontri formali fissati dal *Project Manager* in cui dovevo mostrare i progressi tramite una demo, vi erano spesso delle nuove richieste oppure delle modifiche da effettuare a ciò che avevo presentato. Per questo motivo, dopo ciascuno di questi incontri, mi fermavo per discutere con il *team* quanto emerso e decidere come agire di conseguenza. Nella prima fase, descritta nel capitolo §2.2.1, mi sono dedicato allo studio individuale dei protocolli e ho gestito il mio tempo autonomamente. Ogni giorno,

durante il *daily scrum* (§1.4.3), aggiornavo il *team* sui miei progressi e nel resto della giornata mi consultavo con loro in caso di necessità. Dopo questa prima attività era giunto il momento di creare la prima demo che utilizzasse correttamente il protocollo scelto. Questa si è rivelata essere l'operazione che ha consumato più tempo di quanto preventivato poiché, anche il *team* non aveva mai fatto uso diretto di questo protocollo e quindi non poteva darmi delle risposte risposte immediate.
Tuttavia, una volta superato questo ostacolo, il lavoro è proseguito secondo il piano di lavoro originale con alcune accelerazioni nell'implementazione di alcuni requisiti che si sono rivelati essere piuttosto semplici. Nella parte finale dello stage ho avuto inoltre la possibilità di eseguire delle attività fuori pianificazione, infatti ho effettuato alcune modifiche al prodotto successivamente al suo rilascio in ambiente di produzione.

3.3 Analisi

Come già discusso nella sezione §2.2.1, l'attività di ricerca e analisi dei protocolli di autenticazione, è stata la prima ad essere portata a termine. Infatti, nei primi giorni di stage abbiamo discusso con il *team* per stabilirne le modalità. Successivamente abbiamo individuato il primo requisito principale, ovvero l'autenticazione base di un utente su Zimbra tramite l'identity provider Okta. Il resto dei requisiti sono stati analizzati nel dettaglio dopo l'implementazione del primo, poiché ci serviva avere una base di partenza funzionante, al fine di capire cosa si potesse implementare e cosa invece non fosse fattibile.
Dopo aver capito le possibilità che avevamo, abbiamo stabilito che il sistema da progettare doveva essere modulare dal punto di vista degli identity provider supportati e, allo stesso tempo, dovevamo avere la possibilità di implementare dei servizi specifici per quanto riguarda l'integrazione delle configurazioni utente di Zimbra e Okta. Abbiamo dovuto prendere delle decisioni per quanto riguarda questo aspetto. Ad esempio l'associazione tra i gruppi ai quali un utente appartiene su Okta e la classe di servizio a cui il corrispettivo utente Zimbra appartiene poteva essere solo di tipo uno a uno. Per soddisfare un requisito di questo genere ho dovuto pensare ad una soluzione *ad hoc* che richiedeva di fare delle sperimentazioni con un sistema di base funzionante, capace di mettere in atto una comunicazione tra Zimbra e Okta.
I requisiti raccolti in seguito all'attività di analisi sono:

- **Obbligatori**: 14;
- **Desiderabili**: 1;
- **Opzionali**: 2;

Alcuni di questi sono illustrati nella seguente tabella:

Identificativo	Descrizione
R01	Autenticare un utente non esistente su Zimbra tramite Okta, previa creazione dell'account
R05	Durante il processo di creazione di un nuovo utente su Zimbra, aggiungerlo alle giuste liste di distribuzione a seconda dei gruppi ai quali appartiene su Okta
R08	Durante la fase si autenticazione su Zimbra effettuata tramite Okta, aggiornare le liste di distribuzione, se i gruppi ai quali cui l'utente appartiene su Okta sono cambiati
R11	Configurazione del protocollo SAML a livello di dominio
R14	Autenticazione a due fattori

Tabella 3.1: Tabella dei requisiti

3.4 Scelta del protocollo di autenticazione

In seguito all'analisi condotta sui protocolli di autenticazione, già accennata nella sezione §2.2.1, sono emersi due protocolli di interesse, che ho proposto e discusso insieme al *team*. I due protocolli individuati erano SAML e OpenID. Di seguito illustrerò i vantaggi e gli svantaggi di ciascuno.

3.4.1 SAML

Vantaggi

- Fornisce meccanismi di sicurezza nel momento in cui la connessione *HTTPS* non è disponibile o potrebbe essere compromessa;
- Si tratta di uno dei protocolli più utilizzati e affermati sul mercato;
- Se implementato correttamente è molto affidabile e sicuro;
- Il meccanismo chiamato *federated identity* permette di ridurre i costi per la gestione delle identità degli utenti che hanno accesso a servizi di più organizzazioni.

Svantaggi

- Nell'implementazione di alcuni casi d'uso è necessario effettuare dei reindirizzamenti HTTP, che conviene eseguire con una richiesta *POST*. Il problema risiede nel fatto che per poterla fare in automatico è necessario scrivere del codice aggiuntivo con un altro linguaggio;
- Per il problema evidenziato al punto precedente, non è facile implementare questo protocollo su applicazioni native per dispositivi mobili;

- Essendo un protocollo basato su XML, soffre di una tipologia di attacchi chiamati *XML Signature Wrapping*, che potrebbero modificare i documenti XML scambiati tra gli interlocutori del protocollo;
- L'XML-Schema del protocollo è particolarmente complesso.

3.4.2 OpenID

Vantaggi

- I *JSON Web Token (JWT)* utilizzati per rappresentare le informazioni dell'autenticazione sono in un formato portabile e moderno e supportano diversi algoritmi di crittografia;
- Essendo basato su un altro protocollo chiamato OAuth 2.0, il quale funziona su applicazioni native per dispositivi mobili, permette di essere implementato facilmente su di esse;
- Essendo basato su un protocollo di autorizzazione, citato al punto precedente, con OpenID si ha un singolo protocollo per autenticazione e autorizzazione;
- Facile da implementare.

Svantaggi

- Al momento dell'analisi effettuata risultava essere una soluzione poco comune sul mercato, da ciò ne deriva la mancanza di *best practice*;
- *HTTPS* è l'unico livello di criptazione e sicurezza tra il *client* e l'identity provider.

3.4.3 Motivazioni della scelta

Dopo una discussione con il *team* di sviluppo e il *Project Manager*, abbiamo preso la decisione di utilizzare il protocollo SAML. Le ragioni che ci hanno portato a sceglierlo sono le seguenti:

- Essendo il protocollo molto diffuso, vi erano *best practice* note e una comunità di sviluppatori su cui fare affidamento;
- Okta supporta questo protocollo facilitando la comunicazione con un servizio personalizzato che implementa SAML;
- Disponibilità di una libreria scritta in *Java*, il linguaggio utilizzato in azienda, già affermata e utilizzata in altre applicazioni, che implementa le funzionalità base del protocollo. Tale libreria è open source, di fondamentale importanza visto la filosofia aziendale;
- Non vi era necessità di portare questo sistema di autenticazione su dispositivi mobili ma, nel caso in futuro ci fosse la necessità di farlo, ci sarebbero delle soluzioni valide per risolvere il problema descritto nella sezione §3.4.1.

3.4.4 Il protocollo SAML

Attori SAML

Il protocollo SAML coinvolge generalmente tre attori:

- ***Service provider***: è l'entità che fornisce il servizio all'utente, solitamente un sito *web* o una risorsa disponibile su di esso. In questo progetto è la *webmail* di Zimbra;
- ***User-agent***: è un software permette la fruizione dei contenuti ad un utente. Di solito un *web browser* che renderizza i contenuti delle pagine *web*;
- ***Identity Provider***: è l'autorità che certifica l'identità di un utente. In questo progetto questo ruolo è assunto da Okta.

Un esempio di come queste tre parti interagiscono è illustrato nella figura §3.1.

Componenti SAML

SAML è costituito da diversi componenti che possono essere combinati tra di loro in diversi modi, a seconda del caso d'uso richiesto dal problema da risolvere. Le principali funzioni di questi componenti sono autenticazione, gestione dell'identità e autorizzazione tra organizzazioni che abbiano stabilito un rapporto di fiducia. I principali "concetti" del protocollo sono i seguenti:

- ***Assertion***: è un documento in formato XML che contiene le informazioni sull'autenticazione e/o autorizzazione di un utente. Tale documento è solitamente generato dall'identity provider e inviato al service provider;
- ***Protocols***: sono dei meccanismi per fare in modo che le richieste e le risposte SAML siano compatibili tra loro;
- ***Bindings***: definiscono il metodo di comunicazione a basso livello utilizzato dai partecipanti del protocollo. Per esempio permettono di specificare la comunicazione avviene tramite richieste HTTP o SOAP;
- ***Profiles***: i profili definiscono una particolare configurazione di *assertion*, *protocols* e *bindings* che serve per soddisfare un caso d'uso specifico;
- ***Metadata***: è un documento XML che serve per condividere delle configurazioni tra due parti. Per esempio il metodo di crittografia delle asserzioni o di un insieme di attributi;
- ***Authentication Context***: è un meccanismo, definito da un XML-Schema, che serve per descrivere la tipologia di autenticazione impiegata dall'utente quando si autentica con l'identity provider.

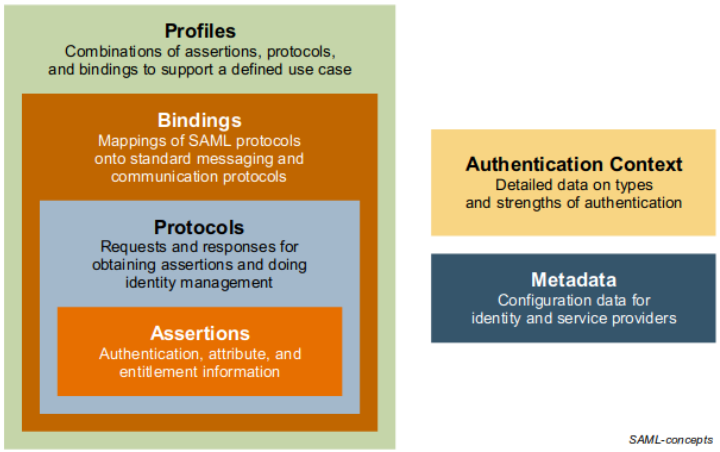


Figura 3.3: *SAML Core concepts*

Fonte: oasis-open.org

3.5 Progettazione

3.5.1 Configurazione applicazione Okta

Un'applicazione Okta è un servizio basato sullo scambio di informazioni tramite REST API tra Okta e un'altro servizio *web*. Sapendo che Okta supportava il protocollo SAML e che in azienda erano state già create altre applicazioni Okta per integrazioni con gli strumenti utilizzati internamente, abbiamo stabilito che la cosa migliore da fare per comunicare con l'identity provider fosse quella di creare una SAML *application* che avesse come endpoint l'indirizzo dell'*handler* HTTP descritto nella sezione successiva. Di seguito un esempio di configurazione della SAML *application* di Okta con gli endpoint dell'*handler* HTTP.

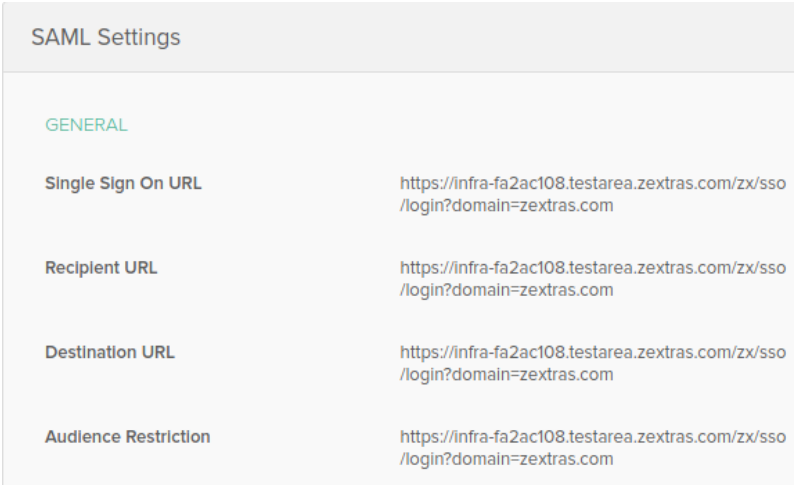
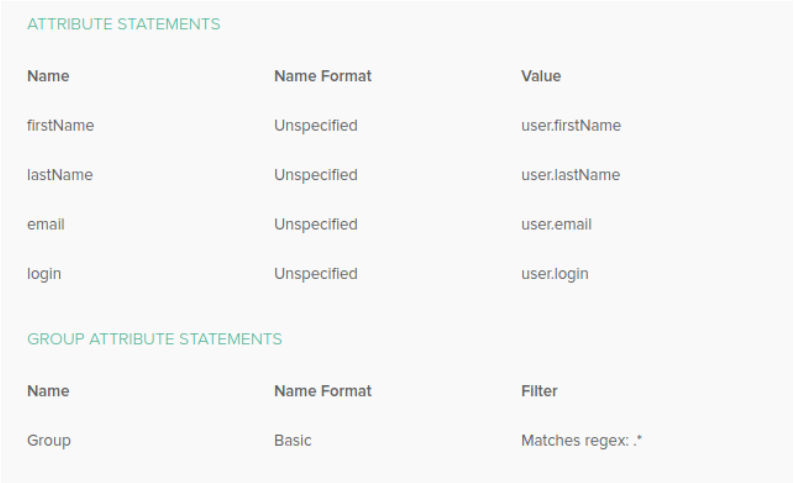


Figura 3.4: *SAML endpoints*

Essendo questa un'applicazione specifica per SAML, era possibile specificare quali attributi dell'utente inviare all'*handler* HTTP (per esempio l'anagrafica e i gruppi ai quali appartiene su Okta) per avere la possibilità non solo di autenticarlo, ma anche di creare il nuovo *account* su Zimbra come già speficificato nella sezione §2.2. Tali informazioni vengono inviate all'*handler* HTTP tramite una SAML Assertion. Di seguito un esempio di configurazione:



ATTRIBUTE STATEMENTS

Name	Name Format	Value
firstName	Unspecified	user.firstName
lastName	Unspecified	user.lastName
email	Unspecified	user.email
login	Unspecified	user.login

GROUP ATTRIBUTE STATEMENTS

Name	Name Format	Filter
Group	Basic	Matches regex: .*

Figura 3.5: Attributi *SAML*

3.5.2 Progettazione handler HTTP

I servizi *web* di **Zextras** sono gestiti tramite degli *handler* HTTP, ovvero delle classi *Java* che restano in ascolto su uno o più endpoint tramite i quali ricevono dei dati. Questi vengono elaborati e viene generata una risposta che può ridursi ad un'azione eseguita, un messaggio di ritorno oppure un reindirizzamento ad una certa pagina *web*. In particolare l'*handler* HTTP che gestiva tutte le operazioni che il mio sistema doveva svolgere, doveva ricevere una *SAML Response*, cioè un documento XML con una SAML Assertion al suo interno. L'asserzione conteneva i dati dell'utente con i quali era possibile:

- Autenticare l'utente;
- Creare il nuovo *account* su Zimbra al primo tentativo di *login*;
- Assegnare l'utente alla giusta classe di servizio e alle liste di distribuzione.

3.5.3 Sistema di mappatura dei gruppi Okta

Una volta giunti i dati all'*handler* HTTP era necessario prendere una decisione su come mappare i gruppi ai quali l'utente apparteva su Okta con una sola classe di servizio e allo stesso tempo più liste di distribuzione di Zimbra. Purtroppo non era possibile eseguire questa operazione in automatico poiché se un utente apparteneva ad un gruppo chiamato `A` su Okta, il sistema avrebbe dovuto inserire l'utente in una classe di servizio chiamata `A`. Ma non era scontata la presenza si una classe di servizio di nome `A` su Zimbra. Il problema è analogo per le liste di distribuzione, in quanto

un gruppo di nome `A` doveva essere mappato sulla lista denominata `A@dominio`, ma l'esistenza di tale dominio su Zimbra non era garantita. Per i motivi sopra elencati, tentare una mappatura automatica avrebbe portato ad errori e comportamenti inattesi. Ho investito un po' di tempo per cercare una soluzione e ho proposto di dare la possibilità all'amministratore di Okta e Zimbra di configurare manualmente le associazioni desiderate tramite un *file* in formato JSON. Nel caso della classe di servizio, la quale è unica per ciascun utente, era possibile effettuare un'associazione uno a uno tra gruppo di Okta e classe di servizio, a patto che questa esistesse su Zimbra. In caso contrario il sistema avrebbe ignorato l'associazione in fase di mappatura. Per quanto riguarda le liste di distribuzione, era possibile associare un gruppo Okta con più di liste di distribuzione presenti su Zimbra. Una volta discussa e approvata dal *team* e dal *Project Manager* ho messo implementato questa soluzione. Di seguito un esempio di mappatura per la classe di servizio e le liste di distribuzione.

```
{
  "zextras": "zextrasCos",
  "developers": "dev"
}
```

Figura 3.6: Mappatura classe di servizio

```
{
    "commercial": ["commercial@fa2ac108.testarea.zextras.com", "commercial@zextras.com"],
    "developers": ["developers@fa2ac108.testarea.zextras.com", "developers@zextras.com"],
    "zextras": ["zextras@fa2ac108.testarea.zextras.com", "zextras@zextras.com"],
    "manager": ["manager@fa2ac108.testarea.zextras.com"],
    "interns": ["interns@fa2ac108.testarea.zextras.com", "interns@zextras.com"]
}
```

Figura 3.7: Mappatura liste di distribuzione

3.5.4 Configurazione Zimbra

La configurazione di tutto il sistema di autenticazione l'ho realizzata attraverso la configurazione di Zimbra, la quale offre la possibilità di definire attributi di diversi tipi, tra cui valori binari (vero o falso), stringhe e oggetti JSON. Le mappature descritte nella sezione precedente e illustrate nelle figure §3.6 e §3.7 erano configurate attraverso due attributi specifici. Oltre a questi ne ho definiti altri, tra cui:

- Configurazione del protocollo SAML (informazioni riguardanti identity provider e service provider);
- Attivazione e disattivazione della creazione automatica di un nuovo *account* su Zimbra;

- Attivazione e disattivazione della sincronizzazione dei gruppi Okta ai quali appartiene un utente, con classe di servizio e liste di distribuzione ad ogni accesso tramite SAML.

Questi attributi vengono letti dall'*handler* HTTP ogni volta che il sistema di autenticazione riceve una richiesta.

3.6 Sviluppo

In questa sezione illustrerò alcuni dettagli riguardanti all'implementazione, in particolare alcuni ostacoli che ho dovuto superare per proseguire con lo sviluppo.

3.6.1 Parsing SAML assertion

Per poter estrapolare i dati in arrivo dalla *SAML Response* di Okta era necessario avere un modo per poter effettuare il parsing del documento XML. Tuttavia non c'era tempo a sufficienza per poter progettare e implementare da zero, un *parser* in grado di gestire i documenti XML generati da SAML, poiché questi hanno una struttura molto complessa.

```
<samlp:Response xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol" xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertio
  <saml:Issuer>http://idp.example.com/metadata.php</saml:Issuer>
  <samlp:Status>
    <samlp:StatusCode Value="urn:oasis:names:tc:SAML:2.0:status:Success"/>
  </samlp:Status>
  <saml:Assertion xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xs="http://www.w3.org/2001/XMLSchema"
  [...]
    </saml:AuthnStatement>
    <saml:AttributeStatement>
      <saml:Attribute Name="uid" NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic">
        <saml:AttributeValue xsi:type="xs:string">test</saml:AttributeValue>
      </saml:Attribute>
      <saml:Attribute Name="mail" NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic">
        <saml:AttributeValue xsi:type="xs:string">test@example.com</saml:AttributeValue>
      </saml:Attribute>
      <saml:Attribute Name="eduPersonAffiliation" NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic">
        <saml:AttributeValue xsi:type="xs:string">users</saml:AttributeValue>
        <saml:AttributeValue xsi:type="xs:string">examplerole1</saml:AttributeValue>
      </saml:Attribute>
    </saml:AttributeStatement>
  </saml:Assertion>
</samlp:Response>
```

Figura 3.8: *SAML Response*

Fonte: samltool.com

Per fare ciò ho incluso una libreria open source che, data una *SAML Response*, restituiva una struttura dati contenente i dati necessari, estrapolati dai *tag* `<saml:AttributeValue>` come illustrato in figura §3.8.

3.6.2 Adattamento libreria HTTP

La libreria citata nella precedente sezione ha però introdotto un altro problema. Questa, utilizzava la gestione delle richieste e risposte HTTP tramite le classi della API di *Java*, in particolare utilizzava il *package* `javax.servlet.http` [1]. **Zextras**, invece, utilizzava un framework chiamato *Netty* [2], pertanto c'era una situazione di incompatibilità. Ho quindi dovuto provvedere ad adattare tutti i metodi della libreria affinché fossero compatibili con il framework utilizzato dall'azienda.

3.7 Documentazione

La documentazione ha fatto parte dei miei compiti sin dall'inizio dell'attività di stage, rivelandosi materiale utile per diversi motivi. Per tutto il *team*, soprattutto per coloro che non hanno lavorato direttamente con me, ma che necessitano di sapere come funzionano alcuni aspetti del prodotto (per esempio i sistemisti devono sapere come effettuare le configurazioni sul *server*). Inoltre è di fondamentale importanza per chi ora dovrà effettuare la manutenzione, dato che potrà risalire facilmente a tutto il lavoro da me svolto. Infine mi è stata molto di aiuto durante la presentazione dei miei progressi durante gli incontri con il *Project Manager*.

3.7.1 Fase di ricerca

Durante questa fase ho documentato tutti i risultati dei miei studi in un documento che illustrava tutti i protocolli analizzati, riportando le informazioni tecniche essenziali e discutendone vantaggi e svantaggi per ciascuno di essi. Questo documento è stato poi aggiornato fino al momento in cui, insieme al *team*, abbiamo scelto il protocollo da utilizzare.

3.7.2 Codice

La documentazione del codice è fondamentale per due motivi: spiegare sezioni di codice difficili da comprendere e facilitare la manutenzione a coloro che dovranno farla, soprattutto se verrà fatta da persone diverse dall'autore. Infatti in una *Sprint Retrospective*, in particolare durante la scrittura dello *Starfish Retrospective* (come descritto nella sezione §1.4.3) è emerso che tutto il *team* avrebbe dovuto dedicare più tempo alla documentazione del codice, scrivendola secondo le regole del javadoc.

3.7.3 Manutenzione

Per quanto riguarda la documentazione dell'architettura e della configurazione del prodotto, ho utilizzato ***Confluence***, come descritto nella sezione §1.4.5. Il documento, accessibile a tutti i reparti dell'azienda, era necessario ai sistemisti per poter installare e opportunamente configurare il sistema e al *team*, nel caso in futuro dovesse estenderlo o apportare delle correzioni.

[1] https://javaee.github.io/javaee-spec/javadocs/javax/servlet/http/package-summary.html
[2] https://netty.io/

3.8 Verifica e Validazione

3.8.1 Verifica

Come descritto nella sezione §1.4.7, in azienda abbiamo attuato diverse tecniche e strumenti per verificare che i prodotti sviluppati non contenessero errori. La verifica automatica avviene tramite **Jenkins**, il quale ha una *pipeline* configurata in modo da mettere in funzione il ciclo di continuous integration e continuous delivery ad ogni *commit* sul repository, illustrato nella figura §1.7.

Test

All'interno della *pipeline* della continuous integration vi era l'esecuzione automatica delle varie tipologie di *test* (per esempio test di unità e *test* di integrazione). Poiché il progetto da me sviluppato interagiva con molte parti dell'architettura di **Zextras**, ho scritto solo *test* di integrazione. I *test* sono parte fondamentale dello sviluppo *software* e meritano di essere progettati e mantenuti allo stesso modo del codice di produzione. Per fare ciò, ho utilizzato due strumenti:

- **Mockito**: è un framework che facilita la scrittura dei *test* in *Java*;
- **Guice**: è un framework open source per il linguaggio *Java*, sviluppato da **Google**, che permette di applicare in modo semplice e scalabile la dependency injection.

Di seguito una tabella con alcuni di essi:

Identificativo	**Descrizione**	**Esito**
IT01	Se un utente è inserito in più gruppi Okta mappati con la corrispondente classe di servizio, allora il sistema non assegna nessun classe di servizio all'utente	Superato
IT04	Se la creazione automatica dell'*account* è disabilitata viene mostrata una pagina di errore informativa per l'utente	Superato
IT05	Quando un utente effettua il *login*, vengono aggiornate le liste di distribuzione alle quali appartiene se i suoi gruppi di Okta sono cambiati	Superato
IT07	Se un utente tenta di autenticarsi da Okta per la prima volta, gli viene creato un nuovo *account* su Zimbra, associato al profilo Okta	Superato
IT10	Se l'utente tenta di accedere a Zimbra senza essere autenticato su Okta, viene reindirizzato sulla pagina di *login* di Okta	Superato

Tabella 3.2: Tabella dei test di integrazione

Code review

Le code review rappresentano un altro strumento di verifica, in questo caso non automatica. Ad ogni nuova *pull request* sul repository vengono assegnati dei revisori, con il compito di analizzare il codice scritto da un altro membro del *team*. Se il codice rispetta gli standard di qualità, la *pull request* viene approvata con conseguente *merge* sul *branch* principale del repository, altrimenti l'autore del codice deve risolvere quanto segnalato dai revisori prima di poter sottoporre il codice ad una nuova revisione. Le code review sono molto utili sia per assicurare che il codice scritto sia conforme alle norme di codifica, sia per confrontare la soluzione implementata con il resto del *team*.

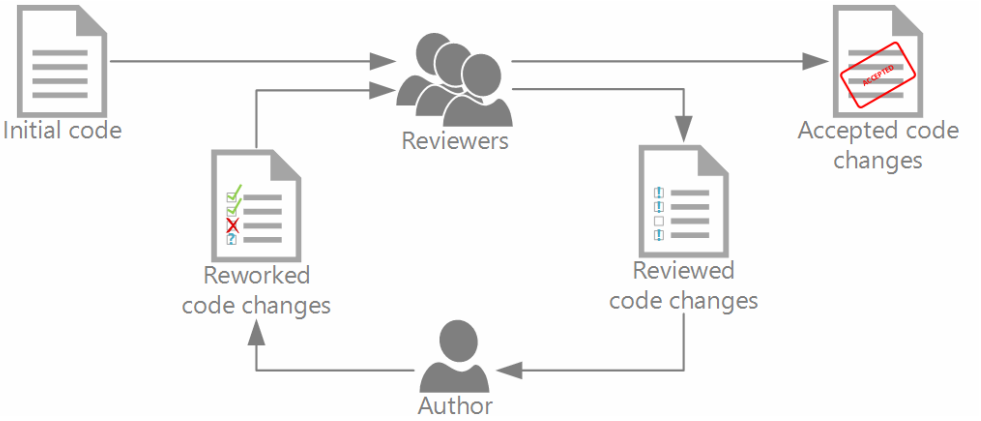


Figura 3.9: *Code review flow*

Fonte: causecode.com

3.8.2 Validazione

Gli incontri formali con il *Project Manager* e alcuni membri del *team* di sviluppo sono serviti fin da subito per controllare lo stato di avanzamento dello sviluppo del prodotto. Infatti già alla dimostrazione della prima demo avevo implementato le funzionalità di base del sistema di autenticazione e ricevuto i primi *feedback*. Da ciò ne conseguiva il tracciamento dei requisiti soddisfatti e di quelli ancora da soddisfare. L'ultimo incontro si è tenuto il giorno 6/12/2019 a cui hanno partecipato:

- Il *CEO* dell'azienda;
- Il *Project Manager*;
- Il responsabile tecnico dell'azienda;
- Il *team* di sviluppo con il *tutor* aziendale;
- Il responsabile di un altro *team*;

Durante questa riunione ho mostrato il sistema di autenticazione in azione e abbiamo discusso alcuni miglioramenti da apportare. In questa sede il responsabile tecnico ha fatto delle nuove richieste, soprattutto per quanto riguarda la configurazione, al fine di

facilitare il lavoro dei sistemisti. Dopo aver messo a punto le ultime sistemazioni e aver accertato che il prodotto fosse conforme alle aspettative iniziali, il giorno 11/12/2019, abbiamo rilasciato il prodotto in produzione per l'utilizzo aziendale.

Capitolo 4

Valutazione retrospettiva

4.1 Soddisfacimento degli obiettivi

Il bilancio degli obiettivi descritti nella sezione §2.4 raggiunti durante lo stage è riassunto nella seguente tabella:

Obiettivo	**Stato**
Analisi stato dell'arte dei protocolli di autenticazione più diffusi	Soddisfatto
Implemenazione di un sistema di autenticazione per Zimbra tramite il protocollo scelto	Soddisfatto
Provisioning (creazione dell'*account* su Zimbra)	Soddisfatto
Importazione dei dati dell'utente di Okta su Zimbra	Soddisfatto
Flusso di autenticazione a partire sia dall'identity provider sia da Zimbra	Soddisfatto
Controllo della classe di servizio di Zimbra tramite l'identity provider	Soddisfatto
Gestione delle liste di distribuzione di Zimbra tramite l'identity provider	Soddisfatto
Autenticazione a due fattori	Soddisfatto
Integrazione con *WebAuthn*[1]	Non Soddisfatto

Tabella 4.1: Tabella degli obiettivi

Come è possibile evincere dalla tabella, ho soddisfatto quasi tutti gli obiettivi prefissati. L'unico obiettivo non soddisfatto è quello relativo all'integrazione con *WebAuthn*[2]. Il motivo deriva dal fatto che nella fase avanzata dello sviluppo, l'azienda non era più interessata ad esplorare questa integrazione. Infatti è proprio per questo motivo che ho avuto il tempo di finire tutto il resto del lavoro senza fretta, permettondomi di rilasciare il prodotto e di effettuare le correzioni successive al collaudo.
Mi ritengo molto soddisfatto dei risultati ottenuti, perché nella fase iniziale di ricerca non pensavo di andare oltre l'implementazione del sistema di autenticazione base. Questo perché mi aspettavo che l'analisi dei protocolli sarebbe durata più del dovuto, lasciandomi solo il tempo necessario a creare un prototipo. Inoltre come già accennato nella sezione §2.5 uno dei miei obiettivi era quello di lavorare ad un progetto per tutto il suo ciclo di vita e che fosse in grado di portare valore all'azienda, la quale è rimasta anch'essa sorpresa e soddisfatta degli obiettivi raggiunti.

Figura 4.1: *Goal*

Fonte: brooksplanning.wordpress.com

[2] https://www.w3.org/TR/webauthn-2/

4.2 Conoscenze e abilità acquisite

Questa esperienza si è rivelata molto positiva anche sotto il punto di vista delle conoscenze e competenze che ho acquisito durante i due mesi di stage. Queste non si limitano solo all'ambito prettamente tecnico ma riguardano anche la realtà aziendale.

Azienda

Ho appreso come un'azienda caratterizza i clienti che utilizzano i suoi prodotti, un aspetto fondamentale dal punto di vista commerciale poiché per creare *software* di successo è necessario che questo abbia dei clienti che lo utilizzino nel tempo. Inoltre preso parte ad un contesto aziendale che porta avanti la filosofia del *software* open source ed è stato molto interessante vederne le dinamiche.

Team di sviluppo

Ho avuto inoltre modo di lavorare con un *team* di sviluppo completo e di poter collaborare anche con altri reparti dell'azienda che si occupavano di aspetti talvolta distanti dallo sviluppo *software*. Ciò mi ha portato a dover impiegare il giusto linguaggio per comunicare con persone che hanno un punto di vista sul prodotto differente dal mio.

Linguaggi e tecnologie

Dal punto di vista tecnico ho acquisito alcune conoscenze di cui mi ritengo soddisfatto. Innanzitutto ho potuto osservare e, in parte, mettere in azione in modo professionale alcune tecniche per l'utlizzo di un linguaggio di programmazione (*Java*) da me conosciuto in ambito accademico.
Inoltre ho compreso i vantaggi e le potenzialità di *Docker*, una tecnologia che negli ultimi anni si è ampiamente diffusa sul mercato.

Strumenti

Ho visto in prima persona come avviene la gestione di un progetto costituito da molte parti e sviluppato da *team* diversi. In particolare l'utilizzo di strumenti di coordinamento come **Jira** per la gestione di progetto e **Confluence** per la gestione della documentazione aziendale.
Inoltre ho appreso molto dagli sviluppatori *senior* del *team*, i quali mi hanno mostrato come sfruttare al meglio e in modo professionale gli strumenti di sviluppo al fine di incrementare la produttività e risparmiare tempo.

Modo di lavorare

Oltre ai metodi di lavoro messi in atto dall'azienda, ho potuto sperimentare come condurre la ricerca e lo studio di nuovi argomenti, nel mio caso i protocolli, e come gestire il tempo a disposizione. Tuttavia su questo aspetto ho capito che devo migliorare il mio metodo in quanto, nonostante abbia raggiunto gli obiettivi, non era adeguato per un'analisi approfondita di ciò che stavo studiando. Infatti dopo aver terminato questa attività ho dovuto rivedere alcuni concetti che non avevo ancora compreso completamente.

4.3 Valutazione personale

Questa esperienza mi ha fatto riflettere sulla relazione tra il mondo accademico e quello del lavoro. Tutto sommato ritengo che le nozioni apprese durante il corso di laurea triennale siano state molto utili durante questa esperienza, seppur non sufficienti per certi aspetti.
Credo che questo meccanismo sia abbastanza naturale poiché l'obiettivo dell'università è quello di erogare conoscenze soprattutto teoriche, necessarie per poter mettere in pratica dei concetti in modo consapevole. Inoltre, essendo il mondo dell'informatica molto ampio, non è pensabile esplorarlo in soli tre anni.
Tuttavia, l'informatica è un campo molto dinamico, cresce e muta anno dopo anno portando alla luce nuovi metodi e tecnologie. Per questo motivo credo che l'offerta didattica erogata dai corsi universitari debba essere adeguatamente aggiornata per poter offrire dei contenuti che siano in linea con ciò che viene utilizzato in ambito professionale.
Inoltre ho notato che molto spesso, quando viene insegnato un linguaggio o una tecnologia, capita che alcune soluzioni vengano etichettate come *bad practice*, quando in realtà esistono dei casi d'uso specifici che vengono soddisfatti tramite l'applicazione di tali metodi. A tal proposito avrei preferito che alcuni corsi proponessero degli esempi meno didattici e più concreti, al fine di rendere l'apprendimento più coinvolgente ed efficace.
Per esempio il modello adottato dal corso di *Ingegneria del Software* potrebbe essere applicato anche agli altri corsi che prevedono un progetto, per fare in modo che lo studente abbia la possibilità di consolidare le conoscenze teoriche con l'ausilio di un progetto significativo, cercando di diminuire il divario tra le conoscenze teoriche e le applicazioni pratiche.
Nel complesso sono comunque soddisfatto di ciò che mi ha dato l'università, poiché mi ha messo di fronte a nozioni e argomenti che da solo non avrei mai esplorato, anche perché spesso molti aspetti teorici vengono ignorati in campo pratico. In questo modo ho compreso che il modo migliore di risolvere problemi, talvolta complessi, sia quello di unire teoria e pratica in modo strategico. Chiaramente dopo questa esperienza sarò più aperto ad una esplorazione più ampia del mondo dell'informatica, anche dopo aver terminato gli studi.
L'attività di stage mi ha pienamente convinto e lo consiglio a tutti coloro che hanno la possibilità di farlo, svolto anche in modalità diverse dalla mia, perché è importante per far capire allo studente cosa aspettarsi dal mondo del lavoro e per avere un'idea della tipologia di azienda in cui vuole lavorare.

Glossario

Agile Approccio allo sviluppo software che pone il focus sul consegnare al cliente un *software* completo, funzionante e di qualità in tempi brevi. 5, 35

API (Application Programming Interface) Nell'ambito dello sviluppo *software* si intende un insieme di procedure, opportunamente organizzate, che risolvono un determinato problema. Spesso si usa questo termine per rifersi alle librerie offerte da un linguaggio di programmazione. 27, 35, 37

Backup Quando si parla di *backup*, si fa riferimento al processo di duplicazione di dati su più supporti (fisici o *cloud*) al fine di poterli recuperare in caso di perdita inattesa. 2, 35

Bug In informatica si tratta di un errore *software* che produce risultati inattesi. 9, 35

Class of Service In Zimbra, una classe di servizio è un identificativo per un gruppo di utenti che condividono un insieme di permessi e proprietà. 16, 18, 19, 24–26, 28, 31, 35

Closed Source Con questo termine si fa riferimento ad un *software* proprietario utilizzabile sotto certe condizioni. Di solito non è possibile modificarlo, condividerlo e ridistribuirlo. 2, 35

Code Review Attività di revisione del codice effttuata da persone diverse dagli autori, al fine di correggere errori, migliorarne la qualità ed eventualmente proporre soluzioni alternative. 8, 9, 29, 35

Container Un *Docker container* è un'unità *software* che contiene un ambiente di esecuzione completo di librerie, dipendenze e configurazioni che è in grado di essere eseguito in modo sicuro e deterministico in altri ambienti *Docker* ospitati su diverse macchine. 14, 35

Continuous Delivery Pratica nell'ambito dell'ingeneria del *software* che consiste nel rilasciare la *build* di un *software* pronta per l'ambiente di produzione. 9, 28, 35

Continuous Integration Pratica nell'ambito dell'ingeneria del *software* che consiste nell'integrazione frequente del lavoro svolto negli ambienti locali degli sviluppatori verso l'ambiente condiviso, ovvero il *repository* in remoto. 9, 28, 35

CRUD Questo acronimo viene spesso usato in ambito di *database management* e indica:

- ***Create***: creazione di un utente;

- ***Read***: richiesta attributi di un utente;
- ***Update***: aggiornamento attributi di un utente;
- ***Delete***: non si parla di una cancellazione vera e propria di un utente ma di *deprovisioning*, ovvero una disabilitazione dell'*account* di quest'ultimo o di un cambio di permessi

. 37

Demo Dimostrazione di in tempo reale di un prodotto, in questo caso di un *software*. 13, 18, 19, 29, 36

Dependency Injection In ingegneria del *software* è una tecnica nella quale un oggetto si occupa di fornire tuttle le dipendenze necessarie ad un altro oggetto. Viene utilizzato per semplificare l'attività di *test* sul *software*. 28, 36

Endpoint L'*endpoint* è un *URL* tramite il quale è possibile raggiungere un servizio. 23, 24, 36

Framework Insieme di strumenti che definiscono la struttura di un sistema a livello concettuale. Nel caso del *software* si può intedendere come un'architettura sulla quale basare lo sviluppo di un prodotto. Quando si parla di modello di sviluppo si intende l'insieme di strumenti teorici che permettono di mettere in atto un concetto specifico. 5, 7, 27, 28, 36

HTTP (Hypertext Transfer Protocol) Protocollo di applicazione alla base dello scambio di informazioni tra *client* e *server* all'interno dei servizi *web*. 18, 20, 22–24, 26, 27, 36, 37

Identity Provider Un *identity provider* è un sistema che crea, mantiene e gestisce le informazioni sull'identità di un utente. Si occupa di fornire il servizio di autenticazione ai service provider. v, 12, 15, 16, 19, 21–23, 25, 31, 37

Issue Tracking System *Software* che permette di gestire in maniera ordinata un insieme di *issue*, ovvero dei task da svolgere. Tipicamente viene utilizzato in ambito collaborativo in quanto permette di tenere traccia delle *issue* portate a termine da tutti i membri del *team*. 7, 36

Javadoc Strumento che permette di generare la documentazione in formato *HTML* per il linguaggio *Java*. 27, 36

JSON (JavaScript Object Notation) É uno *standard* che definisce un formato di *file* che permette di rappresentare oggetti composti da coppie chiave-valore ed *array*. Questo formato è molto utilizzato nello scambio di dati tra *client* e *server* e viene spesso preferito al formato XML, il quale risulta più verboso. 21, 25, 36, 37

Libreria In informatica, una libreria è un insieme di funzioni, metodi e strutture dati che possono essere incluse in un altro modulo *software* rispettando opportune precondizioni. 21, 26, 27, 36

Distribution List In Zimbra, una lista di ditribuzione è un meccanismo simile alla *mailing list* classica disponibile sui *server* di posta elettronica. 16, 18, 20, 24–26, 28, 31, 37

OAuth 2.0 Si tratta di un protocollo di autorizzazione, ovvero permette di autorizzare un utente ad accedere ad una particolare risorsa. 21, 37

Okta Okta è una società di gestione di identità e di accessi, quindi un identity provider. v, 11, 12, 15, 17–26, 28, 31, 37

Open Source Con il termine *open source* si fa riferimento ad un *software* la cui lincenza permette di utilizzarlo, modificarlo e redistribuirlo. 1, 8, 16, 21, 26, 28, 33, 37

OpenID É un protocollo di autenticazione che permette di verificare l'identità di un utente tramite l'ausilio di un identity provider che scambia informazioni con ilservice provider nelle stesse modalità di una REST API. In particolare, le informazioni riguardanti l'autenticazione di un utente, sono gestite tramite un JSON *Web Token*[3]. 20, 21, 37

Parsing In informatica, è un processo che stabilisce se un testo scritto con i simboli di un determinato linguaggio sia conforme ad esso. Questo processo viene eseguito da un *software* chiamato *parser*. 26, 37

Plug-in Componente *software* che aggiunge funzionalità all'applicazione su cui viene installato. 2, 37, 38

Provisioning Con il termine *provisioning* si intende, generalmente, la gestione degli utenti. Questo termine include un insieme di funzionalità riassunte dall'acronimo CRUD. v, 15, 31, 37

Real-time *Software* che opera sotto condizioni temporali ben definite. 2, 37

Repository Nell'ambito dello sviluppo *software* rappresenta un contenitore di codice sorgente, gestito da un sistema di versionamento. 8, 9, 28, 29, 35, 37

REST API Sono una tipologia di API che utilizzano richieste HTTP per lo scambio di informazioni. 18, 23, 37

SAML (Secuity Assertion Markup Language) É un protocollo bastato su XML che permette lo scambio di messaggi per effettuare autenticazione e autorizzazione tra domini distinti. Tipicamente gli attori del procollo sono un identity provider che fornisce l'identità dell'utente da autenticare e un service provider che fornisce il servizio a cui l'utente ha richiesto l'accesso o una risorsa. v, 20–26, 37

SAML Assertion Una asserzione SAML è un documento in formato XML che contiene le informazioni sull'autenticazione e/o autorizzazione di un utente. Tale documento è solitamente generato dall'identity provider e inviato al service provider. v, 24, 37

Service Provider É un sistema che fornisce un servizio a degli utenti. Lo si può intendere come un sito *web* che eroga un certo servizio. 22, 25, 36, 37

[3] https://jwt.io/

SOAP (Simple Object Access Protocol) Protocollo per lo scambio di messaggi tra componenti *software*. L'idea risiede nello strutturare i messaggi secondo il paradigma della programmazione ad oggetti. 22, 38

SSO (Single Sign-On) Si tratta di un sistema di autenticazione che permette ad un utente di effettuare un'unica autenticazione, valida per più servizi e/o risorse che lo supportano. Questo permette all'utente di avere un'unica credenziale valida per più servizi indipendenti. 12, 17

Struttura dati In informatica, è un'entita che viene utilizzata per organizzare un insieme di dati in memoria (*RAM* o di massa). Vengono pesantemente utilizzate per la progettazione di algoritmi efficienti. 26, 38

Task Incarico di piccole dimensioni assegnato al soggetto che dovrà portarlo a termine. 5–7, 36, 38

User experience Il *feedback* manifestato dall'utente nell'interagire con un certo prodotto, sistema o servizio. 3, 38

Workflow Flusso di esecuzione di un insieme di attività. 8, 38

WYSIWYG (What You See Is What You Get) Si riferisce ad una tipologia di *editor* di testo in grado di mostrare in tempo reale, durante la scrittura, quale sarà l'aspetto finale del documento. 7, 38

XML Linguaggio di *Markup* che consente la definizione di metadati. 21, 22, 24, 26, 36–38

XML-Schema Rappresenta la definizione della struttura di un particolare documento XML. Tutti i documenti che utilizzano un certo *schema* devono adeguarsi alle regole in esso definito. 21, 22, 38

Zimbra Collaboration *Software* collaborativo che offre servizi come posta elettronica e calendario condiviso. É possibile estenderlo con nuove funzionalità tramite meccanismi chiamati *Zimlet*, simili al concetto di plug-in. v, 1–5, 7, 11, 12, 14–20, 22, 24, 25, 28, 31, 35, 37, 38

Bibliografia

Siti web consultati

Confluence. URL: https://www.atlassian.com/software/confluence.

Docker. URL: https://opensource.com/resources/what-docker.

Guice. URL: https://github.com/google/guice/wiki.

Javadoc. URL: https://docs.oracle.com/javase/8/docs/technotes/tools/windows/javadoc.html.

Jenkins. URL: https://jenkins.io/.

Jira. URL: https://www.atlassian.com/software/jira.

JSON. URL: https://www.json.org/json-en.html.

Manifesto Agile. URL: https://agilemanifesto.org/.

Mockito. URL: https://site.mockito.org/.

oauth. URL: https://oauth.net/2/.

OpenID. URL: https://openid.net/developers/specs/.

Provisioning. URL: https://support.okta.com/help/s/article/Provisioning-Concepts-and-Methods.

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

SCRUM. URL: https://www.scrum.org/resources/what-is-scrum.

Wikipedia. URL: https://www.wikipedia.org/.

Zextras. URL: https://www.zextras.com/.