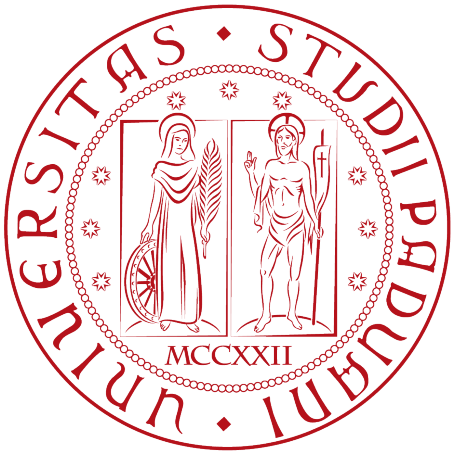
Università degli Studi di Padova

Dipartimento di Matematica

Corso di Laurea in Informatica



Sviluppo di un plug-in cross-platform allo scopo di realizzare il Single Sign-On per un sistema di Identity and Access Management

TESI DI LAUREA TRIENNALE

Relatore:
Prof. Mauro Conti

Laureanda:
Valentina Codogno
Matricola: 1029348

Anno Accademico 2014-2015

Sommario

Il presente documento ha lo scopo di presentare le attività, da me svolte, durante lo stage svolto presso l'azienda Athesys S.r.l di Padova. Vengono, inoltre, illustrati il prodotto finito e le varie tecnologie e strumenti utilizzati per realizzarlo. Lo stage aveva come obiettivo lo sviluppo di un plug-in cross-platform al fine di realizzare il Single Sign-On di applicazioni web presenti all'interno di un sistema di Identity and Access Management.

La tesi, dunque, inizia con un'introduzione in cui viene presentata l'azienda, l'ambito del progetto e il prodotto da realizzare, per poi spaziare con la descrizione dell'attività di analisi dei requisiti in cui vengono illustrati i vari casi d'uso. Successivamente, si passa alla trattazione della parte di progettazione con la spiegazione delle varie componenti realizzate; infine, si ha una presentazione del prodotto software finito e della sua implementazione, per poi concludere con la parte di verifica, validazione e le considerazioni finali.

Convenzioni utilizzate: ogni parola contrassegnata da una 'G' a pedice come la seguente $_G$ è presente nella Sezione 7 (Glossario) a fine documento.

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Introduzione

In questo primo capitolo verrà brevemente descritta l'azienda e la sua metodologia di lavoro rispettivamente nelle sezioni: 1.1 e 1.2; verrà, poi, illustrato in modo dettagliato il progetto aziendale di stage nella Sezione 1.3.

1.1 L'azienda

Athesys S.r.l (`www.athesys.it`), il cui logo è rappresentato in Figura 1.1, è nata nel 2010, inizialmente come azienda di consulenza nell'ambito informatico di Database Management e Business Intelligence. In questi settori Athesys offre numerosi servizi e prodotti che aiutano le aziende nella progettazione e realizzazione dei database più adatti alle esigenze di quest'ultime garantendo sicurezza per i dati e assistenza 24 ore su 24 nel primo caso, mentre nel secondo offre servizi di analisi e modellazione dei dati e strumenti per l'estrazione e la trasformazione di quest'ultimi. Di recente, l'azienda ha deciso di ampliare la propria offerta proponendo lo sviluppo di un sistema di Identity and Access Management che gestisca le identità e i loro accessi in base alla struttura organizzativa delle aziende clienti. La missione di Athesys è quella di garantire al proprio cliente un lavoro ad alto valore professionale con un costante e affidabile supporto; ed è per questo che l'azienda ha rinnovato il proprio certificato ISO9001 dimostrando un controllo maggiore sui propri processi interni.

Figura 1.1: Logo Athesys S.r.l

1.2 Metodologia di lavoro

Il team aziendale utilizza una metodologia di sviluppo $agile_G$ e in particolare quella Scrum [1] descritta in Figura 1.2:

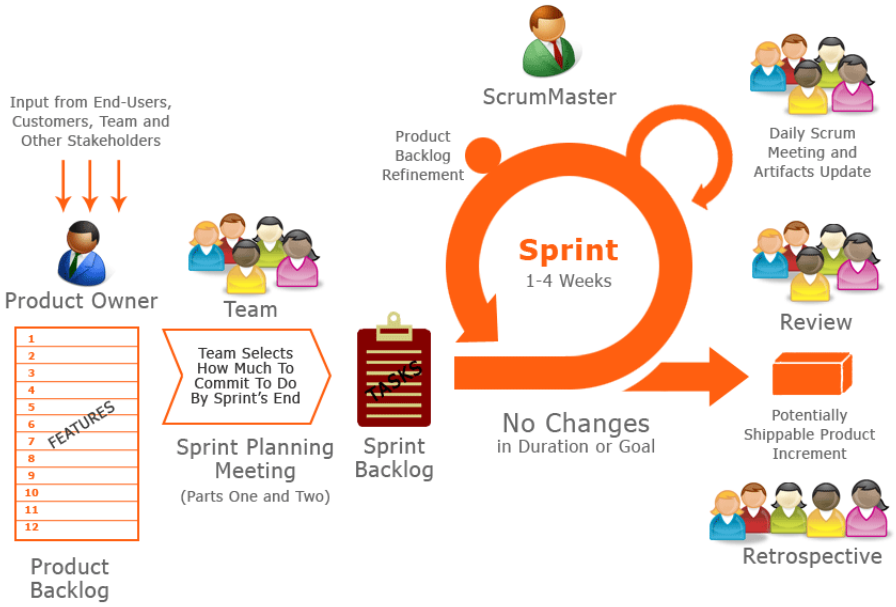


Figura 1.2: Scrum

Tale modello si basa sulla teoria del controllo empirico del processo la quale afferma che la conoscenza deriva da decisioni di esperienza e che la preparazione si basa su ciò che è già noto. Scrum, dunque, prevede un approccio iterativo e incrementale per cercare di ottimizzare la prevedibilità e il controllo dei rischi. Sono tre i principi che sostengono la teoria del controllo empirico del processo:

1. **trasparenza:** i responsabili devono avere una visione degli aspetti più significativi del processo per poter monitorare il risultato del lavoro che sia conforme alle attese;

2. **ispezione:** il lavoro di uno Sprint deve poter essere ispezionato per rilevare variazioni indesiderate rispetto all'obiettivo;

3. **adattamento:** se durante un'ispezione viene riscontrato uno o più aspetti che si discostano dai limiti accettabili e il prodotto risulta inacettabile, il processo in esecuzione deve essere adattato e questo deve avvenire nel minor tempo possibile al fine di evitare ulteriori deviazioni.

Il punto centrale dello Scrum è dato dallo Sprint, esso può essere considerato come un progetto con la durata massima di 4 settimane nelle quali deve essere portato a termine e che, alla sua conclusione, porterà ad un incremento del prodotto. Durante uno sprint non vengono apportate modifiche che potrebbero minarne l'esistenza, inoltre gli obiettivi di qualità non devono diminuire mentre, l'ambito di applicazione del progetto può essere chiarito e rigoneziato tra team di sviluppo e stakeholders. Nel modello Scrum i requisiti, le funzionalità, i miglioramenti e le correzioni sono ordinati all'interno del

Product Backlog; esso non è mai completo, è dinamico ed evolve con il prodotto. Lo Sprint Backlog, invece, è l'insieme degli elementi selezionati dal Product Backlog per lo Sprint; esso è una previsione redatta in sede di riunione dal team di sviluppo con gli obiettivi che deve raggiungere lo Sprint e il lavoro che ne deriva anche in termini di tempo. Lo Sprint Backlog aiuta, quindi, a rendere trasparente il lavoro necessario per raggiungere gli obiettivi dello Sprint.

Athesys adotta Sprint di durata quindicinale, preceduti da riunioni per definire gli obiettivi e i tempi dello Sprint tra i membri del team di sviluppo. Ad ogni Sprint ne segue la sua ispezione e in caso l'adattamento con miglioramenti e correzioni.

1.3 Il progetto

L'obiettivo di questa sezione è la descrizione del progetto aziendale, dell'ambito da cui esso deriva e delle funzionalità del prodotto, da realizzare per l'attività di stage, che costituisce parte di tale progetto.

1.3.1 Ambito del progetto

Il progetto nasce dalla volontà dell'azienda di realizzare un prodotto software nell'ambito dell'Identity and Access Management (IAM) e di presentarsi al consumatore finale e alle aziende con un servizio innovativo.

L'Identity and Access Management, come si può evincere dalla traduzione, si occupa della gestione delle identità dei vari utenti e permette loro l'accesso solamente alle applicazione per cui sono accreditati. In ambito aziendale, l'Identity and Access Management provvede in modo automatico al controllo delle identità virtuali (dipendenti, amministratori, partner etc.) dalla crezione alla cessazione gestendo l'intero ciclo di vita dell'identità dell'utente in base alle politiche aziendali implementate nel sistema in cui, per esempio, certi utenti potrebbere essere accreditati solamente a specifiche applicazioni aziendali imposte dal loro ruolo all'interno dell'azienda. Un tale sistema offre numerosi vantaggi, specie in aziende di medio-grandi dimenesioni: in primis la sicurezza nell'accesso alle varie applicazioni aziendali, solo chi ne ha accesso può accedervi e in secondo luogo i costi in termini di risparmio per il ripristino di password, essendo interamente gestite dal sistema e di tempo non lavorato da parte dei dipendenti:

- *"Risparmio per aggiornamenti dati utente: $ 185 per utente/anno."* - *Gartner Group* [2];
- *"Il tempo perso per le procedure di reset password/utenti costa mediamente alle aziende da 40 a 120 euro."* - *Gartner Group* [2];
- *"Riduzione dei costi di help desk grazie ai sistemi IAM: $ 75 per utente all'anno."* - *Gartner Group* [2].

Vorrei, qui, riportare la chiara ed emblematica definizione di Indentity and Access Management data dalla Gartner Gruop [3]:

> "L'identity and access management è la disciplina di sicurezza che permette ai giusti individui di accedere alle giuste risorse, nei giusti tempi e per le giuste ragioni."

Cos'è il Single Sign-On

Solitamente, un sistema di Identity and Access Management comprende anche la funzionalità di Single Sign-On [4]. Quest'ultimo è un meccanismo per cui un utente di un'applicazione web non deve inserire, ogni volta, login e password per accedere all'applicazione ma, è il sistema che realizza tale funzionalità al suo posto e in modo automatico. L'utente, quindi, con un'unica autenticazione al sistema di Access Management ha accesso a tutte le applicazioni per le quali è accreditato. Il sistema di Access Management gestisce le richieste di autenticazione, basandosi sul fatto che le credenziali dell'utente per una certa applicazione siano già state certificate in precedenza da un sistema di terze parti.

I principiali vantaggi che ne derivano sono:

1. aumento della sicurezza, in quanto non viene richiesto all'utente di farsi carico della gestione sicura dei vari accessi, ma è il sistema che se ne occupa;
2. semplificare la gestione delle varie password di un utente, che non dovrà più ricordarle tutte ma basterà quella di accesso al sistema di Access Management; quindi, sarà meno probabile che esso scelga password semplici da ricordare o la medesima password per più applicazioni;
3. semplificare l'accesso alle varie applicazioni, all'utente basterà un semplice 'click', che in ambito lavorativo si traduce in termini di: risparmio di tempo perso per i lavoratori e aumento della produttività per l'azienda.

1.3.2 Monokee

Monokee (logo in Figura 1.3) è un Identity-as-a-Serivce $(\text{IDaaS})_G$ ideato e sviluppato da Athesys S.r.l per realizzare l'Identity and Access Management.

Figura 1.3: Logo Monokee

Allo stato attuale Monokee comprende la parte di Access Management di applicazioni non federate e applicazioni federate $(\text{SAML})_G$ e ne permette il Single Sign-On; inoltre, esso possiede due cataloghi delle applicazioni più comuni rispettivamente uno per le applicazioni non federate e uno per quelle federate.

Gli utilizzatori di questo sistema possono essere sia i consumatori finali sia i dipendenti, gli amministratori e il responsabile di un'azienda; distinguiamo, quindi, le funzionalità di Monokee in base ai suoi utenti:

- il consumatore finale può visualizzare il proprio application broker, ovvero la dashboard nella quale visualizza tutte le applicazioni per le quali può effettuare il Single Sign-On. Esso, inoltre, dal proprio application broker può aggiungere nuove applicazioni direttamente dai cataloghi di Monokee oppure richiedendone l'aggiunta qualora l'applicazione desiderata non fosse presente nei cataloghi.

- Il responsabile/amministratore dell'azienda, invece, visualizza l'application broker collegato alla propria azienda, ovvero al dominio di quest'ultima. Dall'application broker dell'azienda, esso può gestire tutte le applicazioni utilizzate in ambito aziendale: aggiungerne di nuove e successivamente assegnarle o rimuoverle ai vari dipendenti anche in base al loro ruolo lavorativo.

- Il dipendente, dal proprio application broker aziendale, visualizza le applicazioni aziendali che li sono state assegnate dal responsabile o amministratore dell'azienda. Appena gli viene assegnata un'applicazione il dipendente inserisce delle nuove e proprie credenziali per tale applicazione in accordo con la policy richiesta per la password. Il dipendente, quindi, dal proprio application broker aziendale può solamente effettuare il Single Sign-On delle applicazioni a lui assegnate.

In ambito aziendale, può altresì accadere che un dipendente, responsabile o amministratore sia un lavoratore presso anche un'altra azienda, ovvero può possedere più domini; in questo caso all'autenticazione su Monokee viene data la possibilità di scegliere con quale dominio effettuare le operazioni sopra descritte. Per un livello maggiore di sicurezza, ovvero per impedire l'accesso diretto alle applicazioni tramite Single Sign-On una volta autenticati su Monokee, quest'ultimo prevede l'inserimento di credenziali anche per accedere ai domini, anche in questo caso l'utente le inserirà solamente la prima volta al momento dell'avvenuto assegnamento o della creazione. Naturalmente, è necessario che gli utilizzatori siano registrati e autenticati a Monokee per usufruire delle funzionalità descritte.

Molto più comunemente, invece, esiste la possibilità che una persona abbia un account personale su Monokee e che successivamente la stessa persona venga aggiunta con altri diversi account Monokee da una o più aziende perchè ne è appena stato assunto oppure ne è diventato il responsabile o l'amministatore; Monokee, dunque, per evitare il proliferare di account e la loro ridondanza in quanto, di fatto, appartengono alla stessa persona, adotta la seguente organizzazione ad albero dall'alto verso il basso:

1. esiste un'identità associata ad ogni persona fisica che si è registrata e che Monokee riconosce;

2. ogni identità può avere uno o più account Monokee ad essa associati: un account Monokee personale che ha le stesse credenziali dell'identità e uno o più account Monokee dati dalle aziende;

3. ogni account Monokee aziendale può avere uno o più domini oppure ci potrebbero essere domini slegati dagli account ma che fanno sempre riferimento all'identità;

4. un'applicazione può risiede sia all'interno dell'account personale sia all'interno di uno o più domini.

Spiegato ciò, possono verificarsi scenari particolari che devono essere gestiti, come ad esempio:

- la stessa applicazione è presente in due o più account diversi;
- la stessa applicazione è presente in due o più domini diversi;
- la stessa applicazione è presente in uno stesso dominio ma con un nome diverso.

1.3.3 Prodotto da realizzare

L'obiettivo dello stage è lo sviluppo di un plug-in installabile sui principali browser, quali: Chrome, Firefox, Internet Explorer e Safari. Esso aumenta le funzionalità di quest'ultimi realizzando alcuni dei compiti di Monokee per le sole applicazioni non federate: il primo è l'implementazione del processo di Single Sign-On dell'applicazione desiderata dall'utente collegato su Monokee, il plug-in deve:

1. recuperare da Monokee i dati cifrati relativi alla pagina di login dell'applicazione richiesta, ovvero: l'URL$_G$ della pagina di login, gli identificativi dei campi username e password e del pulsante di submit e infine i valori di login e password per accedere all'applicazione;
2. comunicare con Monokee per ottenere la chiave di decifratura;
3. decifrare i dati della pagina ed effettuare la redirect all'URL$_G$ di login dell'applicazione;
4. compilare i campi della pagina di login con login e password e simulare il click dell'utente sul pulsante di submit.

Con tale processo l'utente, a seguito della scelta dell'applicazione effettuata su Monokee, si ritroverà all'interno di quest'ultima già autenticato. Il secondo è l'implementazione di un wizard di learning per l'apprendimento di nuove login applicative, per rispondere alla necessità degli utenti di poter aggiungere al proprio application broker applicazioni non presenti nel catalogo delle applicazioni non federate di Monokee, il plug-in deve:

1. recuperare l'identificativo e l'URL$_G$ dell'applicazione da Monokee;
2. effettuare la redirect all'URL$_G$ dell'applicazione e chiedere all'utente tramite popup di navigare fino alla pagina di login dell'applicazione;
3. alla conferma da parte dell'utente di essere nella pagina di login, salvare l'URL$_G$ e richiedere, tramite popup, la selezione del campo username della pagina;

4. alla conferma della selezione del campo username, salvare l'identificativo dell'elemento del DOM_G associato al campo e richiedere tramite popup la selezione del campo password;

5. alla conferma da parte dell'utente della selezione del campo password, salvare l'identitificativo dell'elemento del DOM_G associato al campo e richiedere la selezione del pulsante di submit;

6. alla conferma della selezione del submit, salvare l'identificativo dell'elemento del DOM_G associato al submit e comunicare con Monokee per ottenere la chiave di cifratura;

7. criptare tutti i dati ottenuti con il wizard di learning e inviarli a Monokee insieme all'identificativo dell'applicazione;

8. ad esito positivo da parte di Monokee dell'aggiunta dell'applicazione nell'application broker dell'utente effettuare la redirect alla pagina di avvenuta modifica di Monokee, altrimenti visualizzare un messaggio d'errore.

Si noti che la responsabilità del corretto apprendimento di una nuova login applicativa è complementamente a carico dell'utente; il plug-in, per esempio, non può sapere se l'utente sta selezionando il campo password quando lo step del wizard richiede la selezione del campo username o se sta selezionando qualsiasi altro campo di input della pagina, se non per il fatto di evidenziare all'utente il campo che sta selezionando oppure indicargli che, per esempio, non ha effettuato alcuna selezione. Il wizard è uno strumento per aiutare l'utente nell'aggiunta di una nuova applicazione passo per passo; è nel suo interesse eseguirlo correttamente altrimenti successivamente non riuscirà ad effettuare il Single Sign-On di tale applicazione.

Per effettuare le due funzionalità appena descritte, il plug-in deve intercettare rispettivamente l'evento di Single Sign-On e l'evento di learning; questi eventi sono scatenati da Monokee nel momento in cui l'utente, dal proprio application broker, clicca sull'applicazione di cui desidera eseguire il Single Sign-On oppure clicca sul pulsante per l'aggiunta di una nuova applicazione non da catalogo. Inoltre, il plug-in intercetta l'evento di controllo dell'installazione, sempre inviato da Monokee, per il quale il plug-in deve manipolare il DOM_G della pagina con cui è stato inviato l'evento ed impostare a vero un campo hidden. Il plug-in prevede anche funzionalità durante la navigazione al di fuori di Monokee, ossia:

- il plug-in mette a disposizione un pulsante nei vari browser dal quale l'utente, nel caso in cui si trovasse in un'applicazione e decidesse di aggiurla al proprio application broker, cliccando su di esso visualizzerebbe un popup di richiesta di aggiunta di tale applicazione, la quale può essere confermata o annullata. Nel caso di conferma il plug-in si limita a inviare a Monokee l'URL_G dell'applicazione, succesivamente è quest'ultimo che decide quale operazione attuare, controllando per prima cosa se l'utente è, a lui, autenticato e in seguito decidendo, a seconda

dei casi, se inviare un evento di learning o di Single Sign-On oppure richiedere all'utente la registrazione delle credenziali per tale applicazione.

- Ad ogni nuova navigazione, il plug-in controlla se l'attuale URL_G di navigazione corrisponde con uno o più URL_G delle pagine di login della lista completa delle applicazioni a cui l'utente è accreditato. Nel caso di corrispondenza il plug-in presenta all'utente un popup per proporgli di effettuare il Single Sign-On direttamente dall'applicazione stessa, naturalmente l'utente deve essere comunque autenticato su Monokee. In caso di conferma il plug-in procede alla redirect della pagina di Monokee che invia l'evento di Single Sign-On passandogli l'URL_G dell'applicazione.

- In caso di annullamento al procedere con il Single Sign-On direttamente dall'applicazione, potrebbe significare che l'utente voglia modificare le credenziali di tale applicazione e successivamente accedervi manualmente; il plug-in, quindi, si mette in ascolto dell'eventuale accesso da parte dell'utente e nel caso in cui questo avvenga, recupera login e password dell'applicazione. Successivamente ad avvenuto accesso da parte dell'utente, il plug-in controlla se tale applicazione è presente in vari domini o account dell'utente o con un nome applicazione diverso e se le credenziali sono cambiate. Ci potrebbero, quindi, essere varie combinazioni e il plug-in presenta all'utente un popup con le possibili scelte della stessa applicazione per le quali è possibili aggiornare le credenziali e per le quali invece le credenziali sono le stesse. L'utente, quindi, può scegliere quali aggiornare con le attuali credenziali appena utilizzate per l'accesso e alla conferma dell'aggiornamento il plug-in richiede a Monokee la chiave di cifratura, cifra le nuove credenziali e le invia a Monokee insieme ai dati dell'applicazione, quali: URL_G, nome del dominio, nome dell'applicazione e account.

Capitolo 2

Tecnologie, strumenti e linguaggi utilizzati

In questo capitolo vengono illustrate le tecnologie in Sezione 2.1, gli strumenti e i linguaggi utilizzati per la realizzazione del progetto, rispettivamente nelle sezioni: 2.2 e 2.3.

2.1 Tecnologie

2.1.1 Crossrider

Crossrider [5], il cui logo è rappresentato in Figura 2.1, è un framework$_G$ cloud-based che permette la creazione e lo sviluppo di estensioni cross-platform, ovvero funzionanti nei principali browser, quali: Chrome, Internet Explorer, Firefox e Safari.

Figura 2.1: Logo Crossrider

Per la realizzazione dell'estensione è sufficiente scrivere un unico codice in Javascript ed esso sarà interpretato da tutti i browser citati, senza dover creare un plug-in specifico per ognuno di essi. Un enorme vantaggio se si pensa in termini di risparmio di tempo per l'azienda che, anzichè sviluppare varie estensioni ne dovrà realizzare soltanto una e questa sarà funzionante in tutti i principali browser.

È importante precisare che Crossrider lavora con due *scope* di esecuzione del codice: *Extension Page Scope* e *Background Scope*. Il primo viene eseguito ad ogni caricamento di una pagina e ha il compito di interagire con il DOM$_G$ di quest'ultima; il secondo, invece, viene eseguito una sola volta in background dal browser e persiste fino a quando quest'ultimo è in esecuzione e l'estensione è installata. Il *Background Scope* permette di gestire le attività in background come: i bottoni e le schede del browser, il recupero dei

dati e la gestione di timer. Gli *scope* lavorano in maniera indipendente l'uno dall'altro e non possono interagire direttamente tra loro; il codice in *Extension Page Scope*, per esempio, non può richiamare funzioni che risiedono nel *Background Scope* e viceversa. Le API_G di Crossrider, però, mettono a disposizione un sistema di *messaging* e *page events* che permettono agli *scope* di comunicare facilmente tra di loro.

Le API_G di Crossrider offrono un modo semplice di comunicazione tra l'estensione e il browser; esse, tra le varie funzionalità, permettono anche di: salvare dati su un database locale, effettuare richieste HTTP, convertire i dati in $JSON_G$ e viceversa, implementare il *fire* e il *bind* di eventi.

2.2 Strumenti

2.2.1 jQuery

jQuery (logo in Figura 2.2) è una libreria Javascript veloce, snella e ricca di fuzionalità che rende la manipolazione dell'HTML, la gestione degli eventi, le chiamate Ajax e le animazioni molto più semplici utilizzando un API_G facile da usare e funzionante su tutti i principali browser.

Figura 2.2: Logo jQuery

2.2.2 Ajax

Ajax [6] (logo in Figura 2.3), acronimo di Asynchronous JavaScript + XML, non è una tecnologia, ma un nuovo approccio alle applicazioni web o meglio è un insieme di tecnologie per poter garantire all'utente un'esperienza più reattiva possibile durante l'utilizzo delle applicazioni web.

Figura 2.3: Logo Ajax

Nel modello di applicazione web tradizionale, il più delle volte l'interazione dell'utente provoca una richiesta HTTP a un server web, il quale elabora i dati e restituisce una pagina HTML al client; il tutto facendo rimanere l'utente in attesa della risposta da parte

del server. L'approccio Ajax, invece, elimina lo start-stop-start-stop del modello tradizionale introducendo un motore Ajax tra l'utente e il server. Il motore Ajax permette che l'interazione con il server avvenga in modo asincrono, il codice dell'applicazione continua senza aspettare la risposta del server cosicchè l'utente possa continuare a interagire con l'applicazione e non rimanga in attesa.

2.2.3 Git

Git [7] (logo in Figura 2.4) è un sistema software di controllo di versione distribuito creato da Linus Torvals nel 2005.

Figura 2.4: Logo Git

Permette di tenere traccia delle varie versioni di un prodotto, consentendo nel caso fosse necessario di tornare a una qualsiasi versione precedente, evitando la perdita dei dati e le sovrascitture accidentali. Questo è possibile grazie al fatto che Git considera i dati come una serie di immagini di un filesystem, ovvero ogni volta che si esegue un *commit* del proprio lavoro Git effettua un'immagine di tutti i file in quel momento e per essere maggiormente efficiente non salva nuovamente i file che non sono cambiati ma ne crea un collegamento al file già salvato in precedenza. L'altro enorme vantaggio di Git è che la maggior parte delle sue operazioni hanno bisogno solo di file e risorse locali, questo significa che tutto il lavoro su di un progetto può essere realizzato in locale e una volta completato può essere tranquillamente *commitato* avendo così la garanzia che il lavoro appena svolto non andrà perso.

2.2.4 BitBucket

BitBucket (logo in Figura 2.5) è un servizio web-based di hosting per progetti Git o Mercurial.

Figura 2.5: Logo BitBucket

Esso permette un sistema efficiente di revisione del codice tramite *pull request* e cronologia dei *commit*. Inoltre, si ha la possibilità di registrarsi con un account gratuito e avere un numero illimitato di repository alle quali è possibili aggiungere fino a cinque collaboratori.

2.2.5 QUnit

QUnit (logo in Figura 2.6) è un framework$_G$ potente e facile da utilizzare per effettuare test di unità di codice Javascript.

Figura 2.6: Logo QUnit

Esso offre la possibilità di testare, oltre al generico codice Javascript, il codice realizzato con jQuery, quindi: la manipolazione del DOM$_G$ e le chiamate asincrone; e di testare il codice scatenato in seguito ad azioni dell'utente, simulando quest'ultime. Inoltre, permette di automatizzare i test grazie a un *Test Runner* per ogni componente da testare.

2.3 Linguaggi

Javascript È un linguaggio di scripting orientato agli oggetti e agli eventi comunemente usato per la creazione di applicazioni web. Viene sviluppato nel 1995 da Brendan Eich di Netscape con il nome di *LiveScript* e permette di creare documenti dinamici in grado di interagire con l'utente. Con Javascript divenne possibile trasferire l'elaborazione di alcuni dati direttamente al browser anzichè effettuare continue richieste al server e attendere le relative risposte, permettendo così la trasformazione delle pagine web da statiche a dinamiche.

HTML Acronimo di *Hyper Text Markup Language*, è un linguaggio di markup per la creazione di ipertesti e descrive, tramite l'utilizzo di appositi *tag*, al browser come deve apparire la pagina web. I *tag* consentono al documento di avere una propria struttura e di organizzare in modo gerarchico i contenuti al suo interno; oltre ad offrire altre numerose funzionalità.

CSS Acronimo di *Cascading Style Sheets*, è un linguaggio che permette di inserire aspetti di visualizzazione nella pagine HTML e ne consente una netta separazione tra presentazione e struttura. Le regole per comporre fogli di stile sono contenute nelle *Recommendations* del World Wide Web Consortium (W3C), redatte a partire dal 1996.

Capitolo 3

Analisi dei Requisiti

In questo capitolo vengono presentati i casi d'uso (Sezione 3.1) e i requisiti (Sezione 3.2) rilevati durante l'attività di analisi del progetto.

3.1 Casi d'uso

Ogni caso d'uso è identificato da un codice con il seguente formalismo:

UC[codice]

dove codice è un identificativo univoco in forma gerarchica.

3.1.1 Attori individuati

Nell'analisi dei casi d'uso sono stati individuati i seguenti attori:

- **Utente:** persona che ha un'identità su Monokee;
- **Utente autenticato a Monokee:** persona autenticata all'interno di Monokee con uno dei suoi account;
- **Monokee:** sistema di Access Management descritto in Sezione 1.3.2;
- **Applicazione web:** applicazione cloud che necessita di un'autenticazione per accedervi.

3.1.2 UC0 Scenario principale

Rappresentato in Figura 3.1:

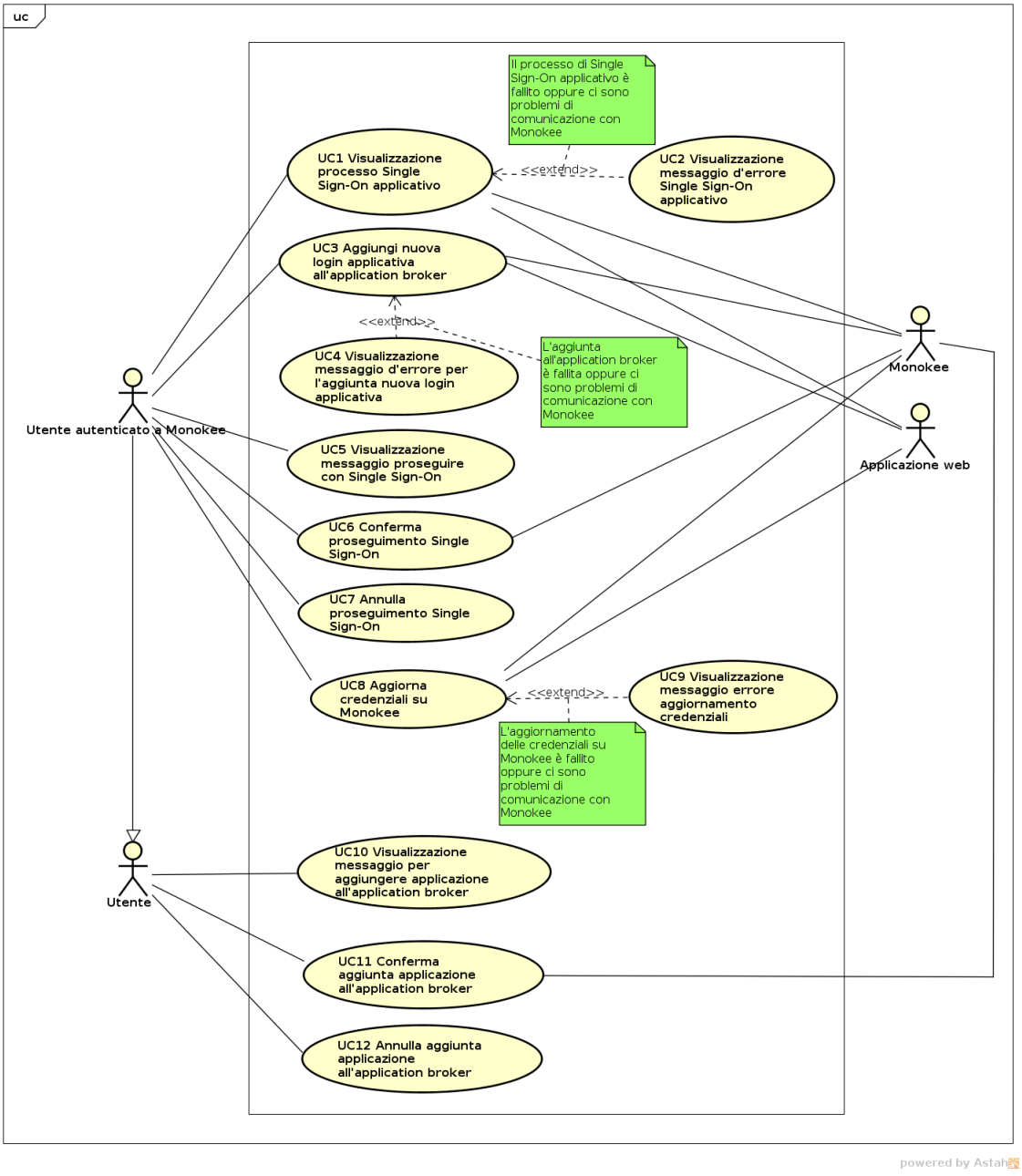


Figura 3.1: UC0: Scenario principale

- **Attori principali:** Utente, Utente autenticato a Monokee.
- **Attori secondari:** Monokee, Applicazione web.
- **Scopo e descrizione:** l'utente autenticato a Monokee ha due funzionalità principali che possono essere attivate dall'application broker personale dell'utente autenticato a Monokee oppure direttamente durante la navigazione dall'applicazione web stessa: visualizzare il Single Sign-On applicativo e aggiungere una nuova login applicativa all'application broker dell'utente per poter effettuare successivamente il Single Sign-On anche di tale nuova applicazione. Durante la navigazione direttamente dall'applicazione web, l'utente autenticato a Monokee può: visualizzare un messaggio per proseguire con il Single Sign-On di tale applicazione e confermare o annullare tale operazione. Qualora decidesse di accedere all'applicazione manualmente, l'utente autenticato a Monokee all'avvenuto accesso all'interno dell'applicazione può visualizzare un messaggio contenente la lista dei possibili aggiornamenti di credenziali che può effettuare; l'utente autenticato a Monokee, quindi, può aggiornare le credenziali della stessa applicazione, presente in altri suoi domini o account, con le attuali credenziali. L'utente, che potrebbe essere sia un utente autenticato a Monokee sia non autenticato, può visualizzare un messaggio per l'aggiunta di una nuova login applicativa direttamente dall'applicazione web in cui si trova e confermare o annullare tale aggiunta.
- **Pre-condizione:** il sistema è installato.
- **Flusso principale degli eventi:**
  1. L'utente autenticato a Monokee visualizza il processo di Single Sign-On applicativo [UC1].
  2. L'utente autenticato a Monokee aggiunge una nuova login applicativa al suo application broker [UC3].
  3. L'utente autenticato su Monokee visualizza un messaggio per proseguire con il Single Sign-On [UC5].
  4. L'utente autenticato a Monokee conferma il proseguimento con il Single Sign-On [UC6].
  5. L'utente autenticato a Monokee annulla il proseguimento con il Single Sign-On [UC7].
  6. L'utente autenticato a Monokee aggiorna le credenziali su Monokee [UC8].
  7. L'utente visualizza un messaggio per aggiungere un'applicazione al suo application broker [UC10].
  8. L'utente conferma l'aggiunta di un'applicazione al suo application broker [UC11].
  9. L'utente annulla l'aggiunta di un'applicazione al suo application broker [UC12].
- **Generalizzazione tra attori:** l'utente autenticato a Monokee può effettuare tutte le azioni dell'utente.

- **Scenari alternativi:** il processo di Single Sign-On applicativo, l'aggiunta di una nuova login applicativa e l'aggiornamento delle credenziali su Monokee possono fallire. Il fallimento può essere dovuto da problemi di comunicazione con Monokee oppure perchè il database di quest'ultimo non è disponibile.
- **Estensioni:**
  - L'utente autenticato a Monokee visualizza un messaggio d'errore del Single Sign-On applicativo [UC2].
  - L'utente autenticato a Monokee visualizza un messaggio d'errore per l'aggiunta di una nuova login applicativa [UC4].
  - L'utente autenticato a Monokee visualizza un messaggio d'errore per l'aggiornamento delle credenziali su Monokee [UC9].
- **Post-condizione:** il sistema ha eseguito le operazioni scelte dagli attori Utente e Utente autenticato a Monokee.

3.1.3 UC1 Visualizzazione processo Single Sign-On applicativo

- **Attori principali:** Utente autenticato a Monokee;
- **Attori secondari:** Monokee, Applicazione web;
- **Scopo e descrizione:** l'utente autenticato a Monokee visualizza il processo di Single Sign-On applicativo;
- **Pre-condizione:** il sistema è installato e ha intercettato l'evento di Single Sign-On inviato da Monokee;
- **Flusso principale degli eventi:** l'utente autenticato su Monokee visualizza il processo di Single Sign-On applicativo [UC1];
- **Post-condizione:** il sistema ha mostrato il processo il Single Sign-On applicativo all'utente autenticato a Monokee per l'applicazione web richiesta.

3.1.4 UC2 Visualizzazione messaggio errore Single Sign-On applicativo

- **Attori principali:** Utente autenticato a Monokee;
- **Scopo e descrizione:** l'utente autenticato a Monokee visualizza un messaggio d'errore con la motivazione del fallimento del Single Sign-On dell'applicazione web richiesta;
- **Pre-condizione:** il sistema ha ricevuto un messaggio contenente il motivo del fallimento del Single Sign-On applicativo;
- **Flusso principale degli eventi:** l'utente autenticato a Monokee visualizza un messaggio di errore del Single Sign-On applicativo [UC2];

- **Post-condizione:** il sistema ha mostrato all'utente autenticato a Monokee un messaggio di errore riguardante il fallimento del Single Sign-On applicativo dell'applicazione web richiesta.

3.1.5 UC3 Aggiungi nuova login applicativa all'application broker

Rappresentato in Figura 3.2:

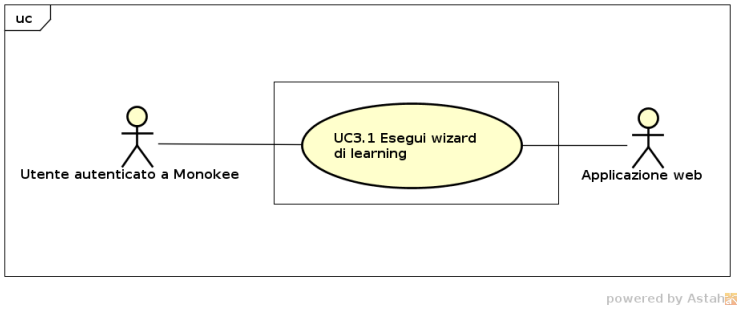


Figura 3.2: UC3: Aggiungi nuova login applicativa

- **Attori principali:** Utente autenticato a Monokee;
- **Attori secondari:** Monokee, Applicazione web;
- **Scopo e descrizione:** l'utente autenticato può aggiungere una nuova login applicativa al suo application broker per effettuarne il Single Sign-On successivamente;
- **Pre-condizione:** il sistema è installato e ha intercettato l'evento di learning inviato da Monokee;
- **Flusso principale degli eventi:** l'utente autenticato a Monokee esegue il wizard di learning [UC3.1];
- **Post-condizione**: il sistema ha ricevuto una risposta con esito positivo per l'aggiunta della login applicativa nell'application broker dell'utente autenticato a Monokee.

3.1.6 UC3.1 Esegui wizard di learning

Rappresentato in Figura 3.3:

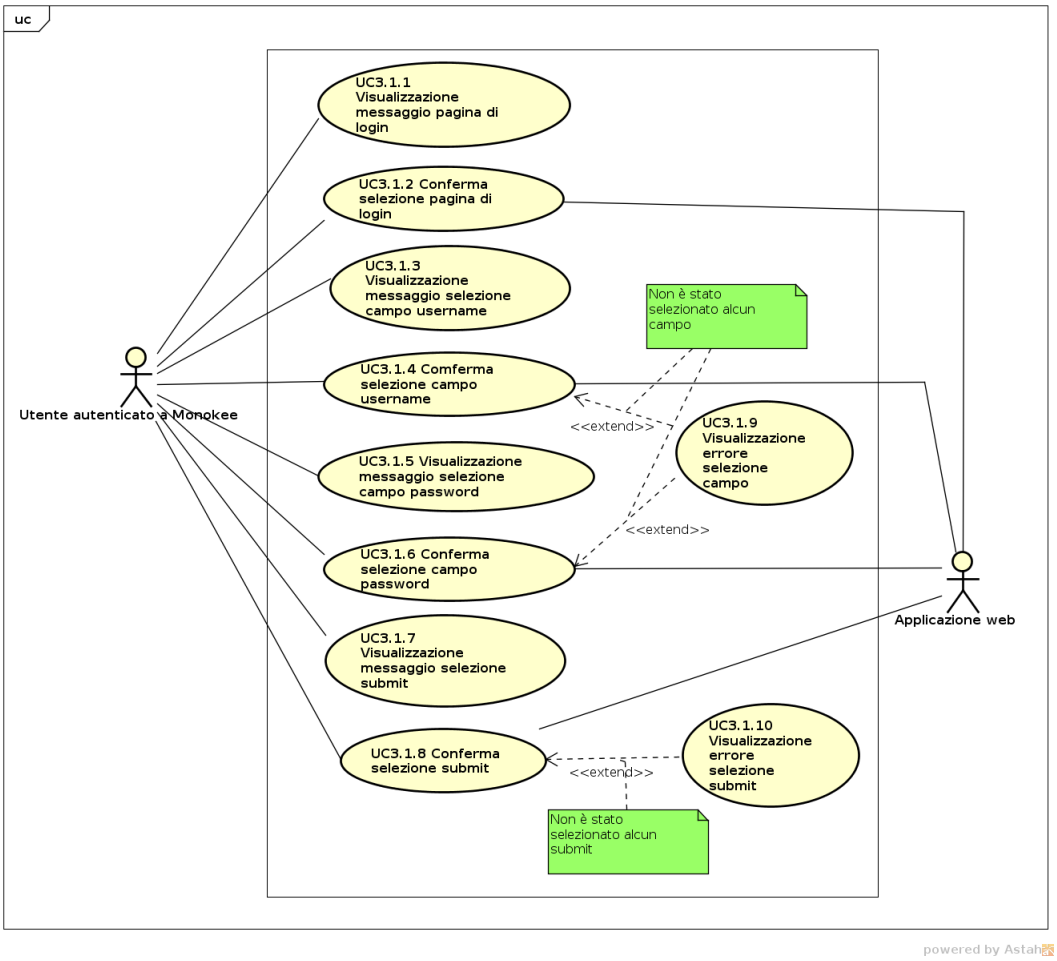


Figura 3.3: UC3.1: Esegui wizard di learning

- **Attori principali:** Utente autenticato a Monokee;
- **Attori secondari:** Applicazione web;
- **Scopo e descrizione:** l'utente autenticato a Monokee esegue il wizard di learning che lo guida nell'aggiunta di una nuova login applicativa al suo application broker;
- **Pre-condizione:** il sistema è installato e ha intercettato l'evento di learning inviato da Monokee;
- **Flusso principale degli eventi:**
  1. L'utente autenticato a Monokee visualizza il messaggio riguardante la pagina di login [UC3.1.1];
  2. L'utente autenticato a Monokee conferma la selezione della pagina di login [UC3.1.2];

3. L'utente autenticato a Monokee visualizza il messaggio di selezione del campo username [UC3.1.3];
4. L'utente autenticato a Monokee conferma la selezione del campo username [UC3.1.4];
5. L'utente autenticato a Monokee visualizza il messaggio di selezione del campo password [UC3.1.5];
6. L'utente autenticato a Monokee conferma la selezione del campo password [UC3.1.6];
7. L'utente autenticato a Monokee visualizza il messaggio di selezione del submit [UC3.1.7];
8. L'utente autenticato a Monokee conferma la selezione del submit [UC3.1.8].

- **Scneari alternativi:**
  - Non è stato selezionato alcun campo dalla pagina dell'appplicazione web;
  - Non è stato selezionato alcun submit dalla pagina dell'appplicazione web.
- **Estensioni:**
  - L'utente autenticato a Monokee visualizza un errore di selezione campo [UC3.1.9];
  - L'utente autenticato a Monokee visualizza un errore di selezione submit [UC3.1.10].
- **Post-condizione:** il sistema ha mostrato il wizard di learning permettendo all'utente autenticato a Monokee la sua esecuzione al fine di aggiungere una nuova login applicativa al suo application broker.

3.1.7 UC3.1.1 Visualizzazione messaggio pagina di login

- **Attori principali:** Utente autenticato a Monokee;
- **Scopo e descrizione:** l'utente visualizza le istruzioni del wizard riguardanti la pagina di login;
- **Pre-condizione:** il sistema è installato e ha intercettato l'evento di learning inviato da Monokee;
- **Flusso principale degli eventi:** l'utente autenticato a Monokee visualizza il messaggio riguardante la pagina di login [UC3.1.1];
- **Post-condizione:** il sistema ha mostrato il messaggio riguardante la pagina di login.

3.1.8 UC3.1.2 Conferma selezione pagina di login

- **Attori principali:** Utente autenticato a Monokee;
- **Attori secondari:** Applicazione web;
- **Scopo e descrizione:** l'utente autenticato a Monokee conferma la selezione della pagina di login dell'applicazione web;
- **Pre-condizione:** il sistema ha mostrato il messaggio riguardante la pagina di login del wizard di learning;
- **Flusso principale degli eventi:** l'utente autenticato a Monokee conferma la selezione della pagina di login [UC3.1.2];
- **Post-condizione:** il sistema ha confermato la selezione della pagina di login dell'applicazione web.

3.1.9 UC3.1.3 Visualizzazione messaggio selezione campo username

- **Attori principali:** Utente autenticato a Monokee;
- **Scopo e descrizione:** l'utente autenticato a Monokee visualizza il messaggio del wizard di learning di selezione del campo username dell'applicazione web;
- **Pre-condizione:** il sistema ha confermato la selezione della pagina di login dell'applicazione;
- **Flusso principale degli eventi:** l'utente autenticato a Monokee visualizza il messaggio di selezione campo username [UC3.1.3];
- **Post-condizione:** il sistema ha mostrato il messaggio di selezione del campo username del wizard di learning.

3.1.10 UC3.1.4 Conferma selezione campo username

- **Attori principali:** Utente autenticato a Monokee;
- **Attori secondari:** Applicazione web;
- **Scopo e descrizione:** l'utente autenticato a Monokee conferma la selezione del campo username della pagina di login dell'applicazione web;
- **Pre-condizione:** il sistema ha mostrato il messaggio di selezione del campo username del wizard di learning;
- **Flusso principale degli eventi:** l'utente autenticato a Monokee conferma la selezione del campo username [UC3.1.4];
- **Post-condizione:** il sistema ha confermato la selezione del campo username della pagina di login dell'applicazione web;

3.1.11 UC3.1.5 Visualizzazione messaggio selezione campo password

- **Attori principali:** Utente autenticato a Monokee;
- **Scopo e descrizione:** l'utente autenticato a Monokee può visualizzare il messaggio del wizard di learning di selezione del campo password della pagina di login dell'applicazione web;
- **Pre-condizione:** il sistema ha confermato la selezione del campo username della pagina di login dell'applicazione web;
- **Flusso principale degli eventi:** l'utente autenticato a Monokee visualizza il messaggio di selezione del campo password [UC3.1.5];
- **Post-condizione:** il sistema ha mostrato il messaggio di selezione del campo password della pagina di login dell'applicazione web.

3.1.12 UC3.1.6 Conferma selezione campo password

- **Attori principali:** Utente autenticato a Monokee;
- **Attori secondari:** Applicazione web;
- **Scopo e descrizione:** l'utente autenticato a Monokee può confermare la selezione del campo password della pagina di login dell'applicazione web;
- **Pre-condizione:** il sistema ha mostrato il messaggio del wizard di learning di selezione del campo password;
- **Flusso principale degli eventi:** l'utente autenticato a Monokee conferma la selezione del campo password [UC3.1.6];
- **Post-condizione:** il sistema ha confermato la selezione del campo password della pagina di login dell'applicazione web.

3.1.13 UC3.1.7 Visualizzazione messaggio selezione submit

- **Attori principali:** Utente autenticato a Monokee;
- **Scopo e descrizione:** l'utente autenticato a Monokee può visualizzare il messaggio del wizard di learning di selezione del submit della pagina di login dell'applicazione;
- **Pre-condizione:** il sistema ha confermato la selezione del campo password della pagina di login dell'applicazione web;
- **Flusso principale degli eventi:** l'utente autenticato su Monokee visualizza il messaggio di selezione submit [UC3.1.7];
- **Post-condizione:** il sistema ha visualizzato il messaggio di selezione del submit della pagina di login dell'applicazione web.

3.1.14 UC3.1.8 Conferma selezione submit

- **Attori principali:** Utente autenticato a Monokee;
- **Attori secondari:** Applicazione web;
- **Scopo e descrizione:** l'utente autenticato a Monokee può confermare la selezione del submit della pagina di login dell'applicazione web;
- **Pre-condizione:** il sistema ha mostrato il messaggio del wizard di learning di selezione del submit dalla pagina di login dell'applicazione;
- **Flusso principale degli eventi:** l'utente autenticato a Monokee conferma la selezione del submit [UC3.1.8];
- **Post-condizione:** il sistema ha confermato la selezione del submit della pagina di login dell'applicazione web.

3.1.15 UC3.1.9 Visualizzazione errore selezione campo

- **Attori principali:** Utente autenticato a Monokee;
- **Scopo e descrizione:** l'utente autenticato a Monokee visualizza un errore di selezione del campo della pagina di login dell'applicazione;
- **Pre-condizione:** il sistema non ha ricevuto alcuna selezione del campo username o del campo password della pagina di login dell'applicazione web;
- **Flusso principale degli eventi:** l'utente autenticato a Monokee visualizza un errore di selezione campo [UC3.1.9];
- **Post-condizione:** il sistema ha mostrato all'utente autenticato a Monokee l'errore di selezione campo della pagina di login dell'applicazione web.

3.1.16 UC3.1.10 Visualizzazione errore selezione submit

- **Attori principali:** Utente autenticato a Monokee;
- **Scopo e descrizione:** l'utente autenticato a Monokee visualizza un errore di selezione del submit della pagina di login dell'applicazione;
- **Pre-condizione:** il sistema non ha ricevuto alcuna selezione del submit della pagina di login dell'applicazione;
- **Flusso principale degli eventi:** l'utente autenticato a Monokee visualizza un errore di selezione submit [UC3.1.10];
- **Post-condizione:** il sistema ha mostrato all'utente autenticato a Monokee l'errore di selezione submit della pagina di login dell'applicazione web.

3.1.17 UC4 Visualizzazione messaggio errore per l'aggiunta nuova login applicativa

- **Attori principali:** Utente autenticato a Monokee;
- **Scopo e descrizione:** l'utente autenticato a Monokee visualizza un messaggio d'errore che descrive il motivo del fallimento dell'aggiunta di una nuova login applicativa all'application broker dell'utente;
- **Pre-condizione:** il sistema ha mostrato il wizard di learning e ha ricevuto un messaggio di fallimento per l'aggiunta di una nuova login applicativa nell'application broker dell'utente;
- **Flusso principale degli eventi:** l'utente autenticato a Monokee visualizza messaggio d'errore per l'aggiunta di una nuova login applicativa [UC4];
- **Post-condizione:** il sistema ha mostrato un messaggio d'errore per il fallimento dell'aggiunta di una nuova login applicativa all'application broker dell'utente autenticato a Monokee.

3.1.18 UC5 Visualizzazione messaggio proseguire con Single Sign-On

- **Attori principali:** Utente autenticato a Monokee;
- **Scopo e descrizione:** l'utente autenticato a Monokee visualizza un messaggio per proseguire con il Single Sign-On direttamente dall'applicazione web durante la navigazione all'esterno di Monokee;
- **Pre-condizione:** il sistema è installato e ha identificato, durante la navigazione all'esterno di Monokee, che l'attuale applicazione web è presente nell'application broker dell'utente autenticato a Monokee;
- **Flusso principale degli eventi:** l'utente autenticato a Monokee può visualizzare un messaggio per procedere al Single Sign-On dell'applicazione web [UC5];
- **Post-condizione:** il sistema ha mostrato all'utente autenticato a Monokee un messaggio per proseguire con il Single Sign-On dell'applicazione web in cui si trova durante la navigazione all'esterno di Monokee.

3.1.19 UC6 Conferma proseguimento Single Sign-On

- **Attori principali:** Utente autenticato a Monokee;
- **Scopo e descrizione:** l'utente autenticato a Monokee può confermare il proseguimento con il Single Sign-On dell'applicazione web in cui si trova durante la navigazione all'esterno di Monokee;

- **Pre-condizione:** il sistema ha mostrato all'utente autenticato a Monokee un messaggio per proseguire con il Single Sign-On dell'applicazione web;
- **Flusso principale degli eventi:** l'utente autenticato a Monokee conferma il proseguimento con il Single Sign-On [UC6];
- **Post-condizione:** il sistema ha ricevuto la conferma al proseguimento del Single Sign-On dell'applicazione web in cui si trova l'utente durante la navigazione all'esterno di Monokee.

3.1.20 UC7 Annulla proseguimento Single Sign-On

- **Attori principali:** Utente autenticato a Monokee;
- **Scopo e descrizione:** l'utente autenticato a Monokee può annullare il proseguimento con il Single Sign-On dell'applicazione web in cui si trova durante la navigazione, all'esterno di Monokee;
- **Pre-condizione:** il sistema ha mostrato all'utente autenticato a Monokee un messaggio per proseguire con il Single Sign-On dell'applicazione web;
- **Flusso principale degli eventi:** l'utente autenticato a Monokee annulla il proseguimento con il Single Sign-On [UC7];
- **Post-condizione:** il sistema ha ricevuto l'annullamento al proseguimento del Single Sign-On dell'applicazione web in cui si trova l'utente durante la navigazione all'esterno di Monokee.

3.1.21 UC8 Aggiorna credenziali su Monokee

Rappresentato in Figura 3.4:

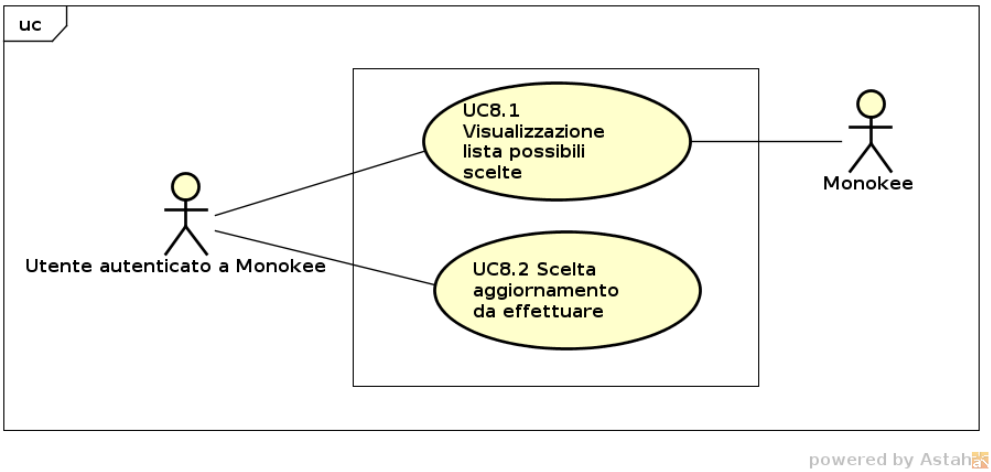


Figura 3.4: UC8: Aggiorna credenziali su Monokee

- **Attori principali:** Utente autenticato a Monokee;
- **Attori secondari:** Monokee, Applicazione web;
- **Scopo e descrizione:** l'utente autenticato a Monokee può aggiornare le credenziali su Monokee della stessa applicazione web, in uno o più dei suoi domini e/o in uno o più dei suoi account;
- **Pre-condizione:** il sistema ha intercettato l'evento di successo dell'accesso, da parte dell'utente autenticato a Monokee, all'interno di una delle applicazione web già presenti nel suo application broker;
- **Flusso principale degli eventi:**
  - L'utente autenticato a Monokee visualizza la lista delle possibili scelte [UC8.1];
  - L'utente autenticato a Monokee sceglie gli aggiornamenti da effettuare [UC8.2].
- **Post-condizione:** il sistema ha ottenuto una risposta da Monokee con esito positivo per l'aggiornamento che l'utente autenticato a Monokee ha scelto di effettuare.

3.1.22 UC8.1 Visualizzazione lista possibili scelte

- **Attori principali:** Utente autenticato a Monokee.
- **Scopo e descrizione:** l'utente autenticato a Monokee visualizza la lista contenente l'applicazione in uno o più dei suoi domini e/o in uno o più dei suoi account. In particolare, per ogni elemento della lista viene visualizzato: l'immagine del dominio, il nome del dominio, il nome dell'applicazione e l'account a cui tale applicazione è associata.
- **Pre-condizione:** il sistema ha intercettato l'evento di successo dell'accesso all'interno di una delle applicazione web già presenti nell'application broker dell'utente autenticato a Monokee.
- **Flusso principale degli eventi:** l'utente autenticato a Monokee visualizza la lista delle possibili scelte [UC8.1].
- **Post-condizione:** il sistema ha mostrato all'utente autenticato a Monokee la lista delle possibili scelte per cui effettuare l'aggiornamento delle credenziali dell'applicazione su Monokee.

3.1.23 UC8.2 Scelta aggiornamento da effettuare

- **Attori principali:** Utente autenticato a Monokee;
- **Scopo e descrizione:** l'utente autenticato a Monokee sceglie quale tra i possibili aggiornamenti delle credenziali effettuare con le attuali credenziali usate per accedere all'applicazione;
- **Pre-condizione:** il sistema ha mostrato all'utente autenticato a Monokee la lista delle possibili scelte;
- **Flusso principale degli eventi:** l'utente autenticato a Monokee sceglie l'aggiornamento da effettuare [UC8.2];
- **Post-condizione:** il sistema ha ricevuto l'aggiornamento delle credenziali da effettuare su Monokee.

3.1.24 UC9 Visualizzazione messaggio errore aggiornamento credenziali

- **Attori principali:** Utente autenticato a Monokee;
- **Scopo e descrizione:** l'utente autenticato a Monokee visualizza un messaggio d'errore che descrive il motivo del fallimento dell'aggiornamento delle credenziali effettuato su Monokee;

- **Pre-condizione:** il sistema ha ricevuto una risposta con esito negativo per l'aggiornamento delle credenziali richieste;
- **Flusso principale degli eventi:** l'utente autenticato a Monokee visualizza messaggio d'errore per l'aggiornamento delle credenziali fallito [UC9];
- **Post-condizione:** il sistema ha mostrato all'utente autenticato a Monokee un messaggio d'errore per l'aggiornamento delle credenziali richiesto e fallito.

3.1.25 UC10 Visualizzazione messaggio per aggiungere applicazione all'application broker

- **Attori principali:** Utente;
- **Scopo e descrizione:** l'utente visualizza un messaggio per aggiungere, al suo application broker di Monokee, l'applicazione web in cui si trova durante la navigazione;
- **Pre-condizione:** il sistema è installato;
- **Flusso principale degli eventi:** l'utente visualizza un messaggio per aggiungere la pagina al proprio application broker [UC10];
- **Post-condizione:** il sistema ha mostrato un messaggio per aggiungere la pagina dell'applicazione web all'application broker dell'utente.

3.1.26 UC11 Conferma aggiunta applicazione all'application broker

- **Attori principali:** Utente;
- **Scopo e descrizione:** l'utente può confermare l'aggiunta dell'applicazione web in cui si trova, durante la navigazione, al proprio application broker;
- **Pre-condizione:** il sistema ha mostrato il messaggio per aggiungere all'application broker l'applicazione web in cui si trova l'utente durante la navigazione;
- **Flusso principale degli eventi:** l'utente conferma l'aggiunta dell'applicazione web al proprio application broker [UC11];
- **Post-condizione:** il sistema ha confermato la richiesta di aggiunta dell'applicazione web in cui si trova l'utente durante la navigazione, all'application broker di quest'ultimo.

3.1.27 UC12 Annulla aggiunta applicazione all'application broker

- **Attori principali:** Utente;
- **Scopo e descrizione:** l'utente può annullare l'aggiunta dell'applicazione web in cui si trova durante la navigazione;
- **Pre-condizione:** il sistema ha mostrato il messaggio per aggiungere all'application broker l'applicazione web in cui si trova l'utente durante la navigazione;
- **Flusso principale degli eventi:** l'utente annulla la richiesta di aggiunta dell'applicazione web al proprio application broker [UC12];
- **Post-condizione:** il sistema ha annullato la richiesta di aggiunta dell'applicazione web in cui si trova l'utente durante la navigazione, all'application broker di quest'ultimo.

3.2 Requisiti

In questa sezione vengono riportati in forma tabulare tutti i requisiti accettati dall'azienda ospitante. Ogni requisito è descritto da un codice che lo identifica univocamente, da una descrizione e dalla fonte dalla quale è stato ricavato. I requisiti possono essere di vari tipo: funzionali, di qualità, prestazionali o di vincolo e avere gradi di importanza diversi; per questo, ognuno di essi è identificato da un codice con il seguente formalismo:

R[importanza][tipo][codice]

dove, importanza assume uno tra i seguenti valori:

- 0: requisito obbligatorio;
- 1: requisito desiderabile;
- 2: requisito opzionale.

tipo assume uno tra i seguenti valori:

- F: requisito funzionale;
- Q: requisito di qualità;
- P: requisito prestazionale;
- V: requisito di vincolo.

e codice rappresenta il codice univoco di ogni requisito in forma gerarchica.

Requisito	**Descrizione**	**Fonti**
R0F1	Il sistema deve intercettare gli eventi di learning e di Single Sign-On inviati da Monokee.	Specifiche
R0F2	L'utente autenticato a Monokee deve poter visualizzare il processo di Single Sign-On applicativo.	UC1
R0F2.1	Il sistema deve recuperare dalla pagina di Monokee i dati per poter eseguire il processo di autenticazione applicativa dell'utente. In particolare, deve recuperare l'URL_G della pagina di login dell'applicazione web richiesta dall'utente, gli identificativi dei campi di login, password e del bottone di conferma e infine i valori dei campi username e password.	UC1

R0V2.2	Il sistema deve comunicare con Monokee per ottenere la chiave con cui decriptare i dati recuperati dalla pagina di Monokee per il processo di Single Sign-On applicativo.	UC1
R0F2.3	I dati recuperati dalla pagina di Monokee per il processo di Single Sign-On devono essere decriptati con la chiave di decifratura.	Specifiche
R0F2.4	Il sistema deve compilare i campi login e password della pagina di login dell'applicazione web con i rispettivi valori decriptati e simulare il click sul bottone di submit per effettuare l'autenticazione al posto dell'utente.	UC1
R2F2.5	Il sistema deve rendere invisibile all'utente il processo di Single Sign-On applicativo.	Specifiche
R0F3	L'utente autenticato a Monokee deve poter visualizzare un messaggio d'errore nel caso in cui il processo di Single Sign-On applicativo non vada a buon fine.	UC2
R0F4	L'utente autenticato a Monokee deve poter aggiungere una nuova login applicativa al suo application broker.	UC3
R0F4.1	L'utente autenticato a Monokee deve poter eseguire il wizard di learning.	UC3.1
R0F4.1.1	L'utente autenticato a Monokee deve poter visualizzare il messaggio del wizard di learning riguardante la pagina di login.	UC3.1.1
R0F4.1.2	L'utente autenticato a Monokee deve poter confermare la selezione della pagina di login.	UC3.1.2
R0F4.1.3	L'utente autenticato a Monokee deve poter visualizzare il messaggio del wizard di learning di selezione del campo username.	UC3.1.3
R0F4.1.4	L'utente autenticato a Monokee deve poter confermare la selezione del campo username.	UC3.1.4

R0F4.1.5	L'utente autenticato a Monokee deve poter visualizzare il messaggio del wizard di learning di selezione del campo password.	UC3.1.3
R0F4.1.6	L'utente autenticato a Monokee deve poter confermare la selezione del campo password.	UC3.1.6
R0F4.1.7	L'utente autenticato a Monokee deve poter visualizzare il messaggio del wizard di learning di selezione del submit.	UC3.1.7
R0F4.1.8	L'utente autenticato a Monokee deve poter confermare la selezione del submit.	UC3.1.8
R0F4.1.9	L'utente autenticato a Monokee deve poter visualizzare un messaggio d'errore quando non viene selezionato alcun campo durante il wizard di learning.	UC3.1.9
R0F4.1.10	L'utente autenticato a Monokee deve poter visualizzare un messaggio d'errore quando non viene selezionato alcun submit durante il wizard di learning.	UC3.1.10
R0F4.2	Il sistema deve comunicare con Monokee per ottenere la chiave di cifratura con cui criptare i dati recuperati dal wizard di learning.	UC3
R0F4.3	I dati recuperati dal wizard di learning devono essere criptati con la chiave di cifratura.	Specifiche
R0F4.4	Il sistema deve inviare a Monokee i dati criptati del wizard di learning, in particolare: l'URL_G della pagina di login dell'applicazione, l'identificativo del campo username, l'identificativo del campo password, l'identificativo del bottone di conferma presenti nella pagina di login dell'applicazione web.	UC3
R0F4.5	Il sistema deve redirigere l'utente autenticato a Monokee a una pagina di avvenuta modifica qualora l'aggiunta di una nuova login applicativa vada a buon fine.	UC3

R0F5	L'utente autenticato a Monokee deve poter visualizzare un messaggio d'errore qualora l'aggiunta di una nuova login applicativa dovesse fallire.	UC4
R0F6	L'utente autenticato a Monokee deve poter visualizzare, durante la navigazione nella pagina di login dell'applicazione web, un messaggio di richiesta per procedere con il Single Sign-On dell'applicazione stessa nel caso in cui quest'ultima sia già presente nell'application broker dell'utente.	UC5
R0F7	L'utente autenticato a Monokee deve poter confermare il proseguimento con il Single Sign-On dell'applicazione stessa durante la navigazione nella sua pagina di login.	UC6
R0F7.1	In caso di conferma al proseguimento con il Single Sign-On, il sistema deve inviare a Monokee l'URL_G della pagina di login dell'applicazione web.	Specifiche
R0F8	L'utente autenticato a Monokee deve poter annullare il proseguimento con il Single Sign-On dell'applicazione stessa durante la navigazione nella sua pagina di login.	UC7
R0F9	L'utente autenticato a Monokee deve poter aggiornare le credenziali dell'applicazione web, di cui ha effettuato l'autenticazione manualmente, in uno o più domini e/o in uno o più account dell'applicazione stessa con le credenziali che ha usato per accedervi.	UC8
R0F9.1	L'utente autenticato a Monokee deve poter visualizzare una lista dei possibili aggiornamenti di credenziali che può effettuare dell'applicazione web in cui si trova.	UC8.1
R0F9.2	L'utente autenticato a Monokee deve poter scegliere quale aggiornamento di credenziali, dell'applicazione in cui si trova, effettuare.	UC8.2
R0F9.3	Il sistema deve criptare le nuove credenziali con la chiave di cifratura ottenuta da Monokee.	Specifiche

R0F9.4	Il sistema deve inviare a Monokee le credenziali criptate, insieme ai dati dell'applicazione, quali: nome del dominio, nome dell'applicazione e account, per l'aggiornamento richiesto.	UC8
R0F10	L'utente autenticato a Monokee deve poter visualizzare un messaggio d'errore nel caso in cui l'aggiornamento delle credenziali richiesto non vada a buon fine.	UC9
R0F11	L'utente deve poter visualizzare un messaggio per aggiungere l'attuale applicazione al proprio application broker direttamente da quest'ultima durante la navigazione.	UC10
R0F12	L'utente deve poter confermare l'aggiunta, al proprio application broker, dell'applicazione in cui si trova durante la navigazione.	UC11
R0F13	L'utente deve poter annullare l'aggiunta, al proprio application broker, dell'applicazione in cui si trova durante la navigazione.	UC12
R0F14	Il sistema deve effettuare un check d'installazione intercettando l'evento d'installazione.	Specifiche
R0V15	La comunicazione dei dati tra il sistema e Monokee deve avvenire in formato JSON_G.	Specifiche
R0V16	Per la cifratura e decifratura dei dati, il sistema deve utilizzare l'algoritmo crittografico AES.	Specifiche
R0V17	Il sistema deve essere installabile nei seguenti browser: Firefox versione 38.0.5 o superiori, Chrome versione 44.0 o superiori, Safari versione 5.1.7 o superiori, Internet Explorer versione 11.	Specifiche
R0V18	Il sistema deve modificare lo stylesheet della pagina di login applicativa quando l'utente seleziona un campo, evidenziandolo.	Specifiche

R0V19	Il sistema deve mostrare tutti i messaggi in italiano o in inglese a seconda della lingua definita nella pagina web o di default in inglese.	Specifiche

Tabella 3.1: Requisiti

Capitolo 4

Progettazione

In questo capitolo verranno presentate l'architettura generale (Sezione 4.1) e le varie componenti (Sezione 4.2) che ho deciso di realizzare per progettare il sistema.

4.1 Architettura generale

L'architettura generale del sistema è rappresentata in Figura 4.1:

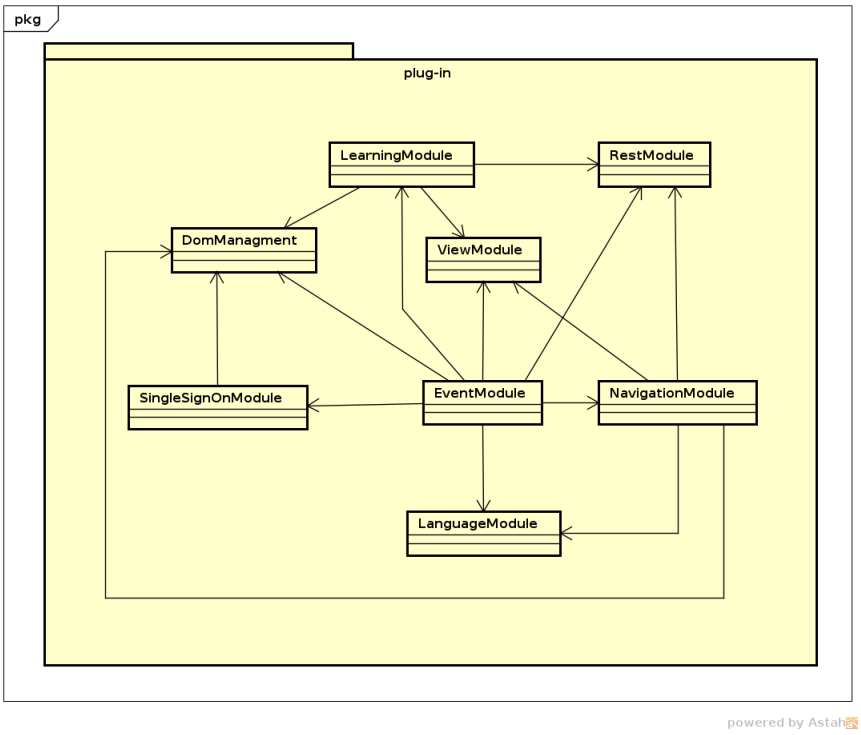


Figura 4.1: Architettura generale

Per realizzare un'architettura ben strutturata e organizzata ho utilizzato il Module Pattern [8]. Esso è un design pattern ormai consolidato nell'Ingegneria del Software, aiuta a mantenere le unità di codice separate e organizzate ottenendo come risultato un codice pulito e strutturato. I moduli, ovvero le unità di codice, non sono altro che dei contenitori di variabili e metodi; esso, quindi, offre la possibilità di incapsulare all'interno di sè attributi e metodi privati e di esporre all'esterno solamente i metodi pubblici.

Il Module Pattern definisce che ogni componente abbia il suo modulo distinto e che più moduli collaborino insieme per implementare le funzionalità del sistema ma, il codice dei vari moduli non sarà mai mescolato. Ciascun modulo realizza soltanto le funzionalità di un preciso componente in modo che siano ben chiare e distinte dagli altri componenti.

Ho utilizzato questo design pattern perchè, oltre a una struttura solida e organizzata, presenta i seguenti vantaggi, realizzando moduli ben suddivisi e distinti:

1. **riusabilità:** i moduli possono essere riutilizzati anche in altre applicazioni qualora fosse necessario o utile;

2. **manutenibilità:** i moduli sono facilmente manutenibili, in quanto, lo sviluppatore sa esattamente dove risiede il codice da aggiornare o a cui apportare delle modifiche;

3. **estendibilità:** anche in questo caso, lo sviluppatore conoscendo già le varie funzionalità di ciascun modulo, può aggiungerne di nuove in modo mirato ad uno o più moduli mantenendo la consistenza tra i vari componenti e i rispettivi moduli oppure, in base alle sue esigenze, può semplicemente aggiungere nuovi moduli per implementare nuove funzionalità non ancora realizzate dal suo sistema.

4.2 Descrizione delle componenti

4.2.1 EventModule

Rappresentato in Figura 4.2:

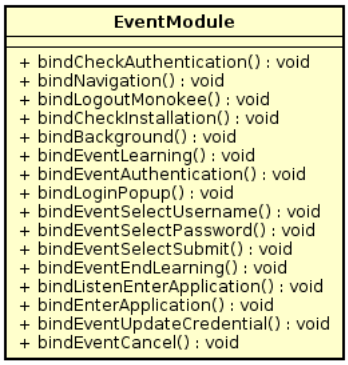


Figura 4.2: Modulo eventi

- **Descrizione:** modulo per la gestione degli eventi. Viene utilizzato per intercettare tutti gli eventi del sistema, quali:
  - eventi inviati da Monokee: di Single Sign-On, di learning, di logout da Monokee e di controllo dell'installazione del plug-in.
  - Eventi scatenati dall'utente durante l'interazione con il sistema: apertura di un nuovo tab durante la navigazione, selezione pulsante del plug-in nel browser, conferma selezione di: pagina di login, campo username, campo password e pulsante submit.
  - Eventi scatenati dall'utente in alcune interazioni con l'applicazione web: quando accede a quest'ultima manualmente, ovvero senza effettuare il Single Sign-On e quando tale accesso è avvenuto correttamente.
- **Dipendenze:**
  - SingleSignOnModule.
  - LearningModule.
  - NavigationModule.
  - RestModule.
  - DomManagement.
  - ViewModule.
  - LanguageModule.
- **Metodi:**
  - `bindCheckAuthentication(): void`
    Metodo che intercetta l'evento dell'utente di essere autenticato a Monokee oppure no, tramite una chiamata a quest'ultimo.
  - `bindNavigation(): void`
    Metodo che intercetta l'evento di apertura di un nuovo tab o pagina . Esso controlla se la pagina aperta deriva da un evento già scatenato di Single Sign-On, in tal caso effettua una chiamata a Monokee per ottenere la chiave di decrifratura e procedere con il Single Sign-On vero e proprio. Altrimenti, invoca Monokee per ricevere la matrice contente le pagine di login di tutte le applicazioni, e i rispettivi identificativi dei campi username, password e del pulsante di submit, che l'utente ha nel suo application broker per confrontarle con l'attuale URL_G.
  - `bindLogoutMonokee(): void`
    Metodo che intercetta l'evento di logout dell'utente da Monokee.
  - `bindCheckInstallation(): void`
    Metodo che intercetta l'evento inviato da Monokee di controllo dell'installazione. Il metodo, quindi, richiama `DomManagement.installed()` per impostare nel DOM_G della pagina corrente il valore `true` a un campo nascosto di controllo dell'installazione.

- `bindBackground(): void`
  Metodo che intercetta l'evento di click da parte dell'utente sul pulsante creato nel browser; esso invoca il metodo `ViewModule.createPopup()` per far visualizzare all'utente un popup per poter aggiungere l'attuale pagina di navigazione come applicazione nel suo application broker.
- `bindEventLearning(): void`
  Metodo per intercettare l'evento di learning inviato da Monokee. Effettua la redirect all'applicazione per la quale l'utente vuole effettuare il learning.
- `bindEventAuthentication(): void`
  Metodo che intercetta l'evento di Single Sign-On inviato da Monokee, il metodo salva sul database locale i dati necessari per il Single Sign-On, ovvero: URL_G della pagina di login, identificativo del campo di login e il login, identificativo del campo password e la password, identificativo del pulsante submit. Il metodo, poi, richiede la chiave di decifratura per decriptare tali dati.
- `bindLogin(): void`
  Metodo che intercetta l'evento di redirect all'applicazione web per cui si vuole effettaure il learning. Viene invocato il metodo `LearningModule.wizardLearning()` che crea la parte riguardante la pagina di login del wizard di learning.
- `bindEventSelectUsername(): void`
  Metodo che intercetta l'evento di conferma della pagina di login. Quindi, viene invocato il metodo `LearningModule.wizardLearning()` che crea la parte del wizard di learning riguardante la selezione del campo username.
- `bindEventSelectPassword(): void`
  Metodo che intercetta l'evento di conferma della selezione del campo username. Quindi, viene invocato il metodo `LearningModule.wizardLearning()` che crea la parte del wizard di learning riguardante la selezione del campo password.
- `bindEventSelectSubmit(): void`
  Metodo che intercetta l'evento di conferma della selezione del campo password. Quindi, viene invocato il metodo `LearningModule.wizardLearning()` che crea la parte del wizard di learning riguardante la selezione del submit.
- `bindEventEndLearning(): void`
  Intercetta l'evento di conclusione del wizard di learning da parte dell'utente. Il metodo effettua una chiamata a Monokee per richiedere la chiave di cifratura per cifrare e successivamente inviare i dati recuperati dal wizard.
- `bindListenEnterApplication(): void`
  Metodo per intercettare l'evento di accesso da parte dell'utente all'applicazione web senza effettuare il Single Sign-On. Il metodo chiama `NavigationModule.listenEnterPage()` il quale rimane in ascolto del click sul pulsante di accesso alla home page dell'applicazione.
- `bindEnterApplication(): void`
  Metodo per intercettare che l'accesso da parte dell'utente all'applicazione,

senza aver effettuato il Single Sign-On, sia avvenuto con successo. A questo punto il metodo, esegue due chiamate a Monokee: la prima per recuperare la matrice delle pagine di login delle varie applicazioni registrate nell'application broker dell'utente. La matrice contiene, inoltre, le informazioni dei domini, degli account dell'utente, del nome dell'applicazione, e i valori di login e password cifrati associati a ciascuna pagina di login. La seconda chiamata è per ottenere la chiave di decrifratura indispensabile per decrifrare i valori di login e password e poterli comparare con quelli usati per l'accesso dall'attuale applicazione web.

- `bindEventUpdateCredential(): void`
  Metodo che intercetta l'evento di aggiornamento delle credenziali di un'applicazione in uno specifico dominio o account. Il metodo richiama `NavigationModule.updateCredential()` passandogli i dati per l'aggiornamento.
- `bindCancel(): void`
  Metodo utilizzato per intercettare l'annullamento deciso dall'utente a procedere con il Single Sign-On oppure l'annullamento dell'aggiunta dell'attuale pagina al suo application broker durante la navigazione.

4.2.2 SingleSignOnModule

Rappresentato in Figura 4.3:

Figura 4.3: Modulo Single Sign-On

- **Descrizione:** modulo che effettua l'autenticazione delle applicazioni presenti nell'application broker dell'utente tramite Single Sign-On.
- **Dipendenze:** DomManagement.
- **Metodi:**
  - `openPageSso(string: keyDecrypting): void`
    Metodo che decripta l'URL$_G$ della pagina di login dell'applicazione ed effettua la redirect a quest'ultima.

    Argomenti: `keyDecrypting`: JSON$_G$ contenente la chiave di decifratura.
  - `ssoAuthentication(string: keyDecrypting): void`
    Metodo che effettua l'autenticazione all'appplicazione desiderata al posto dell'utente, ovvero il Single Sign-On. Recupera dal JSON$_G$ la chiave di decifra-

tura e dal database i dati necessari per effettuare l'autenticazione. Successivamente decripta i valori di login e password e invoca i metodi `DomManagement.compileFieldApp()` e `DomManagement.simulateSubmitApp()` per eseguire il Single Sign-On.

Argomenti: `keyDecrypting`: JSON_G contenente la chiave di decifratura.

4.2.3 LearningModule

Rappresentato in Figura 4.4:

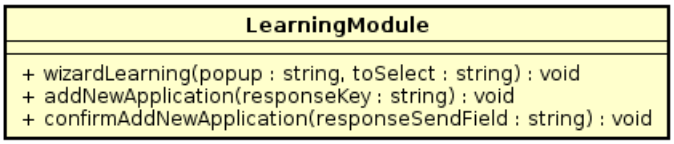


Figura 4.4: Modulo Learning

- **Descrizione:** modulo che si occupa di realizzare il learning per l'applicazione richiesta, ovvero: crea il wizard di learning per guidare l'utente nell'aggiunta di una nuova login applicativa al suo application broker e invia i dati recuperati dal wizard a Monokee per l'effettiva aggiunta dell'applicazione nell'application broker dell'utente.
- **Dipendenze:**
  - DomManagement.
  - RestModule.
  - ViewModule.
- **Metodi:**
  - `wizardLearning(string: popup, string: toSelect): void`
    Metodo per creare i popup del wizard di learning. Se l'utente deve selezionare i campi username e password o il submit viene invocato il metodo `DomManagement.learnField()`.

    Parametri:
    * `popup`: stringa HTML per creare il popup.
    * `toSelect`: stringa che indica quale elemento del DOM_G della pagina deve essere selezionato.
  - `addNewApplication(string: responseKey): void`
    Metodo che comunica a Monokee tutti i valori recuperati dal wizard avendoli prima cifrati.

Parametri: `responseKey`: JSON_G contenente la chiave di cifratura.

– `confirmAddNewApplication(string: responseSendField): void`
Metodo invocato in caso di esito positivo in risposta da Monokee per l'aggiunta della login applicativa richiesta dall'utente. Il metodo redirige l'utente ad una pagina di Monokee di aggiunta avvenuta.

Parametri: `responseSendField`: JSON_G contenente l'URL_G al quale redirigere l'utente.

4.2.4 NavigationModule

Rappresentato in Figura 4.5:

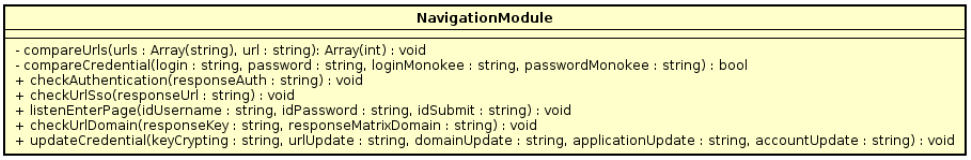


Figura 4.5: Modulo gestione navigazione

- **Descrizione:** modulo che si occupa di gestire la parte di navigazione dell'utente all'esterno di Monokee.
- **Dipendenze:**
  – RestModule.
  – DomManagement.
  – ViewModule.
  – LanguageModule.
- **Metodi:**
  – `compareUrls(Array(string): urls, string: url): Array(int)`
  Metodo privato che individua le occorrenze del parametro `url` nella lista di URL_G passata come primo parametro. Il metodo ritorna un array di interi dove ogni elemento è la posizione nella lista di URL_G nel caso in cui tale URL_G della lista corrisponda all'altro URL_G passato come parametro.

  Parametri:
  * `urls`: lista delle URL_G da comparare.
  * `url`: URL_G da ricercare all'interno della lista.

– `compareCredential(string: login, string: password, string: loginMonokee, string: passwordMonokee): boolean`
Metodo privato che permette di confrontare tra loro due coppie di valori login/password. Restituisce `true` se la prima coppia login/password corrisponde alla seconda, `false` altrimenti.

Parametri:

* `login, loginMonokee:` valori di login da confrontare.
* `password, passwordMonokee:` valori password da confrontare.

– `checkAuthentication(string: responseAuth): void`
Metodo che controlla se l'utente risulta autenticato a Monokee oppure no.

Parametri: `responseAuth:` JSON_G contenente lo stato dell'autenticazione dell'utente su Monokee.

– `checkUrlSso(string: responseUrl): void`
Metodo per far visualizzare all'utente, durante la navigazione all'esterno di Monokee, un popup di richiesta a proseguire con il Single Sign-On nel momento in cui esso naviga nella pagina di login di una delle applicazioni già presenti nel suo application broker. Il metodo recupera la lista delle pagine di login e la compara con l'attuale URL_G di navigazione invocando il metodo `compareUrls`, infine se esiste il match crea il popup tramite `ViewModule.createPopup()`.

Parametri: `responseUrl:` JSON_G contenente la lista delle pagine di login con i rispettivi identificativi dei campi username, password e del submit.

– `listenEnterPage(string: idUsername, string: idPassword, string: idSubmit): void`
Metodo che, durante la navigazione nella pagina di login di una delle applicazioni dell'utente, rimane in ascolto di un possibile accesso manuale all'applicazione da parte dell'utente, ovvero senza effettuare il Single Sign-On. Nel caso in cui l'accesso manuale dovesse avvenire, il metodo invoca `DomManagement.getValueApp()` per recuperare i valori di login e password dell'applicazione.

Parametri:

* `idUsername:` identificativo del campo username.
* `idPassword:` identificativo del campo password.
* `idSubmit:` identificativo del submit.

– `checkUrlDomain(string: responseKey, string: responseMatrixDomain): void`
Metodo che dopo l'accesso manuale da parte dell'utente ad una delle sue applicazioni, compara le attuali credenziali dell'applicazione con quelle presenti

nella matrice e per le quali l'applicazione è la stessa. Quest'ultima potrebbe essere associata a domini o account diversi dello stesso utente; il metodo, quindi, visualizza all'utente una lista delle possibili opzioni della stessa applicazione per cui è possibile aggiornare le credenziali con le attuali usate per l'accesso.

Parametri:

* `responseKey:` JSON_G contenente la chiave di decifratura con la quale decifrare i valori criptati di login e password contenuti nella matrice.
* `responseMatrixDomain:` JSON_G contenente la matrice con le pagine di login delle applicazioni, i rispettivi domini e account a cui sono associate e le credenziali criptate.

– `updateCredential(string: keyCrypting, string: urlUpdate, string: domainUpdate, string: applicationUpdate, string: accountUpdate): void`
Metodo che invia a Monokee i dati cifrati per effettuare l'aggiornamento delle credenziali di una delle applicazioni nell'application broker dell'utente. In particolare invia: l'URL_G, il dominio, l'account, il nome e le nuove credenziali dell'applicazione.

Parametri:

* `keyCrypting:` chiave per la cifratura.
* `urlUpdate:` URL_G della pagina di login dell'applicazione a cui aggiornare le credenziali.
* `domainUpdate:` dominio dell'applicazione a cui aggiornare le credenziali.
* `applicationUpdate:` nome dell'applicazione a cui aggiornare le credenziali.
* `accountUpdate:` account dell'applicazione a cui aggiornare le credenziali.

4.2.5 RestModule

Rappresentato in Figura 4.6:

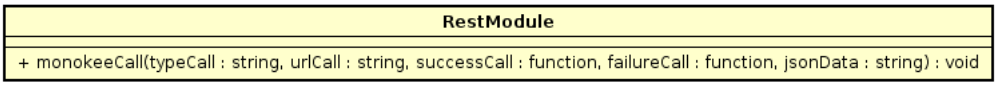


Figura 4.6: Modulo comunicazione con Monokee

- **Descrizione:** modulo che gestisce la comunicazione con Monokee, sia per i dati da inviare sia per riceverne.
- **Metodi:**

– `monokeeCall(string: typeCall, string: urlCall, function: successCall, function: failureCall, string: jsonData): void`
Metodo che effettua le chiamate a Monokee. Si occupa sia dell'invio dei dati sia della loro ricezione; inoltre, gestisce il successo o il fallimento della chiamata invocando il rispettivo metodo.

Parametri:

* `typeCall`: tipo della chiamata, per inviare o per ricevere.
* `urlCall`: URL_G di Monokee al quale effettuare la chiamata.
* `successCall`: metodo invocato all'esito positivo sia della chiamata che della risposta ottenuta.
* `failureCall`: metodo invocato al fallimento della chimata o all'esito negativo della risposta.
* `jsonData`: JSON_G contenente i dati da inviare.

4.2.6 DomManagement

Rappresentato in Figura 4.7:

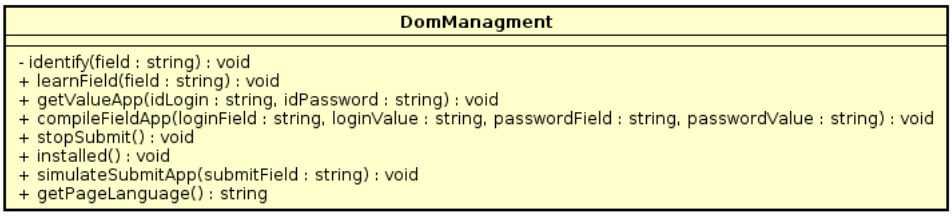


Figura 4.7: Modulo gestione DOM

- **Descrizione:** modulo che permette la gestione del DOM_G delle pagine delle applicazioni web. Viene utilizzato, in questo caso, per manipolare le pagine di login delle applicazioni.
- **Metodi:**
  – `identify(string: field): void`
  Metodo privato per capire se il campo richiesto contiene un attributo valido, ovvero che abbia almeno l'attributo `id` o `name`.

  Parametri: `field`: campo da identificare.
  – `learnField(string: field): void`
  Metodo che al click da parte dell'utente su un campo lo evidenzia e ne riconosce l'identificativo che salva nel database locale.

  Parametri: `field`: campo da identificare.

- `getValueApp(string: idLogin, string: idPassword): void`
  Metodo per recuperare i valori di login e password dai rispettivi campi della pagina e successivamente salvarli nel database locale.

  Parametri:
  - `idLogin:` identificativo del campo login della pagina.
  - `idPassword:` identificativo del campo password della pagina.
- `compileFieldApp(string: loginField, string: loginValue, string: passwordField, string: passwordValue): void`
  Metodo che permette di compilare automaticamente i campi username e password della pagina di login con i rispettivi valori passati per parametro.

  Parametri:
  - `loginField:` identificativo del campo username della pagina.
  - `loginValue:` valore da inserire nel campo username della pagina.
  - `passwordField:` identificativo del campo password della pagina.
  - `passwordValue:` valore da inserire nel campo password della pagina.
- `simulateSubmitApp(string: submitField): void`
  Metodo per effettuare l'accesso all'applicazione in modo automatico, ovvero simulando il click che farebbe l'utente.

  Parametri: `submitField:` identificativo del submit della pagina di login.
- `stopSubmit(): void`
  Metodo per disabilitare il pulsante di accesso all'applicazione web al click da parte dell'utente. Viene utilizzato per la selezione del submit durante il wizard di learning.
- `installed(): void`
  Metodo per impostare il valore `true` al campo hidden della pagina per il controllo dell'installazione.
- `getPageLanguage(): string`
  Metodo per recuperare la lingua dell'attuale pagina web.

4.2.7 ViewModule

Rappresentato in Figura 4.8:

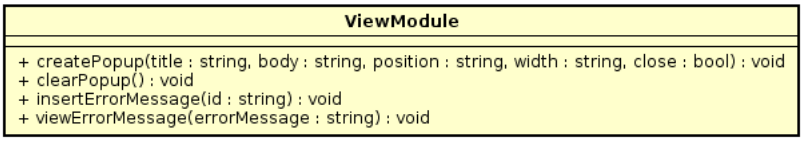


Figura 4.8: Modulo per la visualizzazione

- **Descrizione:** modulo che permette l'interazione con l'utente tramite la visualizzazione di messaggi e popup.
- **Metodi:**
  - `createPopup(string: title, string: body, string: position, string: width, bool: close): void`
    Metodo che permette la visualizzazione di un popup da parte dell'utente.

    Parametri:
    * `title`: titolo del popup.
    * `body`: corpo del popup.
    * `position`: posizione del popup nel browser.
    * `width`: larghezza in pixel del popup.
    * `close`: valore booleano per permettere la chiusura del popup da parte dell'utente.
  - `clearPopup(): void`
    Metodo per permettere la chiusura automatica dell'attuale popup.
  - `insertErrorMessage(string: id): void`
    Metodo per permettere la visualizzazione di un messaggio d'errore all'interno del popup.

    Parametri: `id`: identificativo del messaggio all'interno dell'attuale popup.
  - `viewErrorMessage(string: errorMessage): void`
    Metodo per la visualizzazione di un messaggio d'errore nel browser.

    Parametri: `errorMessage`: messaggio d'errore da visualizzare.

4.2.8 LanguageModule

Rappresentato in Figura 4.9:

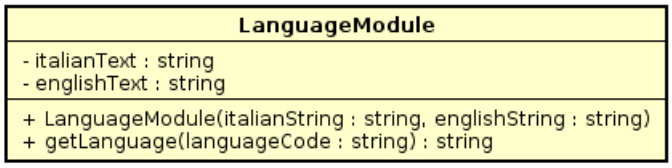


Figura 4.9: Modulo gestione lingua

- **Descrizione:** modulo per la gestione della lingua del testo visualizzato. La scelta della lingua da usare dipende dalla lingua della pagina web; sono disponibili le seguenti lingue: italiano, inglese.

- **Attributi:**
  - `italianText:` variabile contenente il testo in italiano.
  - `englishText:` variabile contenente il testo in inglese.
- **Metodi:**
  - `LanguageModule(string: italianString, string: englishString)`
    Costruttore.
  - `getLanguage(string: language): string`
    Metodo che restituisce il testo nella lingua della pagina web. La lingua di default è l'inglese.

Capitolo 5

Implementazione

In questo capitolo vengono analizzati maggiormente in dettaglio alcuni aspetti implementativi che ho utilizzato per produrre il software richiesto e vengono illustrati alcuni *screenshot* del prodotto finito. In particolare, nella Sezione 5.1 vengono descritte in modo più approfondito le implementazioni di: Module Pattern, eventi e comunicazione con Monokee; mentre nella Sezione 5.2 viene presentato il modo con cui si provvede alla sicurezza dei dati scambiati tra il sistema realizzato e Monokee. Infine, nella Sezione 5.3 vengono illustrati alcuni *screenshot* del software realizzato e, in questo caso, testato tramite l'applicazione *ADProject* (prodotto di proprietà di Athesys S.r.l).

5.1 Plug-in

Come ho già descritto nella Sezione 4.1, per la progettazione ho utilizzato il Module Pattern. In Javascript, il Module Pattern [9] viene utilizzato per simulare il concetto di classi in grado di contenere metodi pubblici e privati e variabili all'interno di un singolo oggetto, evitando così il proliferare di variabili globali che tale linguaggio permette in maniera cospicua. Per realizzare l'incapsulamento di variabili e metodi privati viene utilizzato il meccanismo di *closure* che protegge le variabili all'interno dell'oggetto in cui sono definite evitando, inoltre, eventuali collisioni con altre variabili globali con lo stesso nome. In Javascript non si possono dichiarare modificatori di accesso per le variabili, ma grazie al meccanismo di *closure*, variabili e metodi privati sono disponibili solo all'interno del modulo in cui sono definiti, mentre i metodi definiti nell'oggetto di ritorno sono pubblici e quindi accessibili da tutti.

Per quanto riguarda la gestione degli eventi ho utilizzato la libreria di Crossrider, descritta nella Sezione 2.1.1, `CrossriderAPI.bindExtensionEvent` per intercettare gli eventi inviati da Monokee o scatenati a seguito dell'interazione dell'utente con il plug-in e `CrossriderAPI.fireExtensionEvent` per lanciare gli eventi in concomitanza dell'azione dell'utente sui vari popup visualizzati dal plug-in. In alcune occasioni, ho utilizzato il database locale di Crossrider `appAPI.db`, per esempio, per salvare i dati di accesso a

un'applicazione, che altrimenti, al suo accesso non avrei più avuto.

Per la parte di comunicazione con il back-end di Monokee ho adoperato Ajax, come descritto nella Sezione 2.2.2, per effettuare chiamate asincrone e avere la possibilità, una volta ottenuta la risposta dal server, di aggiornare dinamicamente il popup visualizzato dall'utente. Ajax utilizza un oggetto XMLHttpRequest$_G$ per inviare e ricevere dati dal server. Le chiamate Ajax effettuate a Monokee sono di tipo *POST* o *GET*, inviano o ricevono dati in formato JSON$_G$ e permettono, al successo o al fallimento della chiamata, di richiamare funzioni di *callback*. Inoltre, vorrei precisare che nell'URL$_G$ delle chiamate a Monokee è già insito il cookie dell'utente, nel caso esso sia già autenticato a Monokee, ed è quindi compito di quest'ultimo, a ogni chiamata, controllare che il cookie sia autentico per poter effettuare le funzionalità del plug-in.

Infine, ho utilizzato jQuery, descritta nella Sezione 2.2.1 per recuperare gli elementi del DOM$_G$ e i loro valori in modo semplice e veloce e avere una gestione degli eventi delle pagine più agevole tramite una semplice funzione di libreria. Ho impiegato l'HTML, linguaggio descritto nella Sezione 2.3 per la creazione del corpo dei vari popup del plug-in, mentre il CSS descritto nella Sezione 2.3, per evidenziare i campi della pagina dell'applicazione web di volta in volta selezionati dall'utente.

5.2 Sicurezza

Per la parte relativa alla sicurezza dei dati inviati e ricevuti durante le comunicazioni con Monokee, quindi per la loro cifratura e decifratura, ho utilizzato l'algoritmo AES, come richiesto dall'azienda. Per la sua implementazione ho usato CryptoJs [10], codice realizzato da Google Inc.:

> "CryptoJS è una collezione crescente di algoritmi crittografici standard e sicuri implementati in Javascript utilizzando *best practices* e *patterns*."

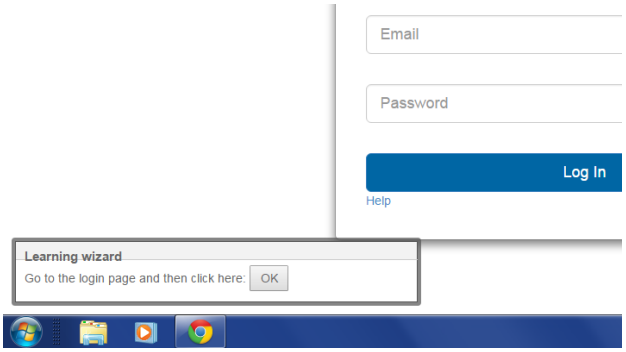
AES [11], Advanced Encryption Standard, conosciuto anche con il nome di Rijndael dall'unione dei cognomi dei due ideatori Rijmen e Daemen, è un algoritmo cifrario simmetrico a blocchi. Simmetrico significa che usa la stessa chiave sia per la cifratura che per la decrifratura, mentre a blocchi significa che prende un blocco di n bit del testo in chiaro e lo trasforma in n bit cifrati utilizzando la chiave. AES adopera blocchi e chiavi con dimensioni da 128 a 256 bit in passi di 32 bit e le lunghezze di chiave e blocco possono essere scelte indipendentemente. L'algoritmo utilizza sostituzioni e permutazioni in diversi round e il numero di quest'ultimi è dato dalla dimensione di chiave e blocco.

Come richiesto dall'azienda, il plug-in deve comunicare con Monokee per ottenere la chiave di cifratura (e decifratura), questo però potrebbe rappresentare un rischio per la sicurezza, in quanto, un intruso potrebbe imposessarsi della chiave. Va, comunque, considerato che la comunicazione tra Monokee e il plug-in avviene in HTTPS, ovvero utilizzando un protocollo per il trasferimento dei dati riservati, il quale crea un canale di

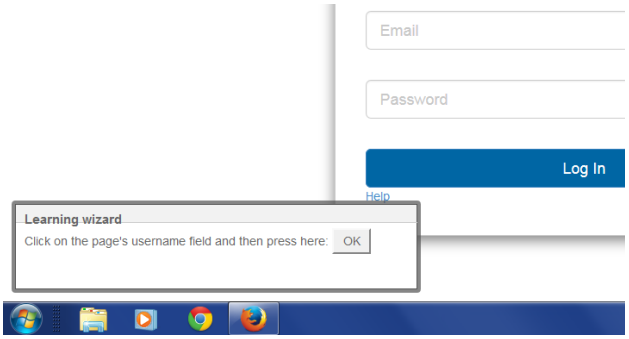
comunicazione sicuro tra server e client facendo uso di algoritmi crittografici asimmetrici, ossia attraverso lo scambio di certificati.

5.3 Prodotto realizzato

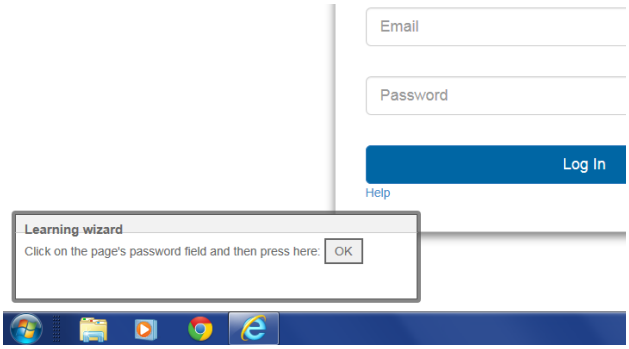
Di seguito vorrei riportare alcuni *screenshot* del prodotto realizzato, in particolare: il wizard di learning in Figura 5.1 che, ho voluto presentare aggiungendo l'applicazione all'application broker tramite: Chrome, Firefox e Internet Explorer per mostrarne il corretto funzionamento in tutti e tre i browser. Non è stato possibile dimostrarlo per il browser Safari per i motivi descritti nel Capitolo 6.



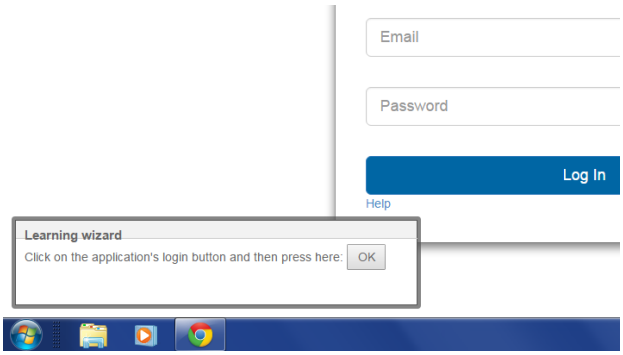
(a) Wizard di learning: step 1 in Chrome.



(b) Wizard di learning: step 2 in Firefox.



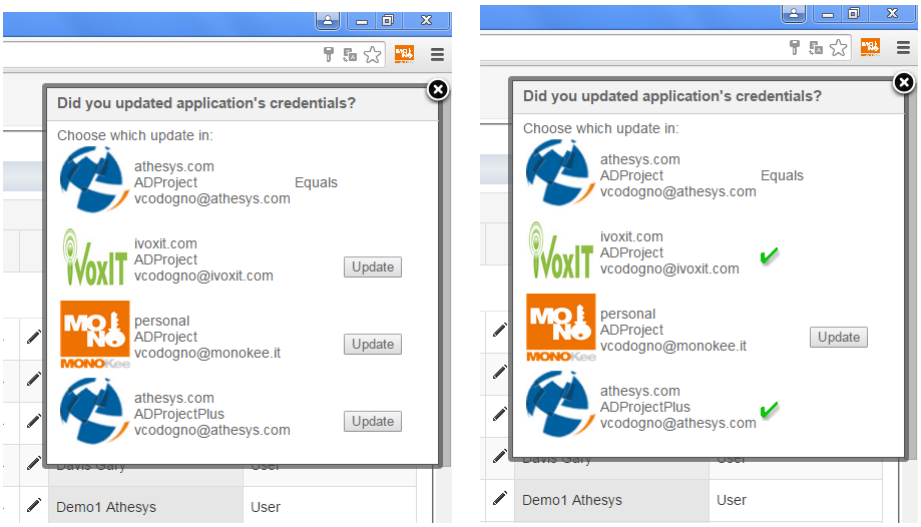
(c) Wizard di learning: step 3 in Internet Explorer.



(d) Wizard di learning: step 4 in Chrome.

Figura 5.1: Wizard di learning

In Figura 5.2, invece, viene presentata la scelta di quali aggiornamenti effettuare una volta eseguito l'accesso manuale all'applicazione e il loro effettivo aggiornamento su Monokee:



(a) Lista dei possibili aggiornamenti delle credenziali.

(b) Visualizzazione degli aggiornamenti effettuati.

Figura 5.2: Aggiornamento delle credenziali

Capitolo 6

Verifica e Validazione

Per la parte di *testing* del software ho utilizzato QUnit, descritto nella Sezione 2.2.5, che mi ha permesso di creare i test di unità durante lo sviluppo del software. Ho scelto questo framework$_G$, in quanto consigliato dal team di sviluppo di Crossrider nel loro blog (`https://getsatisfaction.com/crossrider/topics/what_is_available_js_testing_frameworks_with_crossrider`) e, inoltre, perchè mi ha consentito di testare facilmente: il codice jQuery di manipolazione del DOM$_G$ e la gestione degli eventi sia quelli scatenati da azioni dell'utente sia quelli generati con il *Fire event* di Crossrider. I test di unità realizzati non coprono la totalità delle funzioni Javascript codificate, in quanto non è stato possibile realizzarli tutti per ragioni di tempo, parte del quale anche speso per capire come calare il framework$_G$ di testing all'interno dell'estensione. Infatti, normalmente, per utilizzare QUnit basterebbe includere due file nella pagina HTML che si vuole testare e poi è il *Test Runner* che si occupa di eseguire i test per quel specifico componente e produrre la pagina contenente i risultati di quest'ultimo; in un'estensione, invece, non è possibile inserire i due file richiesti in una pagina HTML specifica ed eseguirne i test solamente di quest'ultima, perchè l'estensione funziona nel contesto di ogni pagina aperta del browser.

Per accertare la completa copertura dei requisiti software, ho eseguito i test di sistema per ciascun browser richiesto: Chrome versione 44.0, Firefox versione 38.0.5 e Internet Explorer versione 11, verificandone il corretto funzionamento del plug-in per ognuno. Purtroppo, non è stato possibile installare e quindi testare il plug-in nel browser Safari; infatti per l'installazione è necessario registrarsi come sviluppatore su *Safari Developer Center* e successivamente ottenere un certificato per l'estensione. Tale pratica è stata avviata dall'azienda ma la sua conclusione non è arrivata in tempo utile per l'attività di stage.

Tutti i requisiti obbligatori sono stati comunque soddisfatti, mentre non è stato soddisfatto l'unico requisito opzionale che consisteva nel rendere invisibile all'utente il processo di Single Sign-On.

Capitolo 7

Conclusioni

Il periodo di stage è stato un'esperienza di grande valore che mi ha permesso di crescere ulteriormente, sia come persona sia nelle mie competenze acquisite nel mondo accademico. Assieme alle persone che lavorano ad Athesys e sopratutto con i membri del team di sviluppo (tra cui anche altri stagisti) abbiamo instaurato un bellissimo rapporto di amicizia e collaborazione che ci ha portato a discutere insieme e a confrontarci sulle soluzioni per cercare di realizzare al meglio Monokee. Inoltre, a livello di conoscenze ho potuto apprendere in modo approfondito il funzionamento di un sistema di Access Management e per la parte specifica a cui si riferiva il mio progetto ho potuto studiare a fondo il linguaggio Javascript e il suo utilizzo; inoltre, ho acquisito numerose conoscenze, quali: l'utilizzo delle librerie jQuery e delle tecnologie Crossrider, Ajax e QUnit.

Le difficoltà maggiori che ho riscontrato sono state sopratutto nell'utilizzo del linguaggio Javascript (di cui avevo pochissime conoscenze) e della tecnologia Crossrider. Infatti, quest'ultimo non dispone di un'architettura su cui basarsi ma espone semplicemente *Scope* e librerie da utilizzare. Inoltre, ho trovato difficoltà nello sviluppo per Internet Explorer perchè, per quanto Crossrider permetta lo sviluppo dello stesso codice per tutti e quattro i browser richiesti, Internet Explorer segnalava errori di compilazione che, invece, nei browser Chrome e Firefox non venivano segnalati. L'errore da parte mia è stato quello di lavorare e quindi testare, inizialmente, il codice su Chrome e solo successivamente su Firefox e Internet Explorer, se invece avessi iniziato da quest'ultimo avrei risolto da subito i problemi riscontrati.

Il tutor aziendale si è, comunque, dimostrato soddisfatto del prodotto realizzato e di come sono state svolte le varie attività di stage. Questo è stato possibile anche grazie al percorso accademico frequentato e ai relativi corsi sostenuti, in particolare quello di *Ingegneria del Software* che attraverso la realizzazione di un progetto aziendale mi ha fornito le capacità di: analisi di un problema, progettazione e implementazione di una soluzione e gestione delle tempestiche per le consegne. Tutte competenze che si sono rivelate indispensabili per realizzare al meglio il progetto richiesto, nei tempi prestabiliti e con soddisfazione da parte dell'azienda.

Ringraziamenti

Vorrei innanzitutto ringraziare il mio tutor aziendale, Roberto Griggio, per la sua disponibilità nel seguirmi, nel spiegarmi anche più volte gli argomenti e per le numerose competenze che mi ha trasmesso; vorrei ringraziare, inoltre, tutte le persone che lavorano ad Athesys S.r.l, che in questi due mesi, mi hanno fatto sentire parte integrante del loro gruppo.

Un grazie di cuore va ai miei amici, in particolare Silvia e Roberta, che in questi anni ho spesso trascurato per dedicarmi allo studio ma, nonostante ciò, hanno sempre saputo darmi, allo stesso tempo, il loro appoggio e la loro gioisità come solo i veri amici sanno fare.

Un immenso grazie all'amore della mia vita per la sua pazienza e perchè senza di lui non ce l'avrei mai fatta a superare i numerosi ostacoli di questo cammino, sempre pronto a credere in me e a incoraggiarmi verso gli obiettivi anche quando quest'ultimi mi sembravano davvero lontani.

Il grazie più importante va ai miei genitori per aver permesso questa Laurea, per avermi sempre sostenuto e incoraggiato in qualsiasi mia scelta ma, sopratutto per avermi reso la persona che sono adesso, la persona che ha raggiunto questo primo grande traguardo.

Grazie di cuore.

Settembre 2015

Valentina

Glossario

A

Agile È un metodo di sviluppo del software che nasce negli anni '90 in contrapposizione a modelli tradizionali troppo rigidi. Si basa su quattro principi fondamentali: gli individui e le interazioni con essi sono più importanti dei processi e degli strumenti, è meglio avere del software funzionante piuttosto che della documentazione esaustiva ma del codice che non funziona, è molto più importante la collaborazione con il cliente che la negoziazione dei contratti e infine, rispondere al cambiamento è più rilevante rispetto a seguire un piano.

API Acronimo di *Application Programming Interface*, indica un insieme di procedure che i programmatori possono utilizzare all'interno del loro codice. Offrono funzionalità già pronte all'uso e, solitamente, raggruppano un set di metodi ciascuno con la relativa spiegazione della sua funzionalità, dei vari parametri e del valore di ritorno.

D

DOM Acronimo di *Document Object Model*, è uno standard ufficiale del World Wide Web Consortium (W3C) per la rappresentazione di documenti, strutturati in modo da essere indipendenti sia dalla lingua che dalla piattaforma. Esso permette a programmi e script di accedere e aggiornare il contenuto, la struttura e lo stile dei documenti.

F

Framework È un'architettura logica di supporto che aiuta i programmatori nello sviluppo di software e nella progettazione di un'architettura riusabile.

I

IDaaS È un servizio cloud-based che fornisce funzionalità di Identity and Access Management ad un sistema nel cloud e/o in locale.

J

JSON Acronimo di *JavaScript Object Notation*, è un formato semplice e adatto allo scambio di dati tra applicazioni client-server. Prende origine dall'*Object Literal Notation* di Javascript, ovvero dalla definizione di un oggetto come coppie nome/valore separate tra loro da una virgola e racchiuse all'interno di parentesi graffe.

S

SAML Acronimo di *Security Assertion Markup Language*, è uno standard per lo scambio di dati di autenticazione e autorizzazione tra domini di sicurezza distinti, tipicamente tra un identity provider e un service provider. SAML cerca di risolvere il problema dell'esecuzione del Single Sign-On tra entità appartenenti a domini di sicurezza diversi.

U

URL Acronimo di *Uniform Resource Locator*, è un sequenza di caratteri che identifica univocamente l'indirizzo di una risorsa in Internet, solitamente presente in un server, redendola accessibile ad un client che ne fa richiesta tramite un browser.

X

XML Acronimo di *eXtensible Markup Language*, è un metalinguaggio per la definizione di linguaggi di markup. È basato su un meccanismo sintattico che consente di definire e controllare il significato degli elementi contenuti in un documento o in un testo.

XMLHttpRequest È un oggetto che rappresenta una richiesta HTTP tra il client e il server e che può essere controllata da Javascript. Con questo meccanismo è possibile, in qualsiasi momento, accedere al server senza avere bisogno del caricamento della pagina.

Bibliografia

[1] scrumguides.org, *The Scrum Guide.* http://www.scrumguides.org/scrum-guide.html

[2] Athesys s.r.l, *Identity and Access Management.* http://www.athesys.it/?page_id=143

[3] Gartner Group, *IT Glossary: Identity and Access Management.* http://www.gartner.com/it-glossary/identity-and-access-management-iam

[4] Pierluigi Paganini, *Single Sign-On: introduzione.* http://www.firmadigitalefacile.it/single-sign-on-introduzione/

[5] crossrider.com, *Crossrider FAQ.* http://crossrider.com/pages/faq

[6] Jesse James Garrett, *Ajax: A New Approach to Web Applications.* http://www.adaptivepath.com/ideas/ajax-new-approach-web-applications/

[7] Scott Chacon e Ben Straub, 2014 Apress *Pro Git* Capitolo: 1.3 Per iniziare - Basi di Git. https://git-scm.com/book/it/v1/Per-Iniziare-Basi-di-Git

[8] Anthony Colangelo, *Il design del codice: organizzare JavaScript.* http://www.italianalistapart.com/articoli/91-numero-75-5-giugno-2013/383-il-design-del-codice-organizzare-javascript

[9] Add Osmani, 2015 O'Reilly Editore. *Learning JavaScript Design Patterns* Capitolo: The Module Pattern. http://addyosmani.com/resources/essentialjsdesignpatterns/book/

[10] Google Inc., *crypto-js: JavaScript implementations of standard and secure cryptographic algorithms* https://code.google.com/p/crypto-js/

[11] Andrew S. Tanenbaum e David J. Wetherall, 2011 Pearson *Reti di calcolatori* Capitolo 8: Sicurezza delle reti, pag 747-748.