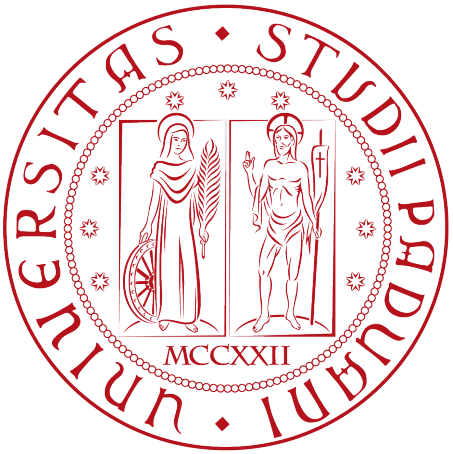
Università degli Studi di Padova

Dipartimento di Matematica

Corso di Laurea in Informatica



Sviluppo di un modulo in Alfresco, sistema di gestione della conoscenza

Tesi di laurea triennale

Relatore

Prof. Gaggi Ombretta

Laureando

Trevisan Davide

Anno Accademico 2016-2017

Abstract

Scopo di questa tesi di laurea è esporre il lavoro svolto dal laureando Trevisan Davide durante lo stage di trecentoventi ore presso l'azienda Ennova Research SRL (che nel presente documento spesso verrà abbreviata in "azienda") con sede in Venezia-Mestre. Il progetto di stage si è incentrato sullo sviluppo di alcune funzionalità connesse al sistema di gestione della conoscenza che l'azienda utilizza per gestire la documentazione relativa ai progetti assegnati dalle aziende clienti e monitorare l'operato dei dipendenti. Il sistema si basa sulla piattaforma Alfresco, un noto e molto utilizzato KMS, la quale offre una SDK (che verrà descritta estensivamente ed esaustivamente in seguito) lanciabile in maniera autonoma tramite Maven che consente lo sviluppo di moduli per personalizzare e ottimizzare la piattaforma a proprio piacimento. Nella realizzazione del progetto è stato possibile, pertanto, approfondire le potenzialità, i difetti e le caratteristiche della piattaforma Alfresco nonché le fasi che hanno portato alla realizzazione di moduli per aggiungere ad Alfresco nuove funzionalità e una maggiore customizzazione del suo aspetto. Tutte le fasi, le problematiche e quanto prodotto durante il progetto sarà accuratamente esposto nei capitoli che compongono la presente tesi.

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Introduzione

In questo capitolo verrà brevemente esposto il contesto in cui si è svolto lo stage, descrivendo le motivazioni che hanno spinto l'azienda a proporre questo stage

1.1 L'azienda

Ennova Research srl è un'azienda che opera nel settore ICT e realizza soluzioni informatiche altamente tecnologiche ed affidabili, che le permettono di agire con successo in settori di business come quello della Pubblica Amministrazione, delle grandi Corporazioni Bancarie, delle Multinazionali ICT e della Grande Distribuzione. È partner di grandi attori del mercato nazionale e internazionale quali Engineering, Toshiba, EMC, HP, Novell, Nvidia, ed altri . Ennova Research si distingue anche nel campo delle tecnologie open source utilizzate per la realizzazione di soluzioni multimediali avanzate destinate ai mercati B2C e B2B e di applicativi software destinati al settore del mobile, ad esempio Slash, che sfrutta le Application Program Interface di Twitter. L'azienda inoltre investe molto in ricerca e sviluppo ed è sempre pronta ad esplorare nuove tecnologie.

1.2 Lo stage

Il progetto di stage svoltosi all'interno dell'azienda Ennova Research è principalmente consistito nello sviluppo di moduli per Alfresco, che è il KMS che l'azienda ha incominciato da relativamente poco ad utilizzare e a cui intende sviluppare alcune funzionalità, quali ad esempio la gestione dei clienti e dei progetti, che al momento è svolta con l'ausilio di altri applicativi, e la rendicontazione, svolta anch'essa mediante l'utilizzo di altri applicativi. L'azienda tuttavia si pone l'obiettivo di creare non solo un prodotto necessario alle necessità interne, ma anche un prodotto vendibile e che generi profitto attraverso la vendita dello stesso ad aziende clienti. Il progetto è stato denominato Coral Tree e lo stage si è posto l'obiettivo di iniziare a porre le basi per questo progetto, attraverso la creazione di un modulo che introduca il nuovo tema per dare l'aspetto desiderato al prodotto e la creazione dei primi moduli che introducano le funzionalità precedentemente citate.

1.3 Organizzazione del testo

Il secondo capitolo descrive le tecnologie e gli strumenti utilizzati durante lo stage.

Il terzo capitolo descrive brevemente le condizioni e le metodologie utilizzate per lo stage.

Il quarto capitolo desctrive brevemente come creare un modulo e l'architettura di Alfresco.

i tre capitoli successivi descriveranno in maniera separata i vari moduli sviluppati, dato che il modulo che aggiunge il nuovo tema differisce sostanzialmente nella sua realizzazione dai moduli che aggiungono funzionalità nel senso stretto. Si hanno quindi i capitoli

il modulo tema ;

il modulo clienti ;

il modulo progetti ;

Capitolo 2

tecnologie e strumenti utilizzati

In questo capitolo saranno esposte le principali tecnologie utilizzate e i principali strumenti che sono stati utilizzati per portare a compimento il progetto assegnato

2.1 Tecnologie utilizzate

In seguito verranno descritte più nel drttaglio le tecnologie apprese o particolarmente approfondite durante lo svolgimento dello stage, che vanno ad aggiungersi a quelle pregresse, come ad esempio l'HTML e il CSS, già noti e ampiamente utilizzati nel corso del progetto.

2.1.1 Linguaggi di Programmazione

Java

Java è un linguaggio di programmazione orientato agli oggetti a tipizzazione statica, specificatamente progettato per essere il più possibile indipendente dalla piattaforma di esecuzione. Uno dei principi fondamentali del linguaggio è espresso dal motto write once, run anywhere: il codice compilato che viene eseguito su una piattaforma non deve essere ricompilato per essere eseguito su una piattaforma diversa. Il prodotto della compilazione è infatti in un formato chiamato bytecode che può essere eseguito da una qualunque implementazione di un processore virtuale detto Java Virtual Machine. Alfresco usa Java per il lato backend, soprattutto per i webscript, per i quali è stato largamente utilizzato assieme alla sua API.

Javascript

JavaScript è un linguaggio di scripting orientato agli oggetti e agli eventi, comunemente utilizzato nella programmazione Web lato client per la creazione, in siti web e applicazioni web, di effetti dinamici interattivi, tramite funzioni di script invocate da eventi innescati, a loro volta, in vari modi dall'utente sulla pagina web in uso. Tali funzioni di script possono essere opportunamente inserite in file HTML, in pagine JSP o in appositi file separati con estensione .js, poi richiamati nella logica di business. Nel progetto esso è stato usato per popolare dinamicamente le pagine di cui si sono composti i vari moduli e per eseguire i controlli sui form una volta inviate le richieste.

2.1.2 Formato per l'interscambio di dati

JSON

JSON (JavaScript Object Notation) è un formato di scambio dati leggero e facile da leggere e scrivere sia per macchine che per umani, si basa su un sottoinsieme del linguaggio di programmazione JavaScript, anche se ne è indipendente, ed inoltre è un formato di testo completamente indipendente da qualsiasi linguaggio. Per queste caratteristiche è un linguaggio di scambio dati ideale. JSON è costruito su due strutture:

* Una collezione di coppie nome/valore, spesso realizzato come un array associativo;

* Un elenco ordinato di valori, spesso realizzato come una lista.

Nel progetto esso è stato usato per poter scambiare i dati tra il lato frontend e il lato backend attraverso il passaggio di oggetti JSON tramite AJAX

2.1.3 Tecnica per l'interscambio di dati

AJAX

AJAX, acronimo di Asynchronous Javascript And XML, è una tecnica di sviluppo software per la realizzazione di applicazioni web interattive. Lo sviluppo di applicazioni HTML con AJAX si basa su uno scambio di dati in background fra web browser e server, che consente l'aggiornamento dinamico di una pagina web senza esplicito ricaricamento da parte dell'utente. AJAX è asincrono, nel senso che i dati extra sono richiesti al server e caricati in background senza interferire con il comportamento della pagina esistente, anche se è possibile, ma sconsigliato, effettuare richieste sincrone. Normalmente le funzioni richiamate sono scritte con il linguaggio JavaScript. Tuttavia, e a dispetto del nome, l'uso di JavaScript e di XML non è obbligatorio, come non è detto che le richieste di caricamento debbano essere necessariamente asincrone. Nel progetto esso è stato usato per mettere in comunicazione il lato frontend e il lato backend, grazie anche all'utilizzo delle librerie AJAX messe a disposizione dell'API Javascript di Alfresco.

2.1.4 Framework

Spring

Spring è un framework per realizzare applicazioni web basate sul Modello MVC avendo come punti di forza l'inversion of control (tramite dependency injection) e la aspect oriented programming. Esso si occupa di mappare metodi e classi Java con determinati url, di gestire differenti tipologie di "viste" restituite al client, di realizzare applicazioni internazionalizzate e di gestire i cosiddetti temi per personalizzare al massimo l'esperienza utente. Questo framework è strutturato a livelli, e permette di scegliere quale dei suoi componenti usare, fornendo nello stesso momento un framework di ottima qualità per lo sviluppo di applicazioni distribuite. Questa architettura a livelli consiste in diversi moduli (o componenti) ben definiti, ognuno dei quali può rimanere da solo o essere implementato con altri. Su questo framework si basa la struttura di base di Alfresco.

Jquery

JQuery è un framework nato con il preciso intento di rendere il codice più sintetico e di limitare al minimo l'estensione degli oggetti globali per ottenere la massima compatibilità con altre librerie. Grazie a questo principio jquery è in grado di offrire un'ampia gamma di funzionalità, che vanno dalla manipolazione degli stili CSS e degli elementi HTML, agli effetti grafici, per passare a comodi metodi per chiamate AJAX cross-browser. Nel progetto esso è stato utilizzato per la creazione di popup ed effetti per l'utente.

Alfresco Surf

Alfresco Surf è un framework messo a disposizione della piattaforma Alfresco con cui è possibile creare interfaccia grafiche, modelli e componenti sfruttando scripts server-side e templates. Anche se Alfresco stesso sta per abbandonarlo in favore di Angular2, esso è ancora il framework con cui viene distribuito il lato frontend nelle distribuzioni di Alfresco.

2.1.5 Protocolli di servizi

LDAP

LDAP è un acronimo che sta per LIGHTWEIGHT DIRECTORY ACCESS PROTOCOL. Come suggerisce il nome stesso, è un protocollo leggero per accedere ai servizi di directory, basati sul protocollo X.500. LDAP opera su TCP/IP o su altre connessioni orientate ai servizi di trasferimento. LDAP nasce per sostituire DAP in quanto molto oneroso dal punto di vista dell'impiego delle risorse ed è basato sul modello client-sever: un client LDAP invia una richiesta ad un server LDAP, che processa la richiesta ricevuta, accede eventualmente ad un directory database e ritorna dei risultati al client. Il modello di informazioni di LDAP è basato sulle entry. Un'entry è una collezione di attributi aventi un unico nome globale: il Distinguished Name (DN). Il DN è usato per riferirsi ad una particolare entry, senza avere ambiguità. Ogni attributo dell'entry ha un tipo ed uno o più valori. In LDAP, le entry di una directory sono strutturate come in una struttura gerarchica di un albero. L'azienda fa uso di questo protocollo per la profilatura degli utenti in virtù della sua ottima integrazione con Alfresco, e quindi di conseguenza si è dovuta utilizzare questa tecnologia per avere accesso ai dati degli utenti registrati

2.1.6 Data Base Management System

PostgreSQL

PostgreSQL è un completo DBMS ad oggetti rilasciato con licenza libera; spesso viene abbreviato come "Postgres". PostgreSQL è una reale alternativa sia rispetto ad altri prodotti liberi come ad esempio MySQL, con i quali però comunque condivide molti aspetti del suo linguaggio, che a quelli a codice chiuso come ad esempio Oracle, poichè offre caratteristiche uniche nel suo genere che lo pongono per alcuni aspetti all'avanguardia nel settore dei database. È il DBMS usato in maniera predefinita da Alfresco e per questo motivo è stato scelto per questo progetto.

2.2 Strumenti utilizzati

2.2.1 Strumenti di supporto all'attività di Codifica

Eclipse

Eclipse è un ambiente di sviluppo integrato multi-linguaggio e multipiattaforma. Ideato da un consorzio di grandi società quali Ericsson, HP, IBM, Intel, MontaVista Software, QNX, SAP e Serena Software, chiamato Eclipse Foundation. Eclipse è un software libero distribuito sotto i termini della Eclipse Public License ed è lo strumento che viene imposto dall'azienda ai suoi dipendenti, in quanto sofware libero, quindi la sua adozione per questo progetto è stata scontata. Questo IDE è stato utilizzato per l'importazione di un progetto allo scopo di estendere la piattaforma di Alfresco. Le funzionalità presentate da questo IDE in termini di aiuti al programmatore e la sua integrazione con Maven hanno permesso uno sviluppo rapido ed efficiente dei plugin.

2.2.2 Strumenti per il versionamento

Git

Il sistema di versioning adottato per questo progetto e in generale utilizzato dall'azienda è Git, che negli ultimi anni si è affermato come uno dei migliori sistemi di controllo di versione. Le caratteristiche per cui si è distinto dagli altri software sono:

* L'architettura, progettata per essere totalmente distribuita, in modo da rendere possibile il lavoro e il versionamento offline ed anche un versionamento integrato con i principali IDE;
* Le funzionalità di branching e merging potenti, rapide e comode, che permettono una efficente gestione di progetti, anche di grandi dimensioni;
* Le performance generalmente migliori e la possibilità di pubblicare i repository con i principali protocolli.

Ennova Research si appoggia nello specifico su GitLab, un servizio di hosting per progetti basato su Git.

2.2.3 Strumenti per *build automation*

Apache Maven

Maven è un software usato principalmente per la gestione di progetti Java e *build automation*. Per funzionalità è similare ad Apache Ant, ma basato su concetti differenti. Può essere usato anche in progetti scritti in C#, Ruby, Scala e altri linguaggi. Il progetto Maven è ospitato da Apache Software Foundation. Maven usa un costrutto conosciuto come Project Object Model (POM); un file XML che descrive le dipendenze fra il progetto e le varie versioni di librerie necessarie nonché le dipendenze fra di esse. In questo modo si separano le librerie dalla directory di progetto utilizzando questo file descrittivo per definirne le relazioni. Maven effettua automaticamente il download di librerie Java e plug-in Maven dai vari repository definiti scaricandoli in locale o in un repository centralizzato lato sviluppo. Questo permette di recuperare in modo uniforme i vari file JAR e di poter spostare il progetto indipendentemente da un ambiente all'altro avendo la sicurezza di utilizzare sempre le stesse versioni delle

librerie e la possibilità di impostare profili di esecuzione diversi. Per automatizzare la build e il lancio dell'SDK, Alfresco si basa appunto su Maven.

Capitolo 3

Descrizione dello stage

In questo capitolo si intende illustrare nel dettaglio le fasi e le considerazioni che hanno portato allo sviluppo del progetto e che sono alla base della proposta di stage

3.1 Introduzione al progetto

3.1.1 Considerazioni preliminari

Per prima cosa, è necessario premettere che l'azienda in cui si è svolto lo stage per i suoi prodotti software richiede l'impiego di procedure, regole e strumenti idonei a raggiungere il miglior risultato possibile. In particolare, durante lo stage sono stati richiesti i seguenti punti:

* Tracciabilità: è stato spesso richiesto di rendere conto dello stato di avanzamento dei processi, delle componenti e delle attività in lavorazione, al fine di verificare il rispotto della tabella di marcia,
* Documentazione: descrizione di quanto è stato fatto, delle strategie utilizzate e dei dettagli implementativi in modo puntuale e preciso, per far sì che quanto è stato fatto generi utilità all'azienda e sia riutilizzabile e replicabile in futuro, senza dover rifare tutte le attività, avendo già una strategia e una soluzione pronte e immediatamente utilizzabili.
* Misurazione: per ogni attività eseguita, deve essere misurabile il suo costo(in termini di tempo, nel caso del progetto qui esposto), e la sua qualità complessiva, in base a vari indicatori.

Quanto esposto è stato tenuto in grande considerazione durante lo sviluppo, al fine di realizzare un prodotto quanto più possibile conforme agli standard di qualità aziendali

3.1.2 Considerazioni iniziali

Alfresco, soprattutto nella versione Community, utilizzata dall'azienda perché Open Source è dotata di pochissime utilità di base, dal momento che il profitto dell'azienda che produce il software proviene dalla consulenza tecnica fornita dai loro esperti e dalla vendita di moduli ulteriori che vanno ad estendere la dotazione di base, moduli disponibili, soprattutto nel caso dei più interessanti, solo tramite la sottoscrizione

di un abbonamento, che viene valutato di caso in caso ma è comunque costoso. L'azienda quindi punta a fornire moduli adatti alla gestione di clienti e progetti integrati direttamente in Alfresco, e punta quindi gradualmente ad accentrare in Alfresco più funzionalità possibili, sostituendo gradualmente gli applicativi attualmente utilizzati ad esempio per la gestione dei progetti e la rendicontazione, per poi in un futuro poter proporre a clienti terzi il prodotto finito.Per questo motivo il progetto di stage si è concretizzato in quanto esposto, cercando quindi di introdurre alcune funzionalità.

3.1.3 Approccio alla piattaforma

Le prime due settimane di stage sono state dedicate quasi interamente allo studio della complessa architettura e della SDK della piattaforma. Per affrontare quanto appena illustrate è stato effettuato un attento studio della SDK di Alfresco, scegliendo il tipo di archetipo da utilizzare. La scelta è ricaduta sull'archetipo all-in-one in quanto permette lo sviluppo sia del lato front-end che di quello back-end in simultanea, senza dover distribuire ed installare separatamente un modulo per ogni parte sviluppata. In seguito, si sono analizzate le richieste poste dall'azienda in merito alle funzionalità desiderate al fine di individuare le strategie e i metodi migliori per realizzare quanto richiesto

3.1.4 Ambiente di test

Assieme allo sviluppo dei vari moduli, è proceduta a pari passo anche la creazione e il mantenimento di un ambiente di test dove poter testare i moduli in maniera sicura prima di portarli in produzione e applicare i moduli nel KMS aziendale. Ciò è stato espressamente richiesto dall'azienda e quindi i moduli prodotti, non appena la loro installazione era possibile, sono stati caricati e provati nel KMS di test che replicava quello dell'azienda, e successivamente, prima del rilascio finale, in una copia esatta del software del KMS di produzione.

3.2 Vincoli

per quanto riguarda l'implementazione, non sono stati imposti allo stagista vincoli stringenti per quanto riguarda i dettagli implementativi e accessibilità, anche se è stato esplicitato che il codice doveva essere facile da manutenere e che tutti i parametri e le stringhe utilizzate dovevano essere collocate in opportuni file di properties, in maniera da rendere facile configurazione e localizzazione.

Capitolo 4

Accenni al funzionamento di Alfresco

In questo capitolo sarà esposto come creare un modulo e accenni all'architettura di Alfresco

4.1 architettura di Alfresco

Al cuore di Alfresco c'è un repository che fornisce uno spazio per i contenuti, e un ampio spettro di servizi che possono essere utilizzati dalle applicazioni per manipolare il suo contenuto.il diagramma 4.1 a seguito mostra l'idea alla base di Alfresco, che può essere considerato consistente di tree principali componenti: la piattaforma (Paltform), la User Interface (UI), e la componente di ricerca (Search), basata su Apache Lucene. Queste componenti sono implementate come applicazioni web separate. Il principale componente è chiamato Platform ed è implementato nella web application alfresco.war. Essa fornisce il repository dove i file sono tenuti in memoria oltre a tutti i servizi di accesso e gestione collegati. Alfresco Share fornisce una interfaccia web (UI) per il repository ed è implementata nella web application share.war. Share nasce per facilitare agli utenti la gestione dei loro siti, docomenti, utenti e così via. la funzionalità di ricerca è implementata su Apache Solr 4 e fornisce una indicizzazione di tutti i contenuti, che rende possibile una potente funzionalità di ricerca. Oltre ai web client che accedono al repository via Share ci sono anche i dispositivi mobile che possono accedervi tramite le APIs REST fornite dalla piattaforma.

Se si approfondisce il componente Platform (contenuto nell'alfresco.war) vedremo che esso supporta i workflow nella forma dell'integrato Activiti Workflow Engine. La piattaforma di solito, come anche nell'azienda in cui è stato svolto lo stage, viene integrata con un Directory Server (LDAP), per essere in grado di sincronizzare i gruppi e gli utenti con Alfresco Community Edition. La maggior parte delle installazioni, compresa quella di Ennova, integrano Alfresco anche con un server SMTP così il componente Platform può mandare E-mail, come ad esempio, ma non solo, inviti ai siti.

Per maggiori informazioni si rimanda alla visione della documentazione di Alfresco.

Oltre a Share vi sono anche molti altri client che possono connettersi al repository, e questi includono anche molti client compatibili con CMIS, e via il protocollo SharePoint

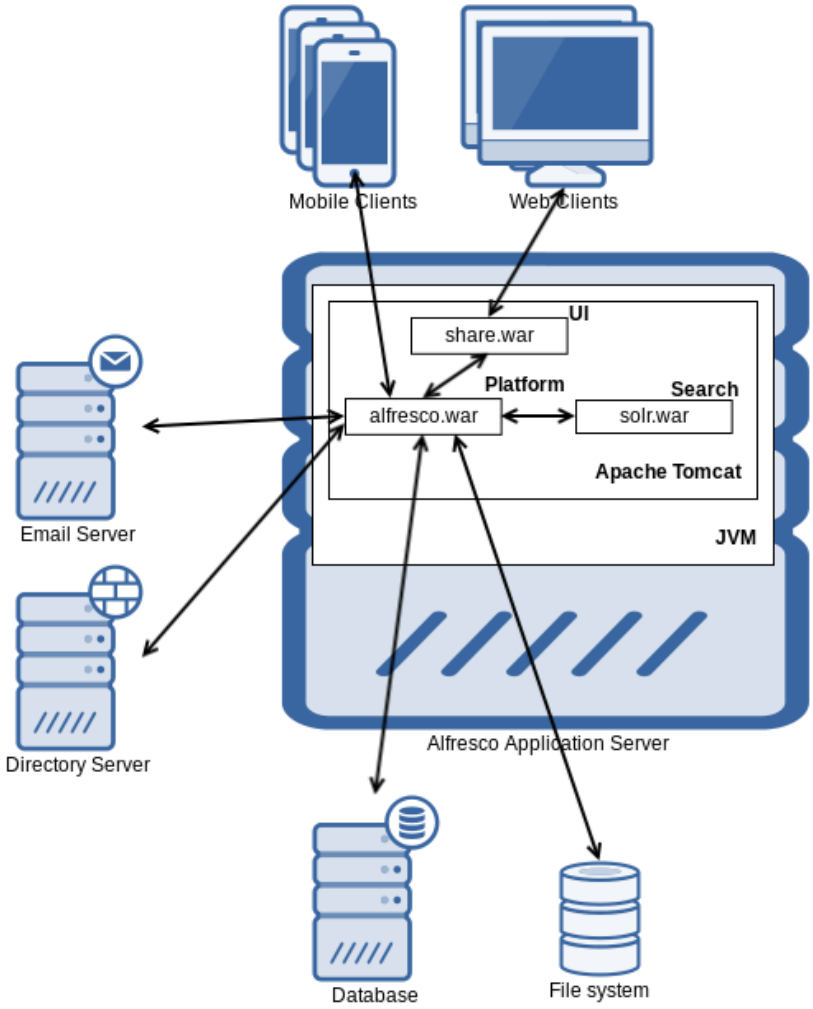


figura 4.1: architettura generale

e il client SharePoint. Esiste anche, a pagamento, la possibilità di sincronizzare i contenuti nel cloud.

Platform inoltre contiene numerose APIs, Servizi, e protocolli.

Il diagramma 4.2 illustra questa architettura estesa

Si noti come i metadati del contenuto sono conservati in un database relazionale come PostgreSQL, MySQL, Oracle, e così via. Il contenuto in se è conservato nel file system (o altri sistemi di storage come Amazon S3).

Alfresco fornisce un buon numero di punti di estensione per permettere ad uno sviluppatore di customizzarlo. Questi punti hanno molte forme, tra cui:

* Platform extension points, illustrati nell'immagine 4.4
* Share extension points, illustrati nell'immagine 4.3

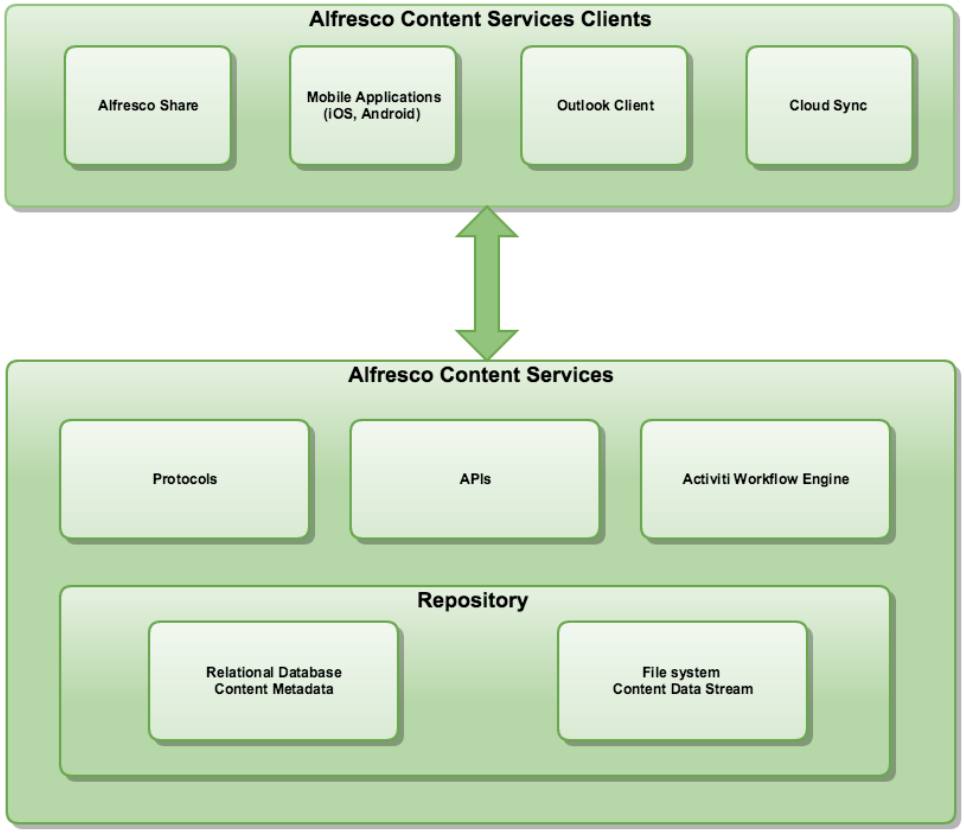


figura 4.2: architettura estesa

* Platform integration points, illustrati nell'immagine 4.5
* APIs
* Protocolli
* Servizi

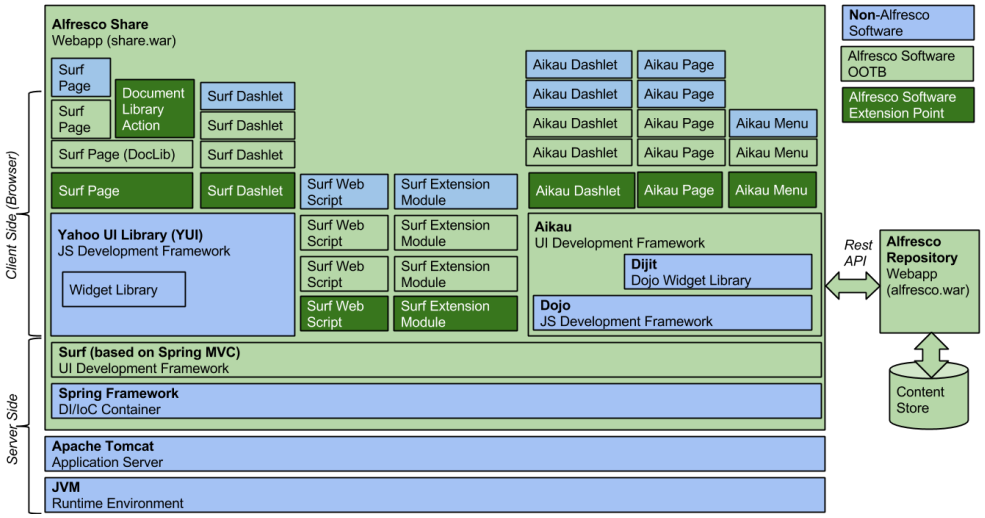


figura 4.3: Architettura di Share

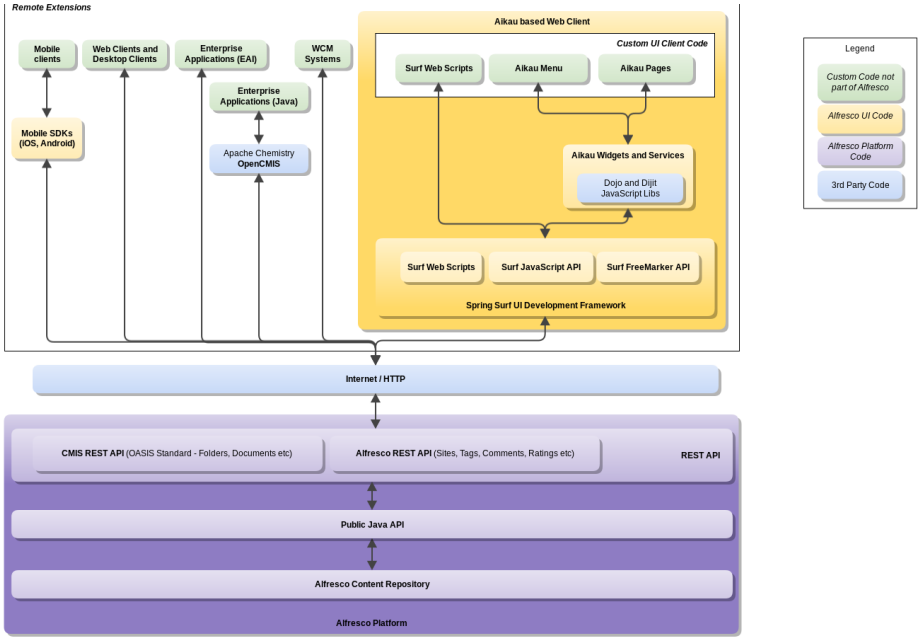


figura 4.4: Architettura del componente Platform

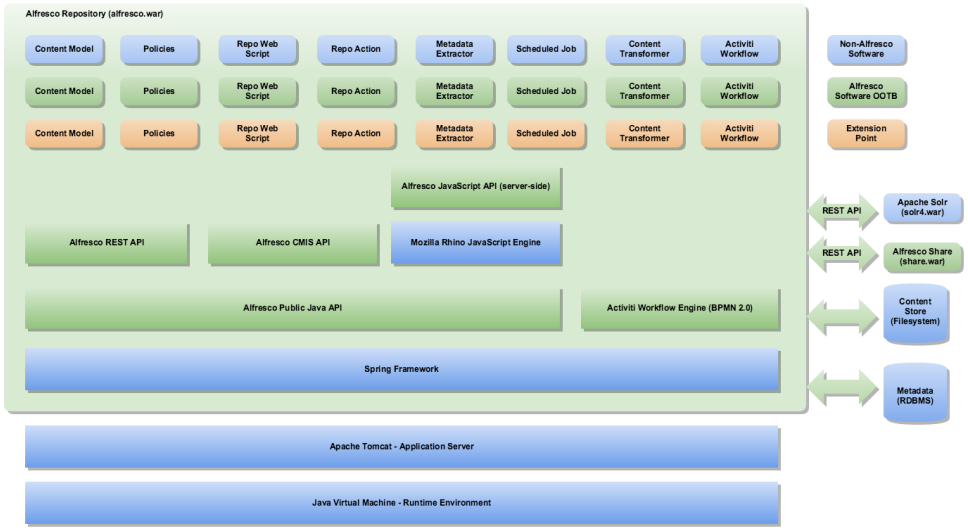


figura 4.5: Architettura del Repository

4.2 SDK

L'Alfresco SDK è un kit di sviluppo basato su Maven che fornisce un approccio semplice da utilizzare per lo sviluppo di applicazioni ed estensioni per Alfresco. Con questa SDK è possibile sviluppare, testare, eseguire, documentare e rilasciare il proprio modulo od estensione per Alfresco. La SDK fornisce tre archetipi di Maven che possono essere usati al fine di generare un progetto per Alfresco, e sono pensati per fornire un approccio standardizzaro allo sviluppo, rilascio e distribuzione dei prodotti relativi ad Alfresco. È quindi possibile generare i seguenti tipi di archetipi:

* Alfresco Repository AMP: questo archetipo si utilizza per creare moduli per il solo Alfresco Repository Web Application;
* Alfresco Share AMP: questo archetipo si utilizza per creare moduli per il solo Alfresco Share Web Application;
* Alfresco all-in-one (AIO): questo archetipo si utilizza quando vi è necessità di creare un progetto che richieda sia la componente Share che quella Repo, è la più potente e completa delle distribuzioni SDK, ma anche la più pesante.

tutti questi archetipi hanno delle caratteristiche comuni, quali:

* AMP packaging, cioè sono in grado di generare con la loro compilazione i file AMP, che sono i file che contengono i dati che poi verranno installati nei WAR corrispondenti.
* La gestione delle dipendenze in Maven;
* La generazione di uno scheletro di base sul quale poter creare il proprio progetto;
* Il supporto ai test di unità, test che vanno inseriti in un'apposita directory.
* Facilitazioni per l'integrazione di un progetto negli IDE, come IntelliJ IDEA e Eclipse.

4.2.1 Repository AMP archetype

L'Alfresco Repository AMP genera un scheletro di progetto per gestire le estensioni e le personalizzazioni dell'Alfresco Repository. Questo tipo di archetipo dovrebbe essere utilizzato:

* per lo sviluppo di un componente repo che non si vuole distribuire nello stesso modulo di un componente all-in-one, per facilitare il lavoro in progetti di grandi dimensioni, che così possono suddividere il progetto in componenti più piccole.
* per poter creare un componente che sia riusabile e si possa reincludere a sua volta in componenti all-in-one.
* per creare un modulo che vada ad interagire con la sola parte repo.

Questo archetipo presenta caratteristiche ulteriori:

* Ha il supporto di un database H2 per simulare il database di Alfresco.
* supporta test di integrazione e di unità tramite Junit e RAD (rapid application development)

4.2.2 Share AMP archetype

Alfresco Share AMP Archetype genera un progetto esempio per gestire le estensioni e le personalizzazioni di Alfresco Share. Questo tipo di archetipo dovrebbe essere utilizzato:

* Realizzare un modulo Share per poi includerlo all'interno di un progetto all-in-one.
* Costruire un Add-On, componente, modulo da distribuire separatamente.
* per poter spezzare il lavoro, per esempio per delegare a team diversi il lato front-end e il lato back-end
* per creare temi e personalizzazioni che coinvolgano il solo lato UI.

Anche questo modulo presenta alcune caratteristiche particolari:

* Sono presenti alcuni componenti di esempio per facilitare la creazione di nuove funzionalità
* può interfacciarsi con qualsiasi componente repo di Alfresco, previa configurazione; questo permette di poter provare in sicurezza tutte le modifiche alla UI apportate direttamente in un ambiente molto vicino a quello di produzione.

4.2.3 All-in-One archetype

All-in-One archetype è un progetto multi-modulo per personalizzare ed estendere la Share e la Repository di Alfresco.Questo tipo di archetipo dovrebbe essere utilizzato per:

* Realizzare un progetto che permetta la personalizzazione in contemporanea e in un singolo modulo della Share e della Repository;
* Accedere alla suite completa dei test di regressione per la Alfresco Share UI;
* Accedere ai test funzionali basati sulla libreria Alfresco Share Page Object (PO).

Le principali caratteristiche di questo archetipo sono:

* Inclusione semplice degli AMPs extra;
* Il supporto ad estensioni Out-of-the-box come la gestione dei records, il protocollo SharePoint, la gestione dei Media;
* Possibilità di personalizzare i test funzionali;
* La replicazione di un'intera distribuzione di Alfresco, che funziona autonomamente e non ha bisogno di dipendenze aggiuntive.

Ovviamente questo archetipo è il più pesante, e il suo completo deploy richiede una quantità di tempo non indifferente, per cui andrebbe usato solo se vi è reale necessità di utilizzare tutte le caratteristiche uniche che esso fornisce

4.3 Gestione dei moduli

4.3.1 creazione

Bisogna innanzitutto rispondere ai prerequisiti descritti in questa pagina. Fatto questo il progetto può essere importato in un IDE, ad esempio per Eclipse in questo sito Le cartelle su cui andremo a sviluppare saranno la repo-amp e la share-amp. La share-amp verrà utilizzata per sviluppare l'interfaccia grafica del modulo mentre la parte di repo-amp per lo sviluppo del "back-end", le altre folder create servono per replicare l'ambiente di Alfresco in fase di esecuzione del modulo, e quindi non ha senso lavorare su di esse. è necessario installare il driver jdbc se si intende lavorare nell'SDK interagendo con database terzi, e dato che non è incluso nell'SDK anche se presente nell'ambiente finale, va incluso seguendo quanto detto in questo luogo . Una volta fatto questo, bisogna aprire nell'IDE o anche a mano all'indirizzo, partendo dalla cartella di installazione (sono riportati gli indirizzi utilizzando i nomi dell'esempio sopra citato: `/ alfresco-extension / acme-cms-poc / acme-cms-poc-share-amp / src / main / amp / config / alfresco / web-extension / site-webscripts / com / example / pages` (questo porta a dove è situata la pagina dimostrativa). È sufficiente fermarsi due o tre livelli sopra se l'intenzione è quella di mettere la gerarchia per una pagina custom (come fatto in questo progetto). Una pagina fatta in alfresco share è composta da tre file, per semplicità prendiamo in considerazione la pagina di esempio già fornita: un file .xml, che descrive i dati essenziali della pagina:

```
<webscript>
    <shortname>Simple Page</shortname>
    <description>Simple page definition</description>
    <family>Share</family>
    <authentication>user</authentication>
    <url>/simple-page</url>
</webscript>
```

molto importanti sono il tag authentication, che specifica il livello di autenticazione necessario per accedere alla pagina, e il tag url, dal momento che definisce l'indirizzo della pagina, che sarà `<ip>:port/share/page/hdp/ws<url>` (hdp genera la pagina con header e footer, dp la pagina grezza). Vi è poi il file .html che si comporta proprio come il body di una pagina html. Vi è infine un file .js, che serve per applicare delle modifiche via json al model. Quanto detto è sufficiente per creare una pagina che non necessiti di interazioni con java.

Per quanto riguarda una pagina che invece fa uso delle api di Alfresco il procedimento è più complicato: la pagina che verrà mostrata all'utente è collocata nella cartella `alfresco-extension /acme-cms-poc /acme-cms-poc-repo / amp / src / main / amp / config / alfresco / extension /templates / webscripts`. La pagina generata è situata all'indirizzo `<ip>:<porta>/alfresco/service<url>`. E' inoltre necessario creare un bean che fa da collegamento tra il codice java, situato in `alfresco-extension / acme-cms-poc / acme-cms-poc-repo-amp / src / main / java / <package>`, e la pagina che verrà generata. Per farlo seguire quanto descritto in questo sito.

4.3.2 Manutenzione

Nel momento in cui si vorrà ad andare a modificare un modulo per evenienze riguardanti aspetti grafici o logici indiretti, bisognerà aprire il modulo con un IDE a piacere, e applicare le modifiche desiderate sui file. Il risultato delle modifiche effettuate è visionabile lanciando il comando `./run.sh` una volta posizionti all'interno della folder del modulo su linux, `run.bat` su windows, in alternativa al comando `mvn install -Prun` o `mvn clean install -Prun`, se è necessario anche svuotare la cache; avviato il server basterà aprire il browser e digitare l'url : `<ip>:8080/share/page`. Per quanto riguarda l'importazione dei moduli nell'ambiente di Alfresco vero e proprio, si dovrà lanciare il comando `mvn clean install` all'interno sempre del folder del modulo e all'interno della cartella target delle rispettive cartelle repo-amp e share-amp si creeranno un repo-amp file e uno share-amp file. Questi dovranno essere inseriti rispettivamente nelle cartelle amps e amps_share nell'ambiente dell'Alfresco di produzione. Per installare il modulo è necessario lanciare il file `alfresco-community/bin/apply_amps.bat`, oppure lanciandolo tramite il flag `-force` se si ha già un modulo con lo stesso nome che si vuole sovrascrivere. Attenzione L' installazione dei moduli comporta un deploy che sovrascriverà il contenuto delle cartelle `tomcat/webapps/share` e `tomcat/webapps/alfresco`. Di conseguenza nel momento in cui si è andati ad effettuare delle estensioni direttamente all'interno di queste cartelle il contenuto andrà perso. Quindi prima di effettuare questa operazione è necessario fare il backup delle cartelle indicate precedentemente.

Per comodità d'ora in poi il nome del progetto sarà indicato come AIOProject.

Capitolo 5

Il modulo clienti

In questo capitolo verrà esposta l'implementazione e le fasi che hanno portato la creazione del modulo clienti

5.1 scopo del modulo

il modulo clienti si pone come scopo quello di creare un semplice sistema di CRUD in un database. In questo caso è stato implementato in un database Postgres, che è quello che viene incluso di default con la distribuzione di Alfresco Community, ma è stata esplicita richiesta dell'azienda che fosse possibile una configurazione su ambienti simili ma che non fossero esclusivamente Postgres.

5.2 implementazione delle funzionalità

Preparazione del DB Per preparare il db bisogna innanzitutto creare un nuovo database.

Nel caso del progetto da me sviluppato si è usato PostgreSQL

Basta quindi spostarsi nella cartella di postgresql tramite dos e digitare il comando createdb -h localhost -p <porta> -U postgres <nome del db>

Bisogna quindi creare una tabella, che verrà chiamata per comodità da adesso clienti.

La tabella ha la seguente struttura

```
CREATE TABLE clienti
(
  identificativo text NOT NULL,
  name text NOT NULL,
  piva text NOT NULL,
  descr text,
  datainizio text NOT NULL,
  datafine text,
  id bigserial NOT NULL,
  CONSTRAINT clienti\_pkey PRIMARY KEY (id)
  CONSTRAINT clienti\_identificativo\_name\_piva\_descr\_
      datainizio\_datafine\_key UNIQUE (identificativo, name, piva
      , descr, datainizio, datafine)
)
```

Nel creare la tabella è importante prestare attenzione al fatto che l'owner della tabella debba essere il medesimo che è stato settato nelle properties. L'ultimo vincolo è necessario per non avere record che differiscano solo per id. In particolare si è scelto di tenere le date in formato testuale per rendere il db compatibile a diverse implementazioni sulla rappresentazione della data, e si è scelto di usare come chiave primaria un id chiamato bigserial, in quanto più resistente ad una eventuale estensione dei campi della tabella e più comodo da utilizzare, dal momento che, per le caratteristiche del bigserial, si è certi che l'ultima versione di un record è quella con l'id più alto, quindi per prendere l'ultima versione di un cliente di un derterminato identificativo basta eseguire la query `select * from clienti where id = (select max(id) from clienti where identificativo='param1')`. Si è cercato di non utilizzare viste o altri strumenti particolari per evitare di essere troppo dipendenti dalla piattaforma.

5.2.1 Creazione della parte share

È quindi stato necessario creare le pagine share e le pagine repo, oltre ai bean di collegamento per poter rendere accessibile il webscript. Si è adottato il metodo POST in tutti quante le invocazioni, in quanto è più sicuro e permette lo scambio di una maggiore quantità di dati.

Si è aggiunta innanzitutto una nuova voce di menu aggiungendo il file `/ AIOProject-share-amp / src / main / amp / config / alfresco / web-extension / site-data / extensions / add-create-menuitem-doclib-extension-modules.xml`

```
<extension>
    <modules>
        <!-- This module is dependent on the custom content model
             setup in the repo-amp module -->
        <module>
            <id>Add a new menu item to Create... menu in DocLib</
                id>
            <version>1.0</version>
            <auto-deploy>true</auto-deploy>
            <configurations>
                <config evaluator="string-compare" condition="
                    DocumentLibrary">
                    <create-content>
                        <content id="text-label-clienti" label="
                            Crea nuovo cliente" icon="text" type="
                            pagelink">
                            <param name="page">hdp/ws/clienti</
                                param>
                        </content>
                    </create-content>
                </config>
                            </configurations>
        </module>
    </modules>
</extension>
```

Per ulteriore documentazione si faccia riferimento a al relativo capitolo nella documentazione di alfresco

Le pagine includono inoltre nell'HTML la richiesta di JQuery, mediante le linee

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/
    jquery.min.js"></script>
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/
    bootstrap/3.3.7/css/bootstrap.min.css" integrity="sha384-
    BVYiiSIFeK1dGmJRAkycuHAHRg32OmUcww7on3RYdg4Va+PmSTsz/K68vbdEjh
    4u" crossorigin="anonymous">
```

pagina Clienti

È la pagina di inserimento di un nuovo cliente ed è composta da un semplice form. È composta dai seguenti file:

* `/ AIOProject-share-amp / src / main / amp / config / alfresco / web-extension / site-webscripts / com / clienti / pages / clienti.get.desc.xml`, implementato come descritto nel quarto capitolo
* `/ AIOProject-share-amp / src / main / amp / config / alfresco / web-extension / site-webscripts / com / clienti / pages / clienti.get.html.ftl` che contiene il codice html necessario a generare il form.
* `/ AIOProject-share-amp / src / main / amp / config / alfresco / web-extension / site-webscripts / com / clienti/ pages / clienti.get.js` che contiene le istruzioni necessarie a generare l'header. È necessario includere nell'html la linea <@processJsonModel group="share"/>.
* `/AIOProject-share-amp/src/main/amp/web/js/clienti/clienti.js`, incluso con la linea `<script src="$url.context / js / clienti / clienti.js"></script>`, che effettua i controlli javascript della pagina. In particolare alla pressione del pulsante di conferma viene fatto il controllo sui dati inseriti, cioè sulla correttezza della partita iva, sul fatto che i campi obbligatori siano definiti, sul fatto che la data di fine sia successiva alla data di inizio. Inoltre si occupa di inoltrare la richiesta AJAX al webscript `query_processor.java` che sarà trattato successivamente. Se il webscript riporta uno stato anomalo viene segnalato e così viene anche gestito il caso di inserimento di un identificativo già esistente.

Pagina vis_clienti

La pagina vis_clienti è la pagina di gestione dei clienti, ed è suddivisa nei file:

* `/ AIOProject-share-amp / src / main / amp / config / alfresco / web-extension / site-webscripts / com / visualizza_clienti / pages / vis_clienti.get.desc.xml` che funziona come descritto nel quarto capitolo.
* `/ AIOProject-share-amp / src / main / amp / config / alfresco / web-extension / site-webscripts / com / visualizza_clienti / pages / vis_clienti.get.html.ftl` che fornisce lo scheletro della pagina, che verrà poi popolata tramite richieste AJAX delle parti dinamiche.

* `/ AIOProject-share-amp / src / main / amp / config / alfresco / web-extension / site-webscripts / com /visualizza_clienti / pages / vis_clienti.get.js`, che funziona come descritto prima.
* `/ AIOProject-share-amp / src / main / amp / web / css / vis_clienti / vis_clienti.css`, incluso in maniera analoga al javascript come descritto prima e che definisce il CSS della pagina.
* `/ AIOProject-share-amp / src / main / amp / web / js / vis_clienti / vis_clienti.js`, incluso come descritto prima

In particolare la pagina nella sua parte principale offre la lista degli identificativi dei clienti attivi e non, che è anche il nome con cui sono stati salvati nella cartellatura di Alfresco. Si è scelto questa rappresentazione in quanto è logicamente vicina alla struttura di Alfresco e sarà più facile in futuro implementare funzionalità quali la visualizzazione dei progetti del cliente. I pulsanti dei clienti attivi e inattivi sono generati rispettivamente da `clients.java` e `inactive.java` chiamati all'avvio della pagina tramite `window.onload` che si occupano di definire anche i giusti parametri che verranno poi utilizzati per popolare il form che consente l'aggiornamento dei dati di un particolare cliente. Il form è già presente nella pagina ma viene fatto mostrare da una opportuna funzione javascript che si occupa anche di popolare la tabella sottostante tramite una chiamata AJAX al webscript `TablePageGenerator.java`, che si occupa di fornire la tabella e di iniettare il pulsante cancella e il pulsante per definire la data di fine del rapporto con il corretto id. I pulsanti sono collegati a loro volta a `delete.java` e `update.java`, chiamati tramite opportune chiamate AJAX.

I messaggi di errore e di conferma sono gestiti dove possibile in JavaScript e dove non possibile, poiché la risposta dipende dal database o dal repository, tramite la risposta della chiamata AJAX

Per riassumere la pagina fa uso delle seguenti funzioni JavaScript:

* `set_table(identificativo)` che si occupa di generare la tabella di un determinato cliente tramite richiesta AJAX
* `detail(id,nome,pi,descr,datainizio,datafine)`che si occupa di settare E mostrare il dettaglio di un cliente
* `controllaPIVA(pi)` che esegue semplicemente il controllo della partita IVA
* `sendRequest()` Che invia la richiesta AJAX di aggiornamento di un determinato cliente
* `set_data_fine(ID)` che setta la data di fine di un determinato record di un cliente tramite richiesta AJAX
* `cancella(key)` che fa la richiesta tramite AJAX di cancellazione di un record con un determinato ID
* `up()` che popola la pagina con i pulsanti dei clienti, tramite richiesta AJAX.
* `vai(name)` che reindirizza alla cartella con dato nome.

Sono presenti altre funzioni che svolgono compiti banali quale la codifica e decodifica di entità, per garantire il funzionamento della pagina nel caso i record contengano caratteri speciali

5.2.2 Creazione della parte repo

Creazione delle variabili del modulo

Per rendere possibile la modifica di alcune configurazioni anche dopo che il modulo è stato installato in alfresco, si è provveduto a creare un file in `/AIOProject-repo-amp / src / main / amp / config / alfresco / module / AIOProject-repo-amp / config / configModule.properties`, Accessibile da java aggiungendo al codice le linee:

```
private static Properties properties=new Properties();

public static final String getValue(String value) {
                return properties.getProperty(value);
        }

properties.load(getClass().getResourceAsStream("/alfresco/module/
    AIOProject-repo-amp/config/configModule.properties"));
```

Questo permette quindi di recuperare le variabili, che si è cercato di documentare direttamente sul codice tramite nomi significativi e una breve descrizione:

```
#query configuration
queryselection=select id,name, piva, datainizio, datafine, descr
    from clienti where identificativo='param1' order by id
querylastversion=select * from clienti where id = (select max(id)
     from clienti where identificativo='param1')
queryselectactive=select distinct identificativo from clienti
    where NOT (datafine IS NOT NULL)
queryselectnotactive=select distinct identificativo from clienti
    where datafine IS NOT NULL and identificativo not in (select
    distinct identificativo from clienti where NOT (datafine IS
    NOT NULL))
#do not use values in the form paramX! They may be sobstituted!
queryselectionwithclause=select param1 from param2 where param3
queryinsertion=insert into param1 values ('param2','param3','
    param4', 'param5', 'param6', 'param7')
queryupdate=UPDATE param1 SET param2 = 'param3' WHERE id = 'param
    4';
querydelete=DELETE FROM param1 WHERE id = 'param2';
#set param1 in the previous query
querytable=Clienti
#server configuration
serverip=127.0.0.1
serverid=alfresco
serverpassword=admin
serverport=5432
servername=<nome del database creato>
servertype=postgresql
serverconnector=jdbc
#path in lucene, in the form of PATH:/"/{directory}, the "/""
    character is needed
#because without it it will be escaped by java, java alone puts
    the final " so
#you must not put it when you write the query
```

```
installationpath=PATH:/"/app:company\_home/st:sites/cm:er/cm:
    documentLibrary/cm:\_x0030\_2\_x0020\_-\_x0020\_Clients
#down here you can put the names for the folder that are in the
    client subdirectory
folder.one=01 - Projects
folder.two=02 - General Contracts
folder.three=tre
folder.four=quattro
folder.five=cinque
```

Come si vede dove necessario si è cercato di parametrizzare le funzioni.

Creazione dei webscript

Come già accennato, le pagine in share per il loro completo funzionamento necessitano di webscript di supporto, per interrogare il database e generare le tabelle contenenti i dati. Tutti quanti estendono la classe AbstractWebScript, sono contattabili tramite POST e fanno uso del JDBC per contattare il server remoto dove è presente la tabella. Il file .properties descritto prima permette di cambiare secondo le necessità la configurazione della connessione, definita nella riga

```
connection=DriverManager.getConnection((connector+":"+type+"://"+
    ip+":"+port+"/"+server),id,password);
```

Che sono i parametri definiti nel paragrafo query configuration del .properties

I WebScript sono poi registrati nel file `/ AIOProject-repo-amp / src / main / amp / config / alfresco / module / AIOProject-repo-amp / context / webscript-context.xml` secondo le modalità già descritte.

Per comodità nella chiamata l'indirizzo è stato settato con lo stesso nome della funzione java ad esso corrispondente. Inoltre alcuni accettano parametri in JSON definiti in coppie `key:value` in un JSonObject

Passiamo ora in rassegna i vari webscript implementati, in ordine alfabetico.

clients.java Questo webscript, situato in `/ AIOProject-repo-amp / src / main / java / clients.java` e dalla pagina definita in `/ AIOProject-repo-amp / src / main / amp / config / alfresco / extension / templates / webscripts / clients.post.desc.xml` e in `/ AIOProject-repo-amp / src / main / amp / config / alfresco / extension / templates / webscripts / clients.post.html.ftl`, si occupa di generare i pulsanti necessari a visualizzare poi la pagina di dettaglio per un utente attivo. Si occupa quindi di interrogare il database e di definire di conseguenza i parametri dei pulsanti e il loro nome, che chiamano uno la funzione `detail(id,nome,pi,descr,datainizio,datafine)`,e l'altro la funzione `vai(name)`. Si appoggia sulla query definita in queryselectactive per selezionare gli utenti attivi, percorrendo poi il result set per di volta in volta assegnare i parametri. Viene chiamato senza parametri. La classe si compone delle seguenti funzioni:

* `getValue(String value)`, che recupera una property con una data value.
* `inizialize_values ()`, che inizializza le variabili utilizzate, recuperandole dalle properties.
* `public void execute(WebScriptRequest req, WebScriptResponse res)` metodo obbligatorio che si può intendere come il metodo che viene invocato alla chiamata Ajax del webscript e che contiene le istruzioni per gestirla.

* `execute(Connection con, String query, WebScriptResponse res)`, metodo che esegue la query.

* `writedata(ResultSet rs, String result, Connection con)`, metodo che si occupa di ritornare il codice html necessario a generare i pulsanti dei clienti attivi

delete.java Questo webscript, situato in `/ AIOProject-repo-amp / src / main / java / delete.java` e dalla pagina definita in `/ AIOProject-repo-amp / src / main / amp / config / alfresco / extension / templates / webscripts / delete.post.desc.xml` e in `/ AIOProject-repo-amp / src / main / amp / config / alfresco / extension / templates / webscripts /delete.post.html.ftl` si occupa semplicemente di cancellare un elemento che abbia un determinato id, fornito come parametro della chiamata. Si appoggia sulla query definita alla voce querydelete. La classe si compone delle seguenti funzioni:

* `getValue(String value)`, che recupera una property con una data value

* `inizialize_values ()`, che inizializza le variabili utilizzate, recuperandole dalle properties.

* `public void execute(WebScriptRequest req, WebScriptResponse res)` metodo obbligatorio che si può intendere come il metodo che viene invocato alla chiamata Ajax del webscript e che contiene le istruzioni per gestirla.

* `execute(Connection con, String query)`, metodo che esegue la query di delete di un dato record del database.

inactive.java Questo webscript, situato in `/ AIOProject-repo-amp/src/main/java/inactive.java` e dalla pagina definita in `/ AIOProject-repo-amp / src / main / amp / config / alfresco / extension / templates / webscripts / inactive.post.desc.xml` e in `/ AIOProject-repo-amp / src / main / amp / config / alfresco / extension / templates / webscripts / inactive.post.html.ftl` si occupa di generare i pulsanti necessari a visualizzare poi la pagina di dettaglio per un utente non più attivo. Si occupa quindi di interrogare il database e di definire di conseguenza i parametri del pulsante e il suo nome, che chiama la funzione `detail(id,nome,pi,descr,datainizio,datafine)`. Si appoggia sulla query definita in queryselectnotactive per selezionare gli utenti non attivi. Percorrendo poi il result set per di volta in volta assegnare i parametri. Viene chiamato senza parametri. La classe si compone delle seguenti funzioni:

* `getValue(String value)`, che recupera una property con una data value

* `inizialize_values ()`, che inizializza le variabili utilizzate, recuperandole dalle properties.

* `public void execute(WebScriptRequest req, WebScriptResponse res)` metodo obbligatorio che si può intendere come il metodo che viene invocato alla chiamata Ajax del webscript e che contiene le istruzioni per gestirla.

* `execute(Connection con, String query, WebScriptResponse res)`, metodo che esegue la query.

* `writedata(ResultSet rs, String result, Connection con)`, metodo che si occupa di ritornare il codice html necessario a generare i pulsanti dei clienti non attivi.

Si è deciso di renderla autonoma rispetto a clients per consentire una migliore estensione ad ulteriori eventuali funzionalità.

query_processor.java Questo webscript, situato in `/ AIOProject-repo-amp / src / main / java / query_processor.java` e dalla pagina definita in `/ AIOProject-repo-amp / src / main / amp / config / alfresco / extension / templates / webscripts / query_processor.post.desc.xml` e in `/ AIOProject-repo-amp / src / main / amp / config / alfresco / extension / templates / webscripts / query_processor.post.html.ftl` . Si occupa di inserire un oggetto nel database, utilizzando la query definita in queryinsertion e deve essere chiamato con i seguenti parametri:

* ID:l'identificativo del cliente,
* Nome:il nome esteso del cliente,
* Piva:la partita iva del cliente,
* Descr:una descrizione generica,
* Datainizio:la data di inizio del rapporto,
* Datafine:la data di fine del rapporto(se non presente, verrà settata a null),
* Mode:1=inserimento con la creazione delle cartelle, 2=inserimento senza creazione di cartelle ,
* Box1:flag per la creazione della cartella di nome definito in folder.one,
* box2:flag per la creazione della cartella di nome definito in folder.two,
* box3:flag per la creazione della cartella di nome definito in folder.three,
* box4:flag per la creazione della cartella di nome definito in folder.four,
* box5:flag per la creazione della cartella di nome definito in folder.five,

Lo script si appoggia, oltre al JDBC per l'inserimento nel database, sulla API di Alfresco FileFolderService per creare la gerarchia di cartelle corrispondente, di permissionservice per settare i permessi di accesso alle cartelle (per il momento codificati nel Java e non configurabili dopo l'installazione se non mediante l'editor dei permessi di Alfresco) E di nodeservice per settare la descrizione della cartella principale. Per includere il serviceregisty, necessario per accedere alla API di Alfresco, è necessario, oltre a specificare nel bean, il cui funzionamento è stato già descritto, tramite le righe

```
    <property name="serviceRegistry">
      <ref bean="ServiceRegistry" />
  </property>
```

E poi nel codice java con l'aggiunta di

```
import org.alfresco.service.ServiceRegistry;

private ServiceRegistry serviceRegistry;

public void setServiceRegistry(ServiceRegistry serviceRegistry) {
                this.serviceRegistry = serviceRegistry;
        }
```

Ritorna una stringa di risultato dove è specificato l'esito dell'operazione. La classe si compone delle seguenti funzioni:

* `setServiceRegistry(ServiceRegistry serviceRegistry)`, che inizializza il parametro per il service registry di Alfresco. È richiesto obbligatoriamente da Alfresco se si vogliono utilizzare le sue funzionalità
* `getValue(String value)`, che recupera una property con una data value
* `inizialize_values ()`, che inizializza le variabili utilizzate, recuperandole dalle properties.
* `public void execute(WebScriptRequest req, WebScriptResponse res)` metodo obbligatorio che si può intendere come il metodo che viene invocato alla chiamata Ajax del webscript e che contiene le istruzioni per gestirla.
* `execute(Connection con, String query)`, metodo che esegue la query.
* `create_folder_tree(String ID, String descr,Boolean box1, Boolean box2, Boolean box3, Boolean box4, Boolean box5)`, metodo che si occupa di creare la cartellatura di un nuovo cliente, di settare i permessi alle cartelle e di settare la descrizione della cartella padre.

TablePageGenerator.java Questo webscript, situato in `/ AIOProject-repo-amp / src / main / java / TablePageGenerator.java` e dalla pagina definita in `/AIOProject-repo-amp / src / main / amp / config / alfresco / extension / templates / webscripts / TablePageGenerator.post.desc.xml` e in `/ AIOProject-repo-amp / src / main / amp / config / alfresco / extension / templates / webscripts / TablePageGenerator.post.html.ftl` si occupa di generare la tabella dello storico di un determinato codice cliente, fornito come parametro della chiamata AJAX. Si appoggia sulla query definita alla voce queryselection. Il webscript si occupa di generare il codice della tabella e di iniettare i giusti parametri nel pulsante per cancellare, e, nel caso la data di fine non sia stata settata, anche di generare il pulsante per la modifica della data di fine rapporto. La classe si compone delle seguenti funzioni:

* `getValue(String value)`, che recupera una property con una data value
* `inizialize_values ()`, che inizializza le variabili utilizzate, recuperandole dalle properties.
* `public void execute(WebScriptRequest req, WebScriptResponse res)` metodo obbligatorio che si può intendere come il metodo che viene invocato alla chiamata Ajax del webscript e che contiene le istruzioni per gestirla.

* `execute(Connection con, String query, WebScriptResponse res)`, metodo che esegue la query.
* `dumpData(ResultSet rs, String result)`, metodo che si occupa di ritornare il codice html necessario a generare la tabella dello storico di un cliente.

update.java Questo webscript, situato in `/ AIOProject-repo-amp / src / main / java / update.java` e dalla pagina definita in `/ AIOProject-repo-amp / src / main / amp / config / alfresco / extension / templates / webscripts / update.post.desc.xml` e in `/ AIOProject-repo-amp / src / main /amp / config / alfresco / extension / templates / webscripts / update.post.html.ftl` si occupa semplicemente di cancellare un elemento che abbia un determinato id, un campo da aggiornare e un valore con cui aggiornarlo, forniti come parametro della chiamata. Si appoggia sulla query definita alla voce queryupdate. La classe si compone delle seguenti funzioni:

* `getValue(String value)`, che recupera una property con una data value
* `inizialize_values ()`, che inizializza le variabili utilizzate, recuperandole dalle properties.
* `public void execute(WebScriptRequest req, WebScriptResponse res)` metodo obbligatorio che si può intendere come il metodo che viene invocato alla chiamata Ajax del webscript e che contiene le istruzioni per gestirla.
* `executeupdate(Connection con, String query)`, metodo che esegue la query di update della data di fine di un dato record del database.

5.3 lato estetico

Per il lato estetico si è reso necessario contattare il team di grafici che lavora presso l'azienda per ottenere un aspetto gradevole e che si allineasse con quanto già presente in Alfresco. Sono stati quindi forniti dei mockup delle varie componenti ed è stato chiesto allo stagista di riprodurre tale aspetto in Alfresco.

5.3.1 risultati raggiunti

TODO:qui le immagini

Glossario

API in informatica con il termine *Application Programming Interface API* (ing. interfaccia di programmazione di un'applicazione) si indica ogni insieme di procedure disponibili al programmatore, di solito raggruppate a formare un set di strumenti specifici per l'espletamento di un determinato compito all'interno di un certo programma. La finalità è ottenere un'astrazione, di solito tra l'hardware e il programmatore o tra software a basso e quello ad alto livello semplificando così il lavoro di programmazione. 1

B2B Business-to-business, spesso indicato con l'acronimo *B2B*, in italiano commercio interaziendale, è una locuzione utilizzata per descrivere le transazioni commerciali elettroniche tra imprese, distinguendole da quelle che intercorrono tra le imprese e altri gruppi, come quelle oppure quelle tra una impresa e il governo. 1

B2C Con Business to Consumer, spesso abbreviato in *B2C*, si indicano le relazioni che un'impresa commerciale detiene con i suoi clienti per le attività di vendita e/o di assistenza. 1

ICT Information and communication tecnology:
Le tecnologie dell'informazione e della comunicazione (in inglese Information and Communications Technology, in acronimo ICT), sono l'insieme dei metodi e delle tecnologie che realizzano i sistemi di trasmissione, ricezione ed elaborazione di informazioni (tecnologie digitali comprese). 1

KMS Knowledge management system:
I Knowledge management system sono sistemi software che supportano le fasi del ciclo dell'informazione e la comunicazione all'interno di una comunità di pratica (ad esempio un'azienda) o di apprendimento (ad esempio una classe "virtuale") anche disperse nello spazio. Dovrebbero assistere le persone ad esplicitare la conoscenza tacita, a reperirla, a condividerla, supportando in particolare le seguenti funzioni:

* Cattura delle competenze collettive
* Controllo per realizzare obiettivi comuni
* Integrazione delle conoscenze frammentate

. iii, 1, 10

Acronimi

API Application Program Interface. 31

B2B Business-to-business. 31

B2C Business-to-Business. 31

ICT Information and Communication Tecnology. 31

KMS Knowledge Managment System. 31

Bibliografia

Riferimenti bibliografici

James P. Womack, Daniel T. Jones. *Lean Thinking, Second Editon*. Simon & Schuster, Inc., 2010.

Siti Web consultati

Manifesto Agile. URL: http://agilemanifesto.org/iso/it/.