UNIVERSITÀ DEGLI STUDI DI PADOVA
FACOLTÀ DI SCIENZE MM.FF.NN.

Dipartimento di Matematica Pura ed Applicata
Corso di laurea in Informatica
Classe 26

Tesi di Laurea

EniWiki – SubVersion – Alfresco
Gestione delle informazioni a livello Enterprise

Relatore
Dott. Gilberto Filè

Laureando
Luca Rubin

Anno Accademico 2008/2009

Indice Generale

1 Introduzione

L'informazione è tanto utile quanto complessa da gestire.

La tecnologia avanza senza sosta e non sempre le aziende riescono a tenere il passo con i moderni sistemi di gestione dei dati. Al giorno d'oggi sono tanti i modi con cui l'informazione può essere gestita in ambito Enterprise; uno di questi, forse il più diffuso (ed inefficiente), è quello che fa uso di file organizzati e gestiti in un file system.

Questo metodo se pur semplice ed efficace, può presentare delle lacune molto importanti, soprattutto se la mole di informazioni gestita è molto grande.

La gestione delle informazioni è un concetto molto ampio che non si limita alla sola memorizzazione dei dati, ma riguarda molte altre attività come la diffusione (possibilmente controllata) dei dati, le attività di ricerca, la classificazione, ecc.

In questo documento verrà descritto lo stage da me eseguito presso la ditta EniData nel periodo che va dal 13/10/2008 al 03/12/2008.

1.1 Lo stage

Le trecento ore di stage svolte presso EniData mi hanno permesso (in collaborazione con l'azienda) di apportare degli ammodernamenti alla struttura aziendale in particolare per ciò che concerne la gestione delle informazioni.

Gestire un'informazione significa, secondo una definizione a grandi linee, memorizzarla in un luogo definito e facilmente raggiungibile nonché manipolarla secondo le necessità del/degli utente/i. Il tutto regolato da norme che disciplinano e supervisionano le varie operazioni.

Le tre macro-attività che hanno permesso di raggiungere questo obiettivo sono state: la realizzazione di EniWiki (il wiki aziendale), la realizzazione di un server SubVersion e per finire lo studio di Alfresco.

La prima macro-attività è stata forse la più importante per il conseguimento del

risultato finale. La necessità di creare un wiki aziendale stava diventando negli ultimi tempi un bisogno sempre più risentito dall'azienda. EniWiki ha permesso ad EniData (e alle altre aziende del gruppo) di organizzare una notevole mole di informazioni, che ad inizio stage risultavano difficili da reperire ed usare. Il secondo grande risultato conseguito con la creazione del wiki, è stato il fatto di poter documentare in modo rigoroso, tutte le attività svolte durante lo stage, garantendo all'azienda la possibilità di riutilizzare questi dati anche in momenti successivi. Tutti i dettagli relativi alla realizzazione e al popolamento di EniWiki sono spiegati nel capitolo 2.

Il secondo macro-progetto (descritto nel capitolo 3) ha consentito a TLogic (software-house facente parte del gruppo composto anche da EniData ed E-Masterit) di aumentare la propria produttività mediante l'adozione di un sistema di versionamento avanzato come lo è SubVersion. SubVersion ha permesso una revisione totale della struttura di archiviazione dei file di sviluppo di TLogic; il processo di aggiornamento ha garantito la possibilità di uniformare ogni progetto ad uno standard comune di archiviazione/manipolazione dei file. Tale standard è stato inserito nel wiki aziendale per permetterne una più completa adozione da parte di ciascun sviluppatore del team.

La terza attività è stata caratterizzata da una serie di sotto-attività di analisi e di test relative al prodotto Alfresco. Questa fase è stata molto importante per l'azienda ospitante, in quanto ha permesso di valutare le possibilità di adozione di tale prodotto in azienda; lo studio, in particolare, è stato effettuato nell'ottica di assorbire le funzionalità di Alfresco all'interno dei prodotti sviluppati da TLogic. Nel dettaglio, il software su cui si è posta la maggiore attenzione è stato HotStudio (sviluppato e commercializzato da TLogic). In più, le fasi di analisi e di test hanno permesso di evidenziare in modo preciso le reali potenzialità di Alfresco svelandone pregi e difetti del prodotto. La documentazione relativa all'analisi e ai test è presente nel capitolo 4 di questo documento ed è stata inserita per completezza nel wiki aziendale.

Per concludere voglio mettere in evidenza tutte quelle attività ausiliarie e non espressamente contenute nel piano di lavoro, le quali hanno contribuito al buon conseguimento dei risultati, nonché all'approfondimento di tematiche che mi hanno

aiutato a comprendere meglio quello che viene generalmente chiamato “mondo del lavoro”. Alcuni esempi di queste attività sono: incontri con clienti/fornitori, riunioni aziendali, ecc. In alcuni casi il mio ruolo non è stato solo passivo (di ascolto), ma anzi, in molte occasioni ho avuto modo di partecipare in modo attivo esprimendo le mie opinioni in merito agli argomenti trattati.

1.2 L'azienda

EniData fa parte di un gruppo di tre aziende, le quali cooperano per fornire un servizio completo di digitalizzazione dei documenti cartacei; le società facenti parte di questo gruppo sono: EniData, E-MasterIT e Tlogic.

EniData è la più giovane delle tre società. Il suo ruolo principale è quello di raccogliere l'esperienza di E-Masterit nel campo dell'archiviazione ottica, e le capacità di TLogic (software house competente nel campo della gestione documentale) per fornire un prodotto in grado di digitalizzare ed archiviare in modo rapido ed efficiente documenti di ogni genere. Tale prodotto risponde al nome di Organizer.

Organizer consiste di una semplice e comoda interfaccia web; ogni client è generalmente affiancato da uno scanner ottico ad alte prestazioni, per mezzo del quale ogni documento viene digitalizzato e successivamente archiviato. Organizer è oggi molto diffuso negli ambienti bancari e si sta velocemente diffondendo in tutto il mercato italiano.

E-MasterIT si occupa di attività di archiviazione ottica di documenti mediante tecnologie come OCR, ICR, ecc. L'archiviazione non si limita alla sola digitalizzazione dei documenti ma prevede tutta una serie di attività di contorno come ad esempio l'indicizzazione (con l'utilizzo di metadati) dei contenuti e l'analisi dei dati ricavati dai documenti.

TLogic rappresenta la parte informatica del gruppo. Nel corso degli anni, TLogic è riuscita a sviluppare prodotti software per la gestione documentale con caratteristiche che le permettono di differenziarsi in modo deciso dalle altre soluzioni offerte dal mercato.

HotStudio è il prodotto sul quale TLogic ha e sta investendo di più. La sua facilità d'uso e l'elevato grado di personalizzazione sono certamente alcune delle caratteristiche che hanno permesso il suo successo.

1.3 Scenario aziendale

Le tre aziende (EniData, TLogic ed E-Masterit) cooperano all'interno dello stesso stabile situato in Via Divisione Folgore (civico 38) a Vicenza. In questo luogo avvengono tutte le attività descritte nei punti precedenti come ad esempio: sviluppo di prodotti software per la gestione documentale, archiviazione ottica dei documenti cartacei, attività di data-entry, ecc.

Dal punto di vista informatico, l'azienda può contare su un gran numero di macchine (circa quindici) che fungono da server. Tali server forniscono servizi a client mobili (es. notebook, netbook, palmari, ecc) e fissi (workstation, desktop computer, ecc). Tali client possono accedere alla rete sia dall'interno (LAN, Wireless) sia dall'esterno (mediante internet e l'uso di reti virtuali protette; concetto generalmente abbreviato con la sigla VPN).

L'80% dei server è dotato di sistema operativo Linux (generalmente Ubuntu o Debian). Molti di questi server sono adibiti ad ospitare una o più macchine virtuali ognuna delle quali si specializza nella fornitura di un particolari servizio.

La creazione e l'esecuzione delle macchine virtuali è affidata al software VMWare Server, il quale è distribuito dalla società VMWare Inc a costo zero.

Tutte le macchine (virtuali o fisiche) sono accessibili, anche dall'esterno, tramite desktop remoto.

Il livello di sicurezza dell'intera infrastruttura aziendale, è assicurato dalla presenza di un robusto firewall e dalla presenza di tecniche avanzate di backup e mirroring dei dati. I backup vengono effettuati in modo incrementale; così facendo, i costi e i tempi per il salvataggio dei dati sono drasticamente abbattuti. Le attività di mirroring garantiscono invece robustezza contro eventuali guasti hardware. Per aumentare l'efficacia di queste attività, sono state predisposte tre macchine fisiche a se stanti, le quali replicano in modo consistente e continuo il contenuto di una quarta macchina

(che funge da NAS aziendale). Lo scambio dei dati tra queste macchine avviene su una rete Giga-bit la quale elimina i possibili colli di bottiglia generati da una normale rete a 10/100 Mbit.

La gestione della posta e dei fax è gestita da due server indipendenti creati ad-hoc per l'azienda. Un centralino, poi, permette le comunicazioni interne e verso l'esterno. Tale centralino si basa sul diffusissimo protocollo Voip (Voice Over Internet Protocol) gestito dal software open-source Asterisk.

1.4 Struttura del documento

In questo documento verranno illustrate passo passo le attività che hanno composto lo stage.

L'ordine di esposizione corrisponde alla reale successione nel tempo delle attività. I vari capitoli corrispondono alle macro attività svolte durante stage.

In particolare, nel secondo capitolo verranno esposte le attività relative alla creazione del wiki aziendale mettendo in evidenza le attività di analisi, le scelte e i risultati ottenuti.

Nel terzo capitolo sarà descritta la realizzazione del repository aziendale (dotato di sistema di versionamento dei file) per TLogic, il quale mira nel tempo ad aumentare la produttività dell'azienda.

Nel quarto capitolo sono invece documentate le attività di analisi relative al software ECM open source Alfresco; In particolare saranno descritte le modalità di interfacciamento dall'esterno mediante uso di API e Web Service.

Per finire nel quinto e ultimo capitolo sono presenti le considerazioni personali relative a come si è svolto lo stage nella sua interezza.

Come si nota da questa prima introduzione, le tre macro attività sono accomunate da alcuni aspetti che concernono la gestione delle informazioni a livello Enterprise.

Tutte e tre le attività, hanno lo scopo comune di aumentare nel tempo la produttività delle aziende ospitanti, migliorando la gestione delle informazioni, più di quanto non lo fosse ad inizio stage.

2 EniWiki

Prima dell'inizio dello stage la necessità di creare un wiki aziendale non era particolarmente sentita dall'azienda ospitante, ma data la notevole mole di documentazione da produrre durante tale periodo e le evoluzioni in atto all'interno dell'azienda (come ad esempio l'apertura a nuovi mercati), si è udita l'esigenza di organizzare e diffondere in modo controllato le informazioni, molto più di quanto non lo fosse fino a quel momento.

Tra le varie alternative, quella di creare un wiki è stata l'idea che fin da subito è sembrata più valida ed efficace; le informazioni da gestire sono di interesse dell'intero gruppo aziendale e non necessitano di particolari accorgimenti relativi alla diffusione (non ci sono informazioni da proteggere da occhi indiscreti). La principale forma di controllo da applicare, è quella che permette il tracciamento delle modifiche alle informazioni, mediante memorizzazione di dati come: autore, data, ecc. Così facendo si evitano cambiamenti non controllati (volontari o meno che siano).

In questo capitolo verrà illustrata la creazione di EniWiki: il wiki aziendale che ha fatto da supporto per la gestione della documentazione dello stage e farà da supporto per la documentazione esistente e futura dell'azienda.

EniWiki è stato concepito per diventare nel tempo, un punto di riferimento, liberamente e facilmente consultabile da tutti.

I punti fondamentali sviluppati in questo capito riguardano l'analisi iniziale, la scelta del motore da usare, l'installazione e la configurazione, la definizione della struttura e per finire il popolamento iniziale del wiki.

2.1 Aspettative

La creazione di un wiki interno ha la doppia finalità di:

- voler organizzare un gran numero di idee ed informazioni che ad inizio stage risultano sparse e difficili da reperire in azienda.
- voler sopperire alla necessità di documentare in modo ordinato e rigoroso le

attività che compongono lo stage per apportare valore aggiunto allo stage stesso.

I vincoli posti inizialmente ai piedi della realizzazione di EniWiki sono essenzialmente di usabilità, efficienza e semplicità. Un ulteriore vincolo riguarda la necessità di utilizzare solo ed esclusivamente strumenti open-source, per non dover sostenere costi non preventivati: EniWiki è stato pensato come mezzo sperimentale per la diffusione delle informazioni. L'azienda non ha mai avuto modo di creare e mantenere un proprio wiki, ed è proprio per questo motivo che si è voluti procedere con prudenza senza investire in prodotti troppo costosi ed eventualmente inutilizzabili.

EniWiki è stato progettato per essere utilizzato solo per uso interno, per questo motivo non si sono ritenute necessarie alcune attività di contorno come ad esempio la cura del aspetto grafico.

2.2 Scelta del motore

Un wiki non è nient'altro che un insieme di pagine web scritte in un qualsiasi linguaggio di programmazione, il quale insieme viene pubblicato su un normale web server come ad esempio Apache. Generalmente ogni wiki che si rispetti, fa uso di un database più o meno sofisticato, il quale permette la gestione delle pagine, la gestione degli utenti e la realizzazione di servizi come ad esempio: gestione delle e-mail, forum, gestione delle versioni di ciascuna pagina, ecc.

L'espressione “motore wiki” indicata quel insieme di pagine, database, moduli, eccetera, che permettono ad un wiki di esistere ed essere utilizzato.

L'individuazione del motore wiki da utilizzare è senza dubbio una scelta importante, da effettuare con le dovute valutazioni. Ogni motore di distingue infatti, per funzionalità offerte, velocità, praticità, facilità d'uso, ecc. Per questi motivi si è resa necessaria una scrematura tra i vari strumenti offerti dal mercato open-source per poter individuare, per eliminazione, il motore più adatto per creare EniWiki.

Una prima ricerca ha reso possibile l'individuazione di due principali motori che più si avvicinano alle esigenze aziendali. I prodotti candidati per essere studiati in modo più

approfondito sono stati:

- MediaWiki
- Wikka

Entrambi sono stati scelti secondo una certa logica che comprende essenzialmente queste fasi:

- prova del prodotto mediante l'uso di demo messe a disposizione nel sito del produttore
- analisi (quantitativa e qualitativa) delle funzionalità
- analisi dei requisiti hardware e software

Il primo punto in particolare, ha permesso di selezionare i prodotti più consoni soprattutto dal punto di vista estetico e pratico.

Nei prossimi capitoli verranno descritti i singoli prodotti, mettendo in evidenza i punti forti e i punti deboli di ciascuno.

2.2.1 MediaWiki

E' senza dubbio il più diffuso e usato tra i motori wiki in commercio. Tra i suoi successi è presente anche Wikipedia: la più diffusa e cliccata enciclopedia del web. Questo motore è stato creato, ed è tuttora sviluppato, dalla società non-profit “Wikimedia Foundation Inc” che ha sede a San Francisco in California.

MediaWiki è realizzato in linguaggio PHP; si appoggia ad un database MySQL in cui memorizza tutti i dati relativi a ciascuna pagina (più altri dati di supporto). Necessita della presenza di un web server con supporto PHP per poter visualizzare correttamente le pagine che lo compongono. Il server può essere installato su vari sistemi operativi tra cui Linux e Microsoft Windows.

La sua installazione è semplificata dalla presenza di una comoda interfaccia web che guida l'utente nell'installazione.

Sin dal primo contatto, MediaWiki si è dimostrano un valido candidato per la realizzazione del wiki aziendale. La sua semplicità di configurazione, installazione e

mantenimento conferiscono a questo prodotto un giudizio più che sufficiente per le necessità aziendali.

L'enorme quantità di documentazione facilita in modo notevole le fasi di configurazione e popolamento; punto molto importante, date le limitazioni di tempo previste dallo stage.

Il layout è chiaro e permette un'immediata individuazione delle informazioni all'interno delle pagine. E' disponibile un area di ricerca, in grado di individuare informazioni di ogni genere in modo completo e veloce.

MediaWiki supporta l'aggiunta di estensioni, le quali lo arricchiscono in vari aspetti, secondo le esigenze degli utilizzatori. Presso il sito ufficiale [1], è presente una vasta raccolta di estensioni applicabili in modo semplice al motore.

Il popolamento del wiki è semplice e immediato, anche se necessità di alcune conoscenze basilari di HTML, per la definizione di particolari layout delle pagine.

E' previsto un modulo per la registrazione degli utenti, il quale permette di accedere (mediante login) a funzionalità aggiuntive nonché di mantenere uno storico più dettagliato delle modifiche relative a ciascuna pagina.

Per ogni pagina è prevista un'area di *discussione*, in cui è possibile aprire dei dibattiti relativi alle informazioni contenute in quella sezione.

2.2.2 Wikka

Molto simile a MediaWiki, Wikka si presenta come una valida alternativa al motore analizzato al punto precedente. Le funzionalità offerte non hanno nulla da invidiare a MediaWiki. Realizzato anch'esso in PHP, si appoggia a MySQL per la gestione dei dati su database. L'installazione e la configurazione sono facilitate dalla comoda interfaccia Web.

La documentazione presente in rete è discreta anche se non al livello di MediaWiki.

Le informazioni sono disposte in modo efficace nelle pagine, rendendo pratica la consultazione. Anche Wikka supporta l'aggiunta di estensioni, prelevabili dal sito ufficiale del produttore [11].

2.3 La Scelta

Dato il notevole numero di analogie tra i due prodotti la scelta è ricaduta sul software che senza dubbio è più conosciuto e documentato: MediaWiki. Senza nulla togliere a Wikka, MediaWiki gode sicuramente di maggiore esperienza e diffusione. L'interfaccia web conosciuta a livello mondiale facilita le attività di popolamento e di formazione del personale. EniWiki, per come è stato concepito, potrà essere modificato da qualunque utente possa accedere alla rete aziendale.

La possibilità di consultare il codice delle pagine dei principali wiki presenti nel web, basati su MediaWiki, garantisce un valido aiuto per la creazione e la manutenzione di EniWiki; per ogni pagina creata è possibile visualizzare il codice sorgente della pagina mediante clic sul tasto “modifica” o “visualizza sorgente” a seconda che essa sia bloccata o meno da un utente.

2.4 Installazione e configurazione

Una volta scelto il prodotto da utilizzare, si è passati alla fase di installazione e configurazione. Una prima installazione è stata effettuata ai fini di test. Successivamente un'installazione definitiva ha ospitato il server che è attualmente utilizzato da EniData, Tlogic ed E-Masterit.

2.4.1 Requisiti

Per le sue doti di semplicità e robustezza, MediaWiki non necessità di risorse hardware avanzate, ma anzi, opera benissimo anche su macchine di media potenza, come un normale Personal Computer.

Il test iniziale di installazione, così come l'installazione definitiva, sono state condotte su una macchina virtuale creata inizialmente in un comune PC e trasferita successivamente in uno dei server ospitante già altre macchine virtuali utilizzate dall'azienda.

Le caratteristiche della macchina virtuale utilizzata sono le seguenti:

- numero processori virtuali: 1
- RAM assegnata: 256 MB

- disco virtuale: 4 GB
- numero unità ottiche virtuali: 1

Su tale macchina è stata installata la versione Server di Ubuntu, la quale offre tutte le funzionalità di un normale sistema operativo Linux, con un consumo di risorse limitato (grazie all'assenza di interfaccia grafica e di programmi inutili per il funzionamento del wiki).

La versione di Ubuntu installata è la 8.04 LTS (= Long Term Supported) la quale è stata rilasciata alla fine di aprire dell'anno 2008 ed è supportata dalla comunità per cinque anni; durante tale periodo continueranno ad essere rilasciati aggiornamenti di qualunque tipologia.

Terminata l'installazione del sistema operativo, il server è stato dotato di un database (MySQL 5.0), un Web-server (Apache 2) e PHP. La macchina è così diventata quella che in gergo informatico si definisce un server LAMP (Linux Apache MySQL PHP). Un server di questo tipo soddisfa in modo più che sufficiente i requisiti hardware e software richiesti da MediaWiki.

E' stato inoltre installato il programma “Imagemagick” per la generazione delle miniature delle immagini inserite nel wiki.

L'installazione si è poi conclusa con l'aggiunta del motore wiki, il quale è stato installato in modo rapido ed efficiente con l'ausilio dell'applicazione apt-get (presente di default in ogni sistema Ubuntu) la quale si appoggia al repository ufficiale della comunità di Ubuntu.

Tutti i dettagli per l'installazione di MediaWiki sono stati reperiti presso il sito ufficiale [1] (da notare che lo stesso sito è realizzato con MediaWiki).

Alla macchina virtuale è stato infine assegnato un indirizzo IP statico in modo da creare una via d'accesso certa ed univoca, per tutti gli utilizzatori.

Configurazione

Il motore fa uso di un file con estensione `.php` per configurare l'ambiente di

esecuzione di MediaWiki. In questo file possono essere impostate varie variabili e aggiunte varie estensioni, le quali estensioni sono liberamente scaricabili dal sito ufficiale di MediaWiki [1] .

Le principali variazioni apportate rispetto alla configurazione di default sono le seguenti:

- variazione della path di riferimento per il logo del wiki la quale è stata cambiata per puntare al logo ufficiale di EniWiki.
- abilitazione della funzionalità di upload dei file d'immagine da interfaccia web.
- aggiunta dell'estensione Geshi, per mettere in evidenza la sintassi dei vari linguaggi di programmazione

2.5 Definizione della struttura

Sebbene MediaWiki, come molti altri wiki, non sia dotato di una vera e propria struttura interna per le pagine, è stato utile ai fini organizzativi, stabilire uno schema di navigazione. In particolare si sono scelti due criteri principali di organizzazione:

- un area divisa in pagine relative ai clienti e pagine relative ai fornitori
- un area divisa con criterio aziendale: EniData, TLogic, E-MasterIT più un'area comune.

Per come è concepito un wiki, non si rende necessaria una struttura di questo tipo per il corretto funzionamento, ma sicuramente facilità la lettura e il mantenimento del wiki stesso.

2.6 Popolamento

Il wiki era a questo punto pronto per essere popolato e consultato. Di default MediaWiki crea un unica pagina iniziale, da cui si parte per il popolamento del wiki, più alcune pagine di servizio.

2.6.1 Concetti di base

Creazione delle pagine e dei link

La creazione delle pagine, in MediaWiki, non avviene mediante uno specifico comando, ma in modo indiretto mediante l'uso di link interni.

Nelle pagine si possono usare essenzialmente due diversi tipi di link:

- link interni: fanno riferimento a pagine del wiki stesso; la notazione da usare per la creazione di un collegamento interno è la seguente

  ```
  [[NomePagina]]
  ```

- link esterni: per collegare pagine appartenenti ad altri siti presenti nella rete locale o nel web. La notazione per creare link esterni è:

  ```
  [URL  nome del link]
  ```

  dove `URL` è l'indirizzo della pagina da collegare e `nome del link` è il testo visualizzato al posto del link.

Per i link interni, a seconda che la pagina riferita esista o meno, si ottengono due comportamenti diversi. Nel primo caso un clic sul link porta alla visualizzazione della pagina indicata; nel secondo caso il motore wiki reindirizza l'utente in un ambiente web contenente un editor di testo, con il quale è possibile creare la pagina mancante.

Scrittura delle pagine

Ogni pagina è costituita da un titolo (che corrisponde al nome del link interno per raggiungerla) e dal corpo contenente il testo della pagina stessa.

La scrittura del corpo avviene mediante uso di testo, codice HTML e comandi speciali creati da MediaWiki e dalle sue estensioni.

La possibilità di utilizzare codice HTML, facilità la definizione del layout delle pagine. I comandi speciali permettono di effettuare delle operazioni particolari come: creazione di link interni ed esterni, creazione di tabelle, inserimento di immagini presenti nel database del wiki, ecc.

2.6.2 Le estensioni

Come già anticipato, MediaWiki permette di usufruire di un gran numero di estensioni. Alcune di queste permettono di creare calendari, agende, client di posta, ecc.

2.6.2.1 GeSHi

Per Eniwiki, l'unica estensione aggiunta, è quella che permette di evidenziare (con colori e stili diversi) la sintassi di un particolare linguaggio di programmazione. GeSHi, questo è il nome dell'estensione, supporta più di cento linguaggi di programmazione tra cui Delphi e SQL, i quali sono molto utilizzati in azienda.

L'uso dell'estensione in una pagina del wiki, avviene usando un tag in stile HTML il cui nome è "`source`".

Un esempio d'uso di questo comando è il seguente:

```
<source lang=”sql”>

     ....codice da evidenziare...

</source>
```

Per il tag `source` sono inoltre disponibili i seguenti attributi:

- `line`: se posto a 1, visualizza il numero delle linee
- `source`: permette di specificare un percorso valido ad un file sorgente il cui contenuto verrà inserito ed evidenziato nel wiki.

2.6.3 Istruzioni per il popolamento

Il Popolamento di EniWiki necessita della definizione di uno standard. Tale Standard permetterà, se rispettato, di popolare il wiki in modo produttivo: rendendo chiari i contenuti grazie all'applicazione di uno stile prestabilito.

Lo standard è stato inserito in una pagina di EniWiki chiamata "StandardEniWiki". Per incentivare il rispetto di queste regole, un riferimento a tale pagina è stato inserito anche nel menu laterale, il quale è presente in ogni pagina del wiki.

I punti trattati dallo standard, riguardano essenzialmente:

- il layout delle pagine
- la leggibilità e le convenzioni sul codice sorgente delle pagine
- standard di nominazione delle pagine e delle immagini

Lo stesso standard potrà essere nel tempo ampliato e migliorato; ovviamente tali modifiche dovranno essere ben giustificate e discusse con l'amministratore del wiki.

2.7 Conclusioni

Fatto lo standard, il wiki è stato pubblicato nella rete e popolato giorno per giorno. L'indirizzo per l'accesso è stato reso noto a tutti i membri dell'azienda in modo tale che tutti potessero (e possano tutt'ora) consultarlo ed eventualmente arricchirlo. Una volta digitato l'indirizzo IP nel proprio browser, l'utente viene automaticamente indirizzato nella pagina principale di EniWiki, e da li potrà iniziare la sua ricerca.

Il “redirect automatico” alla pagina iniziale del wiki, è stato implementato modificando il file di configurazione di Apache: specificando la cartella di default in cui cercare le pagine web. Questa modica consiste in un cambiamento indolore, in quanto nella macchina virtuale ospitante EniWiki, non sono presenti altri siti web.

Come specificato dallo standard, è desiderabile che ciascun utente utilizzatore del wiki, si doti di un account, così da permettere un uso più produttivo di EniWiki; in questo modo le eventuali modifiche apportate ad una pagina potranno essere tracciate con cura, potendo eventualmente risalire all'autore della modifica. Non solo, mediante l'accesso con il proprio account, un utente può usufruire di funzionalità altrimenti non attive, come ad esempio il caricamento di nuove immagini.

3 Server SVN

Gestire un gran numero di file, può diventare un problema se non si prendono le dovute precauzioni.

Nelle software-house ad esempio, vengono spesso sviluppati vari progetti software, ognuno dei quali è soggetto a continue modifiche, le quali modifiche possono riguardare un numero variabile di file; se poi gli utenti che manipolano i file sono molti, il problema si espande fino a diventare difficilmente gestibile. Anche un progetto di dimensioni ristrette, può essere facilmente costituito da centinaia di file. Capire come affrontare il problema non è stata certo cosa facile.

Fortunatamente il mercato open-source ci viene in aiuto anche questa volta. Non sono pochi infatti, i programmi in grado di gestire il versionamento dei file. Molti di questi godono di una notevole esperienza nel settore, nonché di un'ampia diffusione anche in aziende di spessore.

Quando si parla di versionamento di file si intendono quegli strumenti e quelle operazioni (automatiche o manuali) in grado di gestire nel tempo le modifiche a uno o più file. Il versionamento avviene mediante l'uso di un numero di versione il quale è univoco e progressivo. Utilizzando questi numeri di versione è possibile confrontare nel tempo, il contenuto di uno o più file. Il versionamento rende poi possibili molte altre operazioni (anche sofisticate), come per esempio la creazione di versioni parallele ed indipendenti di uno o più file.

Nonostante l'esistenza di questi prodotti, il problema non può essere risolto in modo semplice e veloce. Se poi lo scenario d'applicazione è complesso e basato su tecniche obsolete di gestione dei file, la faccenda diventa al quanto complicata da gestire.

E' inoltre doveroso precisare che questi strumenti non hanno alcuna efficacia se, all'interno dell'azienda, non sono previsti mezzi e tecniche per l'organizzazione delle risorse, e per la comunicazione tra le persone che ne fanno uso.

TLogic sente da tempo la necessità di migrare verso un sistema di gestione dei

progetti software più sofisticato e produttivo. Lo stage ha dato la possibilità, ha TLogic di studiare e successivamente adottare un sistema moderno di gestione del versionamento dei file.

I successivi capitoli, descrivono in modo dettagliato questo adattamento tecnologico, partendo con la descrizione della situazione iniziale presente in azienda e terminando con il risultato ottenuto a fine stage.

3.1 La situazione iniziale

TLogic gestisce ad oggi 3 macro-progetti software ben distinti:

- HotStudio: software di gestione documentale
- QuickDoc: software di gestione documentale su web con caratteristiche simili a HotStudio
- Organizer: software di gestione della clientela.

Oltre a questi, TLogic mantiene una molteplicità di progetti che si distinguono dai precedenti per dimensione ed anzianità. Alcuni di questi progetti vengono rispolverati in alcuni occasioni per particolari esigenze dei clienti, ma non sono soggetti ad attività pianificate di sviluppo e/o manutenzione.

I progetti di TLogic sono sviluppati in diversi linguaggi di programmazione e possono in alcuni casi far uso di librerie comuni sviluppate dalla stessa TLogic.

La software-house in questione sviluppa ormai da diversi anni una propria libreria interna, la quale permette di semplificare molte attività di sviluppo, usufruendo di tutti i vantaggi del riuso sistematico del codice. Queste librerie sono in continuo sviluppo e necessitano sicuramente di un sistema di versionamento efficiente ed efficace.

Il principale linguaggio di programmazione usato da TLogic è Delphi il quale gode di un'ottima fama in ambiente Microsoft Windows. Tale linguaggio è al giorno d'oggi in continuo sviluppo e nel corso degli anni ha subito notevoli cambiamenti. Proprio per quest'ultimo motivo, i prodotti basati su Delphi sono mutati nel tempo, rendendo a volte impossibile il riuso di codice scritto per versioni troppo vecchie del linguaggio che fonda le sue radici su Pascal.

E' facile immaginare come la situazione in azienda fosse, ad inizio stage, al quanto complessa. Molti dei progetti esistenti, tra cui anche le librerie citate sopra, si presentavano inizialmente organizzate in un file system che, nel corso degli anni, stava diventando sempre più difficile da gestire e di conseguenza inefficiente.

Ogni progetto presentava, allo stato iniziale dell'analisi del problema, una serie di sottocartelle con lo scopo di creare una sorta di versionamento manuale. Talvolta questo versionamento era ripetuto in più sottolivelli.

Il criterio generale di versionamento era il seguente:

- suddivisione dei progetti per cliente
- versionamento delle principali release a livello di progetto mediante uso di cartelle del tipo: “`v_x.y`”, dove `x.y` è la versione contenuta nella cartella corrispondente.
- versionamento dei sotto-moduli di un progetto con l'uso di cartelle con nome “`fix_z`” dove `z` è il numero di versione del modulo contenuto nel progetto di versione `x.y`.

Non è difficile capire che un versionamento di questo tipo non può che rendere ostica la gestione dei progetti software.

Le rimanenti aziende del gruppo, soffrono anch'esse di problemi simili (seppur in dose minore); anche per esse, come per TLogic, non sono previsti sistemi di versionamento automatizzato dei file. Risolvere il problema di TLogic costituisce sicuramente un valido punto di partenza per migliorare anche la situazione di EniData ed E-MasterIT.

3.2 Le aspettative

Passare ad un nuovo sistema di gestione dei file, è sicuramente un punto delicato, e in quanto tale necessità di un'attenta analisi. Occorre approfondire con molta precisione i seguenti punti:

- esigenze dei futuri utilizzatori del nuovo sistema di versionamento

- caratteristiche del sistema di gestione dei file presente ad inizio stage
- struttura ideale del nuovo sistema di versionamento.

3.2.1 Requisiti

Capire quali fossero le esigenze del team di sviluppo di TLogic era più facile a dirsi che a farsi; il numero dei componenti che costituiscono TLogic è ristretto e si appoggia per molti aspetti a ditte esterne.

Per definire in modo dettagliato le esigenze di TLogic, durante la fase di analisi sono stati organizzati dei momenti di discussione in cui analizzare il problema e definire le principali esigenze del team. In queste occasioni è stata analizzata la struttura del repository aziendale presente ad inizio stage, evidenziando pregi e difetti del sistema esistente.

I requisiti individuati dalla fase di analisi del problema sono elencati nella tabella1.

Ad ogni requisito è stato assegnato un codice identificativo, il quale facilità le operazioni di tracciamento e il loro riferimento all'interno dei documenti. I requisiti individuati sono stati divisi in cinque macro-categorie le quali permettono una più facile organizzazione e gestione degli stessi.

Codice	Descrizione
SVN_R1	Autenticazione
SVN_R1.1	Più utenti possono essere connessi contemporaneamente sullo stesso server su repository diversi
SVN_R1.2	Più utenti possono essere connessi contemporaneamente sullo stesso repository
SVN_R2	Dati
SVN_R2.1	Non devono esserci particolari vincoli di spazio a meno dei limiti fisici di memorizzazione
SVN_R2.2	Tutti i file devono poter essere versionati
SVN_R2.3	Deve essere possibile specificare una “black-list” per i singoli repository, con la quale si escludono dal versionamento particolari file

SVN_R2.4	Devono essere previsti meccanismi per garantire l'accesso esclusivo a tutto o parte del repository (ad esempio dei lock)
SVN_R2.5	Le operazioni di commit e update devono poter riguardare anche solo parte del repository.
SVN_R2.6	Branch
SVN_R2.6.1	Deve essere prevista la possibilità di creare rami indipendenti del repository
SVN_R2.6.2	I rami (o branch) devono poter essere confrontati in modo agevole
SVN_R2.6.3	Deve essere prevista una funzionalità per riunire due rami in modo rapido e consistente
SVN_R2.6.4	Un ramo deve poter essere chiuso (abbandonato).
SVN_R2.6.5	Rami diversi devono poter essere mantenuti allineati (aggiornati) l'uno rispetto l'altro
SVN_R3	Aspetti generali
SVN_R3.1	Il sistema deve avere prestazioni soddisfacenti
SVN_R3.2	Il server deve poter essere installato su sistemi operativi diversi (Linux e Microsoft Windows)
SVN_R3.3	Devono essere disponibili client per i più diffusi sistemi operativi (Linux, Microsoft Windows e Mac OS X)
SVN_R4	Automatismi
SVN_R4.1	Devono essere previsti dei sistemi per l'automatismo di particolari operazioni, come ad esempio la copia automatica di file da un repository all'altro.
SVN_R5	Amministrazione
SVN_R5.1	I tempi di installazione del server devono essere brevi
SVN_R5.2	I repository devono poter essere creati in modo semplice e veloce
SVN_R5.3	I repository devono poter essere riorganizzati in modo agevole sia al loro interno, sia a livello di memorizzazione su file system del server

Tabella1: Requisiti del sistema di versionamento

3.2.2 Il vecchio sistema di gestione dei file

La situazione descritta al punto 3.1 lascia intuire come la necessità di passare ad un nuovo sistema di gestione dei file fosse molto sentita da parte di TLogic.

Analizzando il sistema esistente, si sono potute ricavare informazioni utili sulle caratteristiche che il nuovo sistema di versionamento avrebbe dovuto possedere.

Queste informazioni sono state utilizzate come fonti per popolare la tabella1.

3.2.3 Struttura ideale del nuovo sistema di versionamento

Per completare l'analisi del problema, è stato utile definire una presumibile struttura che il nuovo sistema avrebbe dovuto possedere.

Il sistema esistente gestiva i vari progetti secondo una classificazione gerarchica per cliente e successivamente per progetto. Questo modo di gestire i progetti ha sicuramente suggerito una valida proposta per la strutturazione del nuovo repository.

Ciascun progetto presentava al proprio interno un certo numero di file organizzati secondo diverse logiche; taluni progetti distinguevano in primo luogo i moduli che lo componevano e successivamente in ciascun modulo vi era una classificazione dei file che generalmente consisteva nella distinzione tra: file sorgenti, documenti, librerie, file binari, ecc.; altri progetti invece, eseguivano solo ed esclusivamente una classificazione dei file per tipologia (es. sorgenti, documenti, ecc.).

Il nuovo repository avrebbe dovuto standardizzare questi aspetti, definendo una struttura base su cui ogni progetto avrebbe dovuto basarsi in modo più o meno rigoroso (alcune eccezioni sono state preventivate per particolari esigenze isolate).

Oltre ai progetti software, nel repository iniziale, erano presenti anche altri generi di file: file di setup, documentazione generale, ecc. Per queste particolari situazioni sono stati creati dei repository ad-hoc per permettere il loro versionamento. Per quanto riguarda la documentazione, essa verrà trasferita per la maggior parte sul nuovo wiki aziendale analizzato al punto 2 del presente documento; così facendo, verrà migliorata la sua diffusione e gestione, aumentando di conseguenza la produttività dell'azienda.

3.3 SubVersion

Per i motivi che andremmo ad analizzare in questo capitolo, il server scelto ed utilizzato per il nuovo sistema di versionamento, è stato SubVersion altrimenti

conosciuto come SVN.

La scelta del sistema da utilizzare per la creazione del nuovo repository è stata guidata per la maggior parte dai seguenti punti:

- open-source: nessun costo
- efficienza: veloce, versatile e di elevata qualità
- diffusione: forte diffusione in tutto il mondo
- continuità di sviluppo: mantenuto nel tempo
- portabilità: compatibilità con vari sistemi operativi.

SVN è di gran lunga, il software più usato per il versionamento dei file; progettato da CollabNet, SubVersion eredita da CVS (Concurrent Versions System) molte funzionalità e logiche. Nel momento in cui viene redatto questo documento, SubVersion è disponibile alla versione 1.5.4.

Di seguito sono elencate le principali caratteristiche in aggiunta a quelle possedute da CVS:

- vengono versionate le directory, i cambi di nome, i link simbolici e i metadati
- le operazioni di commit sono azioni atomiche; un'eventuale interruzione di un commit, non lascia il repository in uno stato inconsistente
- le operazioni di tagging e branching sono eseguite in modo veloce e indipendente dalla dimensione dei file
- il protocollo di comunicazione tra client e server, invia solo le differenze in entrambe le direzioni, limitando i costi di comunicazione alla quantità di modifiche anziché alla mole di dati
- gestione efficiente dei file binari
- l'output prodotto da ciascuna operazione è analizzabile da programmi esterni.

La configurazione di default di SVN prevede delle politiche d'accesso ai repository prive di particolari vincoli; in sostanza la configurazione di default permette l'accesso

in lettura e scrittura a tutti gli utenti verso tutti i repository.

Come è giusto aspettarsi da un software di così larga diffusione come lo è SVN, sono previsti vari metodi per mettere in sicurezza l'intero repository o una sua parte. Usufruendo ad esempio dell'integrazione di SVN in Apache, è possibile definire dei diritti d'accesso per utenti e/o gruppi garantendo un accesso limitato e controllato, ai vari repository.

Il reparto sicurezza si arricchisce poi con l'appoggio a tecnologie quali Message Digest e uso del diffuso protocollo SSL.

Nei prossimi capitolo saranno descritte, tra le altre cose, anche le politiche di sicurezza definite per l'azienda.

3.3.1 Alcuni test

Alcuni giorni sono stati dedicati per eseguire alcune prove pratiche su un server di prova, per capire e toccare con mano le reali funzionalità di SubVersion, nonché per definire una strategia di progettazione del nuovo repository.

Questi brevi test sono stati effettuati installando un server SVN su una macchina esistente dotata del sistema operativo di casa Microsoft: Windows Server 2003. Il client usato per le prove è stato TortoiseSVN il quale si integra in maniera eccellente con gran parte dei sistemi operativi Microsoft; questa integrazione è resa possibile grazie alla presenza di un menu contestuale presente ad ogni clic destro del mouse su file e cartelle.

I test sono stati eseguiti mediante la creazione di alcuni repository di prova, all'interno dei quali sono stati inseriti dei file molto simili a quelli che sarebbero andati a popolare il repository finale.

Le prove, oltre a confermare le promesse del produttore (in termini di funzionalità e prestazioni), hanno permesso di capire il livello di soddisfazione di ciascun requisito definito nella Tabella1.

3.4 Progettazione del nuovo repository

Individuati i requisiti e gli strumenti da usare, si è passati alla fase di progettazione del nuovo repository.

In questa fase sono state definite le strutture standard di archiviazione dei dati, nonché alcune strategie per permettere l'interazione dei dati presenti in repository diversi.

Come già anticipato nei capitoli precedenti, i progetti di TLogic, fanno uso di una libreria comune sviluppata anch'essa da TLogic. Questa libreria, come specificato in seguito, sarà immagazzinata in un repository a se stante, il quale sarà quindi dotato di un numero di versione indipendente da ogni altro progetto. Durante la fase di analisi è sorta, tra le altre necessità, quella di mantenere una copia dei sorgenti di tale libreria in ciascun progetto che ne faccia uso, il che significa copiare parte del repository contenente le librerie, negli altri repository contenenti i progetti che ne fanno uso. Questa esigenza scaturisce dal bisogno di avere una correlazione tra le librerie e i singoli progetti. Questo aspetto progettuale sarà analizzato con più attenzione nei prossimi paragrafi.

3.4.1 Struttura del repository

Per come è concepito SVN, la struttura del nuovo repository avrebbe potuto essere essenzialmente di tre tipi:

- creazione di un unico repository contenente tutti i progetti e le librerie sviluppate: un unico numero di versione per molti progetti; la regressione ad una versione passata può cambiare lo stato di molti progetti contemporaneamente
- un repository per cliente: un numero di revisione per tutti i prodotti forniti ad un certo cliente
- un repository per progetto: ogni progetto è versionato in modo indipendente dagli altri.

Quest'ultima possibilità è quella che meglio si è adattata alle esigenze di TLogic; avere un numero di versione indipendente per ogni progetto, rende la gestione del

repository elastica ed efficiente. Tra gli altri vantaggi di questa strategia, troviamo il fatto che, avere repository separati ed indipendenti non impedisce la possibilità di organizzarli secondo un ulteriore grado di astrazione: ad esempio ragruppando i progetti per clienti.

Ogni repository è stato poi dotato della struttura che è diventata ormai uno standard per SubVersion; questa sotto-struttura prevede la creazione delle cartelle:

- trunk: il filone principale del repository
- tags: contenente le versioni statiche del repository le quali una volta create non sono generalmente soggette ad alcuna modifica nel tempo; nella pratica queste versioni corrispondono alle “major release” di un progetto
- branches: contenente alcuni rami del progetto i quali viaggiano in parallelo con la versione presente nella cartella “trunk”; tra questi rami possono essere eseguite operazioni di unione, confronto, aggiornamento, ecc.

Grazie a queste cartelle è possibile realizzare e mantenere versioni diverse di uno stesso progetto; i “rami” sono indipendenti e caratterizzati da numeri di versione separati.

Per ciascun ramo di ciascun repository è stata poi definita una struttura standard per la suddivisione dei file secondo il loro tipo; la struttura definita è indicata dalla figura1.

- */*
  - *bin* : file binari
  - *data* : dati
    - *db* : database
    - *file* : usato per i progetti di gestione documentale
  - *doc* : documentazione
  - *lib* : librerie
  - *obj* : file compilati
  - *res* : risorse (es. immagini, video, ecc.)
  - *src* : sorgenti
    - *modulo1*
    - ...
    - *moduloN*

Figura1: struttura del singolo repository

Con questa struttura ogni progetto è stato uniformato ad uno standard aziendale che facilita le operazioni di manipolazione dei repository.

Per garantire la corretta applicazione della struttura nel tempo, si è resa necessaria la configurazione degli ambienti di sviluppo; in questo modo i file oggetto prodotti dalla compilazione di un progetto vengono riposti automaticamente nella cartella `obj`, le librerie vengono prelevate dalla cartella `lib`, ecc.

3.4.1.1 Organizzazione dei repository su server

Come anticipato il nuovo repository risulta composto da tanti piccoli sotto-repository; l'elevato numero di quest'ultimi a reso necessaria la definizione di un criterio di archiviazione dei repository su file system lato server.

Concretamente la macchina ospitante il server SVN è stata dotata di una unità disco

a se stante. In tale unità sono stati riposti tutti i repository archiviandoli come mostrato dalla Figura2.

- /
  - clienti
    - tlogic
      - progettoY
    - clienteX
      - progettoZ
  - doc
  - lib
  - fornitori
  - script

Figura2: struttura file system server

Come si evince dalla Figura2 la cartella `clienti` contiene tutti i progetti sviluppati da TLogic. Per definire un unico criterio di classificazione, si è scelto di riporre i progetti in fase di sviluppo nella cartella `clienti/tlogic` come se TLogic fosse cliente di se stessa. La cartella `doc` contiene poi, documentazione generica, la quale non è specifica di nessun progetto e/o riguarda tutti i progetti software. `Lib` contiene la libreria sviluppata da TLogic la quale è usata in modo più o meno completo dagli altri progetti. Dentro `fornitori` sono presenti tutti i file relativi ai fornitori di TLogic, mentre `script` contiene alcuni file per l'automazione di alcune operazioni; tali file saranno analizzati nel capitolo 3.4.3.

3.4.2 Accesso al repository

Data la delicatezza dei dati contenuti nel repository, si è scelto di attivare la modalità di autenticazione degli accessi per gli utenti. Essendo il repository raggiungibile solo dalla rete interna, la modalità di autenticazione scelta non è delle più sofisticate; questa scelta permette di ottenere un ottimo compromesso in termini di prestazioni e sicurezza.

Se in futuro si dovesse ritenere necessario l'accesso dall'esterno al server SVN, si potrà procedere in due diversi modi: il primo consiste nel permettere l'accesso al web server anche dall'esterno (questo necessiterà l'utilizzo di strumenti di autenticazione più potenti come ad esempio il protocollo SSL); il secondo modo consiste nel creare una rete virtuale protetta tra la rete di TLogic e quella dell'utilizzatore esterno. Questo secondo metodo fa uso consistente dei concetti di VPN e firewall.

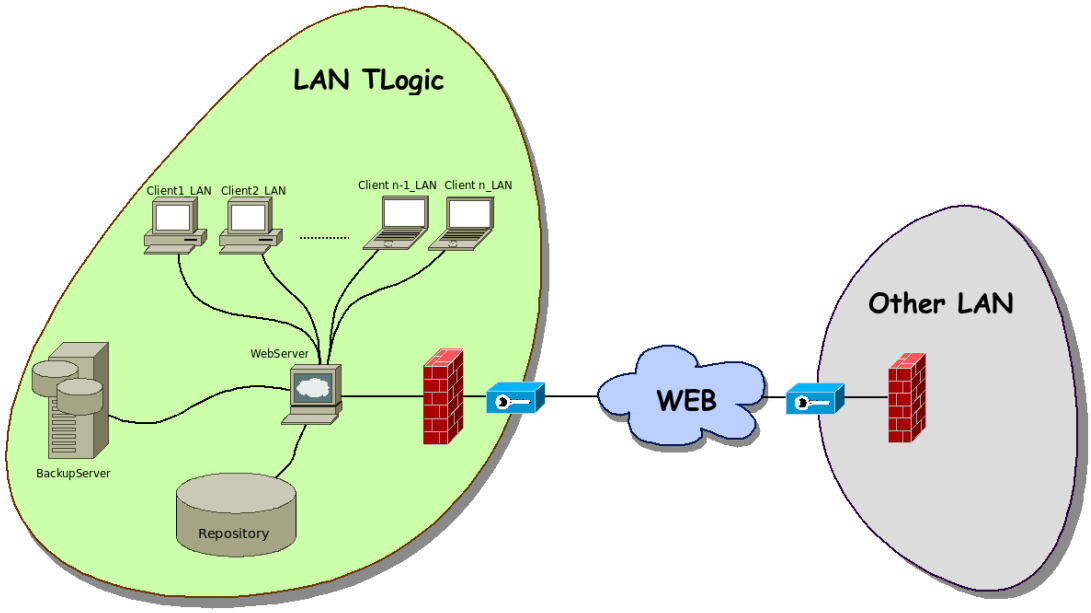

Figura3: accesso al repository dall'esterno mediante VPN.

La politica di sicurezza scelta prevede la definizione di utenti e gruppi, nonché di diritti d'accesso ai repository nella loro interezza o in parti di essi.

Per abilitare queste funzionalità è stato necessario operare sui file di configurazione

di Apache in quanto l'unico protocollo di accesso ai repository definito, è stato HTTP. Ogni comunicazione da e verso il repository avviene attraverso questo protocollo.

3.4.2.1 Gli indirizzi di accesso

Gli indirizzi di accesso ai repository non sono altro che dei comuni indirizzi web. Nei file di configurazione di apache sono stati specificati i repository raggiungibili via HTTP.

L'indirizzo HTTP di accesso ad un repository non coincide necessariamente con la posizione fisica di archiviazione su server, dato che essa può essere variata dall'amministratore semplicemente agendo sui file di configurazione del web server.

Nel caso di TLogic esiste una corrispondenza totale tra percorso di archiviazione su disco dei repository e indirizzi HTTP di accesso.

Esempio:

per accedere al repository ProgettoY indicato nella figura2, è semplicemente necessario fare riferimento all'indirizzo:

http://IP_DelServerSVN/svn/tlogic/progettoY

3.4.3 Hook Script

Tra le molteplici funzionalità di SubVersion, ce né una che permette l'esecuzione di alcuni script definiti dall'utente al verificarsi di particolari eventi. Questa funzionalità è resa possibile grazie all'uso degli Hook Script.

Questi script sono dei normali file `.bat` o `.sh` (a seconda del sistema operativo in uso) i quali possono essere invocati al verificarsi dei seguenti eventi:

- inizio commit: il momento in cui si avvia un commit
- pre-commit: il momento in cui si conferma un commit
- post-commit: terminazione di un commit
- pre-update: inizio di un update

- post-update: fine di un update

Normalmente questi script vengono usati a lato server, ma se il client lo supporta, possono essere usati anche a lato client.

Nel caso di TLogic questi script sono stati utili per permettere di risolvere il problema descritto al punto 3.4: sincronizzare i sorgenti delle librerie presenti nel repository omonimo, con la cartella librerie presente in uno dei repository facente uso. Questa tecnica, come anticipato, permette di mantenere un legame tra la versione delle librerie e la versione di un determinato progetto.

Dato che questa operazione è utile solo per motivi di versionamento, può essere tranquillamente eseguita solo nel momento in cui il repository di uno dei progetti (facente uso delle librerie) viene caricato nel server e aggiornato ad un nuovo numero di versione.

L'evento da catturare è quindi quello di “inizio commit”. Inoltre l'operazione deve essere eseguita a lato client quindi lo script dovrà essere eseguito dal client SVN anziché dal server.

Lo script è stato inserito nella cartella `script` presente nell'unita disco contenente tutti i repository. Tale script riceve in ingresso due parametri: la cartella di origine e quella di destinazione delle librerie.

Ogni client è stato configurato in modo tale da eseguire questo script ogni qual volta l'utente avvia un operazione di commit.

Gli hook script sono specifici per ogni repository, quindi per ogni copia locale del repository sono stati configurati i client in modo da eseguire lo script con i giusti parametri in ingresso.

3.5 Il risultato finale

Terminata la progettazione del repository è iniziata la parte di migrazione dei progetti esistenti per poi passare all'uso del nuovo repository.

Tutti i repository necessari sono stati creati all'inizio; per ognuno di essi è stata creata la struttura standard descritta nei capitoli precedenti; sono stati definiti i diritti

d'accesso stabiliti e per finire sono stati impostati gli eventuali hook script da eseguire.

La migrazione dei progetti esistenti è stata effettuata da un componente del team di TLogic in quanto l'adattamento delle strutture dati esistenti al nuovo standard richiede conoscenze piuttosto avanzate sul funzionamento dei prodotti software sviluppati; per questo motivo io mi sono attenuto a supervisionare il lavoro, e a documentare il tutto in EniWiki.

L'intero processo di migrazione è stato fatto mantenendo una copia del vecchio repository; questo per evitare che eventuali incidenti di percorso potessero compromettere in qualche modo informazioni vitali per l'azienda.

Naturalmente non tutti i repository sono stati migrati nello stesso momento; il processo di migrazione, è durato diversi giorni e tuttora alcuni repository risultano ancora da migrare. Il motivo di questa dilazione temporale, è da imputarsi alla necessità di mantenere attivo il processi di sviluppo e manutenzione dei prodotti software di TLogic, evitando quindi un'interruzione totale delle attività produttive dell'azienda.

A mano a mano che i repository venivano migrati, ambienti di sviluppo e client SVN (generalmente TortoiseSVN) sono stati aggiornati per ogni postazione di lavoro in modo da non creare punti di inconsistenza nel sistema. Tutta l'attività di configurazione, è stata agevolata dall'esistenza del nuovo wiki aziendale il quale si è dimostrato un valido strumento per l'azienda.

3.6 Conclusione e note personali

Ad attività conclusa, l'azienda ha dimostrato un grado di soddisfazione elevato per i risultati ottenuti.

SubVersion si è dimostrato uno strumento all'altezza della situazione, adattandosi agevolmente alla realtà aziendale esistente.

A mio parere TLogic gioverà molto di questa innovazione soprattutto nel medio-lungo periodo, in quanto le fasi iniziali potranno essere caratterizzate da alcuni momenti di incertezza e disorientamento.

Grazie al nuovo sistema di versionamento, TLogic potrà godere di enormi vantaggi soprattutto dal punto di vista della gestione concorrente delle versioni di uno stesso prodotto; cosa che finora risultava difficile da gestire.

4 Alfresco

HotStudio, uno dei prodotti di TLogic, è un software di document management in grado di offrire funzionalità di archiviazione dei documenti digitali e cartacei (mediante scanner e funzioni OCR integrate) per poi permettere operazioni di ricerca, assegnazione di documenti ad utenti, ecc. Ad ogni documento possono essere associati dei metadati, i quali permettono l'esecuzione di operazioni più sofisticate come la ricerca per categoria documentale, ecc..

I punti di forza di HotStudio sono sicuramente l'interfaccia grafica semplice ed intuitiva nonché l'enorme elasticità di configurazione del prodotto.

A causa di alcune lacune dell'attuale versione, TLogic ha deciso di investire parte del suo tempo per analizzare le alternative possibili per lo sviluppo della nuova versione di HotStudio. Tale versione (3) sarà caratterizzata da una totale revisione del codice esistente.

Fatte le dovute considerazioni TLogic ha stabilito i punti chiave su cui la nuova versione dovrà basarsi:

- repository documentale esterno all'applicazione
- interfaccia grafica rivista

E' proprio per il primo punto che Alfresco entra in scena.

Alfresco è un software di gestione documentale e si avvale di un proprio repository interno per l'archiviazione dei dati. Tale repository può essere interrogato e manipolato grazie ad una complessa e completa libreria, la quale mette a disposizione una serie di API accessibili da vari linguaggi di programmazione.

Nella visione di TLogic, Alfresco può essere visto come un valido candidato per l'esternazione del repository di HotStudio.

La parte finale dello stage è stata incentrata proprio su questa idea. Tra i punti fondamentali di questa attività di fine stage troviamo:

- l'analisi del software Alfresco
- lo studio delle API di Alfresco
- le prove di interfacciamento con il nuovo repository mediante un'applicazione di prova
- l'analisi della possibile integrazione di Alfresco in HotStudio

I prossimi capitoli andranno ad approfondire questi punti descrivendo in modo dettagliato lo svolgimento delle ultime due settimane di stage svolte presso EniData.

4.1 Cos'è Alfresco

Alfresco rientra nella categoria dei software ECM (Enterprise Content Management); questa macro-categoria comprende tutti quei prodotti in grado di fornire supporto per la gestione delle informazioni a livello Enterprise.

Alfresco in particolare è in grado di offrire funzionalità di:

- Document Management
- Collaboration
- Content Platform and Repository
- Content Management Interoperability Services (CMIS)
- Records Management
- Enterprise Content Search
- Web Content Management
- Image Management

La presenza di tante caratteristiche non impedisce all'utente di usufruire solo a parte di esse. Per TLogic ad esempio, l'attenzione principale è stata riposta sulle funzioni di: Document Management (per la gestione documentale), Content Platform and Repository e CMIS (per l'uso delle API di interfacciamento con il repository).

Prodotto dalla Alfresco Software Inc, Alfresco viene distribuito in due versioni, le quali

condividono lo stesso codice sorgente; tale codice è completamente open source. Le versioni disponibili sono:

- Alfresco Enterprise Edition
- Alfresco Labs

La prima viene distribuita con licenza commerciale per CPU, mentre la seconda è liberamente scaricabile dal sito della comunità [8]. La principale differenza tra le due versioni, è l'elevato grado di stabilità offerto dalla versione Enterprise, la quale gode della certificazione del codice e del supporto da parte di personale specializzato. La versione a pagamento, inoltre, è stata sottoposta a diversi stress test da parte della software house produttrice, i quali conferiscono un ulteriore valore aggiunto al prodotto.

La versione Labs è indicata per utilizzi in realtà produttive limitate, nonché per chi vuole cominciare ad avvicinarsi ad Alfresco; la versione Enterprise, è invece indicata per gestire grandi moli di dati, come accade nelle medio/grandi imprese.

Mentre viene scritto questo documento, la più recente versione disponibile di Alfresco è la 3.0.

4.1.1 Dettagli tecnici

Il grado di portabilità di Alfresco è elevato: può essere installato su sistemi Windows, Linux e Mac OS X.

Se non ci sono particolari esigenze, il consiglio è quello di installare Alfresco su un ambiente Linux; questo permette di usufruire di tutte le doti di stabilità e velocità, per le quali Alfresco è stato concepito.

Completamente scritto in Java, Alfresco fa uso della tecnologia JSP (Java Server Pages); per funzionare Alfresco necessita della presenza su lato server della piattaforma JDK (Java Development Kit) e di JVM (Java Virtual Machine).

Un database esterno di appoggio, è essenziale per l'utilizzo di Alfresco. L'interfacciamento con il database viene gestito da Hibernate il quale non è altro che un layer intermedio che permette il collegamento a DBMS diversi, semplificando e

unificando l'interfaccia di comunicazione. Tra i DBMS supportati, troviamo: MySQL (consigliato per grandi realtà aziendali), Microsoft SQL Server, Oracle, ecc.

Il web server utilizzato da Alfresco è Apache TomCat.

Per lo svolgimento di particolari funzioni, Alfresco si appoggia a componenti esterni open source come:

- Apache Lucene: motore di ricerca full-text ad alte prestazioni
- Hibernate: framework di astrazione per database (già citato sopra)
- JBoss JBPM: gestione workflow

4.2 La sua importanza in azienda

Come anticipato nella parte introduttiva di questo capitolo, TLogic mira a spostare all'esterno il repository documentale di HotStudio.

I vantaggi che si possono ricavare da questa scelta sono significativi:

- possibilità di gestire lo stesso repository da applicazioni diverse (quindi non solo da HotStudio)
- possibilità di usufruire delle caratteristiche di certificazione di Alfresco per aggiungere valore ad HotStudio
- concentrare lo sviluppo di HotStudio su aspetti che non comprendono la manutenzione di un repository ad-hoc.
- ammodernare l'applicazione grazie all'adozione di un repository più performante e collaudato

D'altra parte, passare ad un nuovo repository comporta degli oneri quali:

- costi di analisi del nuovo repository
- sviluppo di una piattaforma di aggancio al nuovo repository
- verifica dei risultati e test

Lo stage ha aiutato TLogic a rispondere al primo punto: l'analisi di Alfresco.

4.3 HotStudio oggi

La versione attuale di HotStudio (la 2.4) permette la gestione documentale mediante un repository integrato; tale repository è stato creato dalla stessa TLogic fin dalla nascita di HotStudio. Questo repository si è dimostrato negli anni un valido strumento, all'altezza della situazione. Le scelte implementative alla base di questo repository, sono molto simili a quelle di Alfresco. Questo lascia immaginare l'efficacia di tali scelte.

Tuttavia, TLogic si è trovata negli anni, a dover sostenere una serie di cambiamenti (provocati dalle esigenze dei clienti) i quali hanno contribuito alla diminuzione della qualità del codice di HotStudio.

L'idea di una nuova versione di HotStudio mira sicuramente a ridurre questi problemi. Utilizzare un repository esterno è certamente un'ottima alternativa per iniziare questa fase di aggiornamento.

Durante questo processo di ammodernamento, le funzionalità di HotStudio non dovranno per nessun motivo essere mutate.

4.3.1 Specifiche tecniche

HotStudio (versione 2.4) permette di gestire documenti o più in generale file (quindi anche immagini, pdf, ecc.) in un repository. Tale repository consiste in una serie di cartelle e sottocartelle in cui sono memorizzati i vari file. La struttura gerarchica di queste cartelle, è molto simile a quella di Alfresco: i documenti sono organizzati per anno, mese e giorno di creazione/inserimento. Quindi ad esempio i file inseriti il giorno 13/01/2008 saranno contenuti nella gerarchia di cartelle: /2008/01/13/

Ogni file è poi rinominato sostituendo il nome originale con un nome contenente alcune informazioni come data, id (identificativo univoco), ecc.

Il repository può essere dislocato in più volumi di archiviazione; un volume di archiviazione è un percorso che identifica la locazione in cui sono memorizzati i file. La possibilità di gestire più volumi di archiviazione, permette di rendere più flessibile la gestione della memorizzazione dei file (ad esempio memorizzare in volumi diversi, file relativi a clienti e file relativi a fornitori).

Per ogni documento archiviato possono essere specificati dei metadati, i quali permettono di aggiungere informazioni ad un file. Tali informazioni possono essere ricavate in automatico dal documento oppure possono essere inserite a mano dall'utente. Un metadato in HotStudio può essere un normale campo di tipo numero, testo o data, oppure un campo più complesso, come ad esempio un dato ricavato da una tabella di un database.

I metadati sono personalizzabili dall'utente mediante l'interfaccia di amministrazione. La definizione dei metadati avviene mediante la creazione di categorie documentali, le quali sono utilizzate dall'applicazione per identificare la tipologia di un file dal punto di vista delle informazioni contenute in esso. Un esempio di categoria documentale e di relativi metadati è ad esempio:

- cliente: data di creazione, nome del cliente

dove “cliente” è il nome della categoria documentale, mentre “data di creazione” e “nome del cliente” sono i relativi metadati.

La presenza di categorie documentali, facilità le attività di ricerca e di consultazione dei documenti; infatti, è possibile ricercare un documento inserendo come criterio di ricerca un metadato definito in una certa categoria documentale. La consultazione è invece facilitata dall'interfaccia di HotStudio, la quale provvede in automatico alla distinzione per categoria attraverso una comoda visione ad albero.

Tra le altre funzionalità significative offerte da HotStudio troviamo la possibilità di creare utenti e relativi diritti d'accesso al repository. Inoltre, è prevista la possibilità di inoltrare documenti da un utente all'altro.

Per gestire le informazioni di archiviazione, le impostazioni e gli altri dati, HotStudio si affida ad un database esterno; i DBMS supportati sono Microsoft SQL Server e Firebird (il più usato e consigliato).

Per concludere voglio citare le funzionalità di integrazione con Microsoft Office (previste anche da Alfresco) e la capacità di HotStudio di interfacciarsi con il fax server prodotto da TLogic (questo connubio permette una gestione avanzata e automatizzata dei fax).

Poter utilizzare Alfresco per rimpiazzare l'attuale repository, significa capire se e in che modo Alfresco può garantire tutte le attuali funzionalità di HotStudio.

4.4 Le API di Alfresco

L'accesso al repository mediante API (Application Programming Interface) è molto utile nel caso si voglia accedere ad Alfresco in modo alternativo alla normale interfaccia web.

Le API di Alfresco sono classificabili in 3 macro-categorie:

- Remote API
- Scripting API
- Java API

Le “Remote API” sono delle interfacce di programmazione che sfruttano il protocollo HTTP per inviare e ricevere dati; in particolare viene usata la porta 80, la quale normalmente non viene bloccata da alcun firewall: è così possibile aggirare qualsiasi meccanismo di protezione.

Le API di tipo “Scripting” sono utili per le applicazioni web che vogliono accedere ad Alfresco mediante JavaScript o PHP.

Per finire le “Java API” permettono un accesso completo alle funzionalità del repository mediante aggancio diretto al codice sorgente. Queste API fanno uso della tecnologia RMI (Remote Method Invocation) la quale risulta molto performante.

Le necessità di TLogic hanno portato alla scelta obbligata di adottare la prima tipologia di API: “Remote API”.

Queste librerie permettono di comunicare con Alfresco in due modi simili ma distinti:

- mediante Web Service
- mediante Web Script

La prima modalità è forse la più usata, e consiste nello scambio di informazioni mediante la trasmissione di messaggi; tali messaggi sono generalmente inclusi in una sorta di “busta” (la più famosa è SOAP) e sono trasportati mediante protocollo

HTTP in formato XML.

La seconda modalità invece, è una variante della prima; questo metodo sfrutta in modo più efficiente e completo le funzionalità del protocollo HTTP, riducendo notevolmente overhead di comunicazione creato dai normali web service. I Web Script implementano il principio chiamato RESTFul il quale permette di usare i metodi PUT, GET, POST e DELETE per ottenere risultati migliori in termini di performance.

Quest'ultima tecnologia, seppur ottima, non è ancora pienamente supportata dai vari linguaggi di programmazione e si adatta principalmente all'uso in applicazione di tipo web. Per questi motivi, le attività di analisi e di test eseguiti in TLogic sono state condotte mediante l'uso dei normali Web Service.

4.4.1 I Web Service di Alfresco

Come indicato nel wiki della comunità di Alfresco [8], i Web Service disponibili per la comunicazione con Alfresco sono i seguenti:

Nome	Descrizione
Authentication	Login e logout
Repository	Query e manipolazione del repository
Content	Manipolazione dei contenuti
Authoring	Gestione del versionamento dei contenuti
Classification	Gestione categorie e classificazioni
Access Control	Gestione dei diritti di accesso (ACL)
Action	Gestione delle azioni e dei ruoli
Administration	Operazioni di amministrazione
Dictionary	Descrizione dei modelli

Tabella2: Web Service di Alfresco

Ogni servizio mette a disposizione un certo numero di metodi, i quali possono essere invocati da qualsiasi client.

L'aggancio ai web service di Alfresco può avvenire attraverso la maggior parte dei linguaggi di programmazione moderni come: Java, C#, Delphi, C++, ecc.

Ogni web service è definito in un file `.wsdl` (Web Services Description Language); questi file, nel caso di Alfresco sono disponibili all'indirizzo: http://NomeServerAlfresco/wsdl/nomeServizio.wsdl

Ambienti di sviluppo come NetBeans o Eclipse permettono l'importazione automatica di tali file. Il processo di importazione permette la creazione dei file sorgente contenenti la definizione dei web service e dei relativi metodi. L'invocazione di un metodo di un web service, si riduce quindi alla semplice istanziazione di un oggetto (il web service) sul quale verrà poi invocato il metodo desiderato.

Nel ambito di TLogic, i test sono stati effettuati utilizzando Java come linguaggio di programmazione e NetBeans come ambiente di sviluppo.

Data la notevole quantità di servizi offerti, durante il periodo di stage, sono stati provati solo i web service più significativi per un eventuale uso in HotStudio.

I prossimi capitoli spiegheranno più dettagliatamente le prove effettuate analizzando uno per uno ogni web service provato.

4.4.1.1 Note introduttive

In questo capito sono spiegati alcuni concetti che stanno alla base della manipolazione del repository di Alfresco.

Store

Secondo la notazione di Alfresco, un repository è costituito da uno o più store. Ogni store rappresenta un'unità di archiviazione. Di default Alfresco crea i seguenti store:

Riferimento
archive://SpacesStore
system://system
user://alfrescoUserStores
workspace://lightWeightVersionStore
workspace://SpacesStore
workspace://version2Store
avm://sitestore

Tabella3: Store di default

Come si può notare il riferimento ad uno `Store` è composto da due parti separate da "://"; la prima parte indica lo schema, mentre la seconda parte indica il nome dello store. Lo store generalmente usato per l'archiviazione dei file è "workspace://SpacesStore".

Reference:

Ogni store si compone al suo interno di uno o più nodi i quali nodi possono rappresentare un file o una cartella. Questi nodi sono identificati da un codice univoco (uuid), il quale può essere usato per riferirsi ad un nodo. Ad ogni nodo sono associate varie informazioni tra cui:

- `Properties`: proprietà
- `Aspect`: concetto simile alle categorie documentali di HotStudio
- `Permission`: permessi di accesso (ACL)
- `Children`: nodi figli
- `Associations`: particolari associazioni con altri nodi
- `Parents`: nodi padre

Ogni operazione su un file o su una cartella deve fare riferimento a uno o più `Reference`. Un oggetto di tipo `Reference` è caratterizzato da: uno store, un uuid (opzionale) e una path (opzionale).

Parent Reference:

Simile ad un `Reference`, il `ParentReference` punta al padre di un nodo.

Questo tipo di riferimento viene generalmente utilizzato per la creazione di nuovi nodi quali figli del nodo padre riferito dall'oggetto `ParentReference`.

Predicate

Determinati metodi di Alfresco permettono di eseguire un'operazione a più nodi per volta. L'oggetto da utilizzare per indicare un insieme di nodi, è `Predicate`. `Predicate` è costituito da: un array di `Reference` composto da almeno un elemento, uno store e una query.

4.4.1.2 Authentication service

E' il web service che permette l'autenticazione di un utente con Alfresco.

I due metodi da esso offerti sono:

- `AuthenticationResult startSession(String userName, String password)`
- `void endSession(String ticket)`

Il primo, come dice il nome, permette ad un utente di autenticarsi mediante nome utente e password. Il risultato dell'autenticazione è un oggetto di tipo "`AuthenticationResult`" all'interno del quale sono contenuti lo userName dell'utente e un ticket ritornato dal server.

Il secondo metodo, permette il logout di un utente mediante passaggio del ticket di autenticazione.

Il ticket di autenticazione è univoco e permette di identificare un utente (e la relativa sessione) fino al momento della disconnessione. Tale ticket permette all'utente

possessore, l'utilizzo degli altri web service; a livello implementativo, il ticket viene passato al server ogni qual volta l'utente effettua una chiamata a un web service. L'inoltro del ticket viene effettuato in modo implicito mediante la definizione di un client Axis locale.

4.4.1.3 Repository service

I metodi più importanti messi a disposizione da questo servizio sono:

- `Store createStore(String scheme, String address)`
- `Store[ ] getStores()`
- `Node[ ] get(Predicate[ ] where)`
- `UpdateResult update(CML statements)`

I primi due metodi permettono rispettivamente la creazione di un nuovo `Store` e la selezione di tutti gli “Store” presenti nel repository.

Il metodo `get`, permette la selezione dei nodi specificati dall'array di `Predicate` passato come parametro. Un oggetto di tipo `Node`, contiene varie informazioni come il tipo del nodo, le eventuali proprietà, un oggetto di tipo `Reference` che punta al nodo, ecc.

Il metodo `update` infine, è uno dei metodi più potenti tra tutti quelli offerti dai web service di Alfresco. Questo metodo permette la manipolazione dei contenuti del repository. La sua potenza sta nella possibilità di eseguire comandi descritti mediante un linguaggio definito dal team di Alfresco: CML (Content Manipulation Language). Con CML è possibile:

- creare un nuovo nodo
- aggiungere un aspect ad un nodo
- aggiornare un nodo
- cancellare un nodo
- muovere un nodo da una posizione ad un'altra

- copiare un nodo da una posizione ad un'altra
- aggiungere un figlio ad un nodo
- cancellare il figlio di un nodo
- creare un'associazione
- rimuovere un'associazione

Di seguito è riportato un esempio per la cancellazione di uno o più nodi di tipo file (il linguaggio di programmazione usato è Java):

```
public static UpdateResult[] cancellaNodo(Predicate where)
        throws RepositoryFault, RemoteException
{
    CML cml = new CML();
    CMLDelete delete = new CMLDelete(where);
    cml.setDelete(new CMLDelete[]{delete});

    UpdateResult[] results = repositoryService.update(cml);

    return results;
}
```

Figura4: funzione per la cancellazione di un o più nodi dal repository

L'esempio cancella il/i nodi identificati dal parametro `where`. L'oggetto `cml` definisce il tipo di manipolazione da eseguire (nell'esempio la cancellazione di un nodo); l'oggetto `delete` invece contiene il comando di cancellazione, il quale viene passato all'oggetto `cml`. Il metodo `update` viene poi invocato sull'oggetto `repositoryService` che corrisponde ad una istanza del web service `Repository`. Il risultato dell'operazione viene ritornato come un oggetto di tipo `UpdateResult`.

4.4.1.4 Content service

E' il servizio che permette di creare, leggere e cancellare contenuti dal repository. Creare un contenuto, significa creare un nuovo file nel repository; di tale file verranno specificate le proprietà, gli aspects, la posizione nel repository, ecc.

I metodi che permettono la manipolazione dei contenuti sono:

- `Content[ ] read(Predicate items, String property)`
- `Content write(Reference node, String Property, base64Binary content, ContentFormat format)`
- `Content[ ] clear(Predicate items, String property)`

Come si capisce dai nomi dei metodi e dai parametri passati, il metodo `read` permette la lettura di un determinato numero di contenuti, relativamente ai nodi selezionati dal parametro `items`; `write` permette la scrittura del contenuto `content` nel nodo `node`. Il formato del contenuto è specificato dal parametro `format` (es. “application/pdf”). Il metodo `clear`, infine, cancella il contenuto dei nodi selezionati dal predicato `items`.

4.4.1.5 Altri

I restanti web service garantiscono l'accesso a tutte le altre funzionalità di Alfresco. Tali funzionalità non sono state considerate nell'analisi e nei test di Alfresco, dato che si è preferito, concentrare l'attenzione sugli aspetti che più si avvicinano alle caratteristiche di HotStudio. Nel sito ufficiale della comunità di Alfresco è comunque presente una guida per gli sviluppatori, la quale può essere utilizzata per approfondire alcune tematiche relative ai web service.

4.5 I test

Una volta apprese le potenzialità e le modalità di funzionamento dei web service, si è deciso di provare materialmente il prodotto, mediante una applicazione di test.

Essendo un'applicazione di prova, essa non è stata dotata di interfaccia grafica e tanto meno di funzionalità di interazione con l'utilizzatore.

Lo sviluppo di tale applicazione, è stato condotto in NetBeans (IDE sviluppato e distribuito da Sun Microsystems [4]), il quale si è dimostrato più volte un utile strumento con cui sviluppare applicazioni Java.

NetBeans si integra molto bene per lo sviluppo di applicazioni che fanno uso di Web Service; una procedura di importazione guidata, accompagna lo sviluppatore nel processo di aggancio ai web service. Questo processo di importazione, crea alcuni file sorgenti i quali definiscono le classi e i metodi per l'interrogazione dei servizi remoti.

Come anticipato, un web service può essere utilizzato, se e solo se il client dispone di un ticket di autenticazione valido per Alfresco; tale ticket, deve essere trasmesso per ogni singola richiesta al server, in modo tale che Alfresco abbia la certezza sull'identità dell'utilizzatore. La trasmissione del ticket avviene in modo implicito attraverso l'uso di client Axis.

L'ambiente di sviluppo NetBeans non prevede la possibilità di importare web service attraverso un supporto Axis. Per questo motivo si è ritenuto necessario l'utilizzo di una parte del SDK (Software Development Kit) di Alfresco. Tale libreria è disponibile presso il sito della comunità di sviluppo ed è interamente scritta in Java.

Confrontando i sorgenti prodotti da NetBeans con quelli contenuti nel SDK, ho potuto notare che l'unica principale differenza è la classe da cui derivano tutte le altre classi. Nel caso di NetBeans si tratta di "`javax.xml.ws.Service`" mentre nel caso del SDK si tratta di "`org.apache.axis.client.Service`".

I package importati dal SDK di Alfresco sono stati:

- `org.alfresco.webservice.accesscontrol`
- `org.alfresco.webservice.action`
- `org.alfresco.webservice.administration`
- `org.alfresco.webservice.authentication`
- `org.alfresco.webservice.authoring`
- `org.alfresco.webservice.classification`
- `org.alfresco.webservice.content`

- `org.alfresco.webservice.dictionary`
- `org.alfresco.webservice.repository`
- `org.alfresco.webservice.types`
- `org.alfresco.webservice.util`

Come si può facilmente notare, esiste una corrispondenza biunivoca tra i package generati e i web service di Alfresco, ad eccezione degli ultimi due package i quali contengono le classi dei tipi (es. Reference, Predicate, ecc.) e classi di tipo utility.

Le classi che compongono il programma di test, sono state invece mantenute nel package di default visto il loro numero limitato.

Il programma di prova, si compone di cinque classi (più quelle relative ai web service).

Il seguente diagramma delle classi mette in evidenza le classi realizzate per l'applicazione, con i relativi campi e metodi.

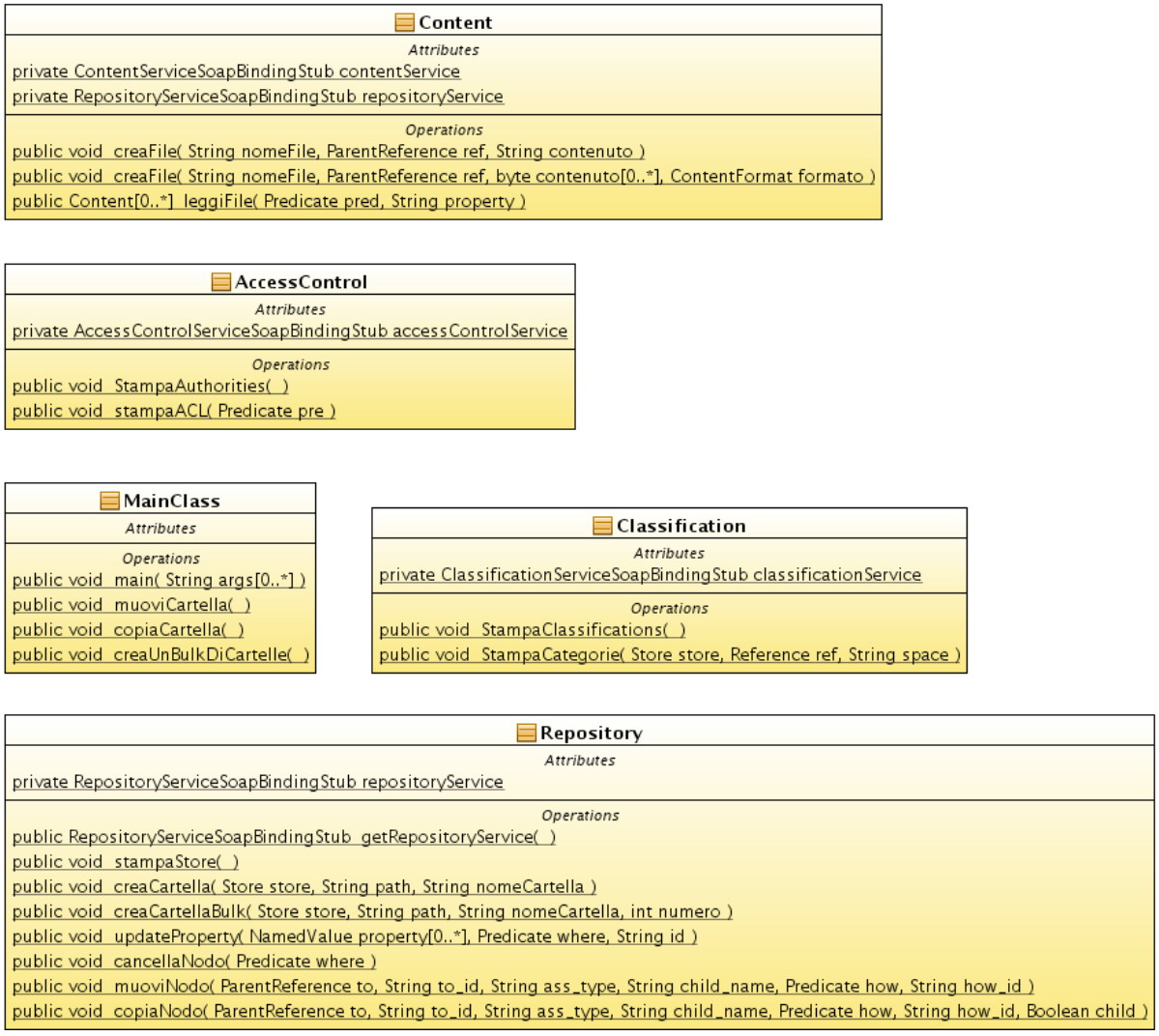

Figura5: digramma delle classi dell'applicazione di test

Come si evince dalla figura5, lo sviluppo dell'applicazione ha comportato la creazione di quattro classi d'appoggio, nominate con il nome del web service corrispondente e di una classe (`MainClass`) che ospita il metodo principale dell'applicazione (`main`).

Le quattro classi d'appoggio (`Content`, `AccessControl`, `Classification` e `Repository`) forniscono dei metodi ad-hoc per la manipolazione del repository. Ciascuna di queste classi, contiene al proprio interno, un campo privato che funge da riferimento al web service utilizzato dalla classe. Il tipo di questo campo statico, varia

a seconda della classe: per la classe `Repository`, ad esempio, il tipo del campo è `RepositoryServiceSoapBindingStub` il quale tipo corrisponde ad una delle classi generate in modo automatico dall'importazione del web service `Repository`; Questi campi sono inizializzati attraverso un blocco di inizializzazione statico (il quale viene eseguito al caricamento della classe nel momento in cui essa viene utilizzata per la prima volta).

Nei prossimi capitoli sono analizzate con più dettaglio le cinque classi sopra.

4.5.1 MainClass

E' la classe principale. I metodi da essa definiti sono elencati nella figura5.

Il metodo `main(String[ ] args)` esegue una serie di operazioni di test sul repository. I restanti metodi sono stati creati per agevolare alcune operazioni ricorrenti:

- `copiaCartella()` : copia la cartella da un nodo ad un altro
- `muoviCartella()` : taglia e incolla una cartella da un nodo ad un altro
- `creaUnBulkDiCartelle()` : crea un certo numero di cartelle (sono previste due differenti modalità di creazione di cartelle; per approfondimenti vedere il capitolo 4.6)

Il metodo `main(String[ ] args)`, inizia autenticando l'utente per mezzo del metodo statico `AuthenticationUtils.startSession(userName, password)`. La classe `AuthenticationUtils` è contenuta nel package `org.alfresco.webservice.util`. In caso di errori di autenticazione viene lanciata un eccezione `AuthenticationFault`.

Successivamente all'autenticazione, vengono create delle cartelle e dei file di prova, ai quali sono assegnate proprietà e contenuti casuali. Vengono poi eseguite operazioni di copia, spostamento, modifica di file, cancellazione di nodi e stampa di informazioni varie.

Per modificare le modalità di test, occorre modificare la successione e la tipologia di

metodi invocati dal metodo `main`.

4.5.2 AccessControl

Personalizza alcuni metodi del servizio “Access Controll”.

Oltre al riferimento verso il web service corrispondente, la classe in oggetto definisce i metodi:

- `StampaAuthorities()` : stampa le authorities (gruppi e ruoli) dell'utente corrente.
- `stampaACL(Predicate pre)` : stampa le ACE (Access Controll Entry) dei nodi identificati dal predicato `pre`.

Per motivi di tempo non è stato possibile approfondire gli altri metodi messi a disposizione dal servizio remoto “AccessControll”.

4.5.3 Classification

Questa classe è stata creata per personalizzare e provare alcuni metodi per la visualizzazione delle categorie documentali definite in Alfresco. La classe si appoggia come dice il nome al web service “Classification”.

L'unico metodo implementato è stato:

- `StampaClassifications()` : stampa tutte le categoria documentali presenti nel repository di Alfresco. La stampa viene fatta a livello testuale mediante una visualizzazione gerarchica (utile per rendere noto il livello di annidamento di ciascuna categoria).

4.5.4 Content

Per motivi di efficienza, questa classe è stata dotata di due riferimenti a web service: il primo punta al servizio che caratterizza la classe (`Content`); il secondo fa riferimento al servizio remoto “Repository”.

Entrambi i riferimenti, come per tutte le altre classi create, sono stati definiti con visibilità privata e la loro inizializzazione è stata affidata ad un blocco di

inizializzazione statica.

I metodi definiti in questa classe permettono la creazione di file e la lettura dei loro contenuti.

Nel dettaglio i metodi creati sono stati:

- `creaFile(String nomeFile, ParentReference ref, String contenuto)` : crea un nuovo file di nome `nomeFile` il cui nodo padre è `ref` e il cui contenuto è definito dal parametro `contenuto`.
- `creaFile(String nomeFile, ParentReference ref, byte[] contenuto, ContentFormat format)` : overloading del metodo precedente; si differenzia per la possibilità di poter specificare il formato del file da creare. Nel caso precedente, il formato è di default “text/plain”.
- `leggiFile(Predicate pred, String property)` : legge il contenuto della proprietà `property` del/dei file selezionati dal predicato `pred`.

Alcune precisazioni sulla gestione delle proprietà dei file: Alfresco memorizza ogni caratteristica dei file sotto forma di proprietà. Il contenuto di un file, ad esempio, è mantenuto nella proprietà `content`, mentre il nome del file è memorizzato nella proprietà `name`.

4.5.5 Repository

E' la classe più ricca di metodi tra tutte quelle create. Essa fa uso del web service “Repository” per la manipolazione dei nodi (file o cartelle) presenti nel repository di Alfresco.

I metodi definiti in questa classe sono:

- `stampaStore()` : stampa la lista degli store presenti nel repository
- `creaCartella(Store store, String path, String nomeCartella)` : crea una cartella con nome uguale al parametro `nomeCartella`, e collocata nello store passato come parametro, nella

posizione definita da `path`.

- `creaCartellaBulk(Store store, String path, String nomeCartella, int numero)` : come il metodo precedente, con la differenza che il numero di cartelle create è uguale al parametro intero passato. Il nome delle cartelle create è uguale al parametro `nomeCartella` al quale viene concatenato il numero progressivo di creazione.
- `updateProperty(NamedValue[ ] property, Predicate where, String id)` : aggiorna la/le proprietà del nodo identificato dal predicato e dall'id passati come parametri.
- `cancellaNodo(Predicate where)` : cancella il/i nodi identificati dal predicato `where`.
- `muoviNodo(ParentReference to, String to_id, String ass_type, String child_name, Predicate how, String how_id)` : muove un nodo dalla posizione identificata dai parametri `how` e `how_id` alla posizione definita dal parametro `to`.
- `copiaNodo(ParentReference to, String to_id, String ass_type, String child_name, Predicate how, String how_id, Boolean child)` : copia un nodo dalla posizione identificata dai parametri `how` e `how_id` alla posizione definita dal parametro `to`. Il parametro `child`, indica se la copia deve essere ricorsiva o meno (nei confronti dei figli).

4.6 Risultati dei test

L'esecuzione dall'applicazione di test, ha permesso di ricavare dati importanti sulle modalità d'uso di Alfresco e sulle sue prestazioni.

Dalle prove effettuate sono sorti particolari dettagli sul funzionamento di Alfresco. Il più importante tra questi, è stato senza dubbio la possibilità di eseguire ottimizzazioni a livello di codice, per la manipolazione del repository.

Come già detto, le operazioni che permettono di modificare lo stato del repository, si affidano ad un linguaggio di manipolazione chiamato CML. Un oggetto di tipo CML, può contenere al proprio interno uno o più comandi di manipolazione. Tali comandi non devono necessariamente essere dello stesso tipo (cancellazione, creazione, ecc.) ma possono riguardare più azioni.

Uno stesso oggetto CML può ad esempio creare, cancellare, spostare, modificare, ecc. file.

Per verificare le diversità in termini prestazionali, l'applicazione di prova è stata configurata in modo da creare duemila cartelle con la modalità non ottimizzata (creazione di una cartella per volta) e duemila cartelle con la modalità ottimizzata (mediante la creazione di un oggetto CML contenente duemila comandi di creazione). I risultati ottenuti sono riportati nella tabella4.

Descrizione	Non ottimizzato	Ottimizzato
Tempo impiegato	381 sec	315 sec
Utilizzo CPU medio lato server	53,30%	81,56%
Utilizzo rete (upload totale)	5321 KB	2335 KB
Utilizzo rete (download totale)	212 KB	156 KB
Utilizzo rete (upload medio)	13,97 KB/s	7,41 KB/s
Utilizzo rete (download medio)	0,56 KB/s	0,49 KB/s

Tabella4: Risultati del test per la creazione di 2000 cartelle. I dati rappresentano la media dei risultati ottenuti in 20 prove successive. Le celle in verde evidenziano un minore uso di risorse, mentre le celle in rosso indicano un uso maggiore delle risorse.

La tabella mostra con chiarezza la differenza prestazionale delle due modalità.

Mediante l'ottimizzazione del codice, l'utilizzo delle risorse è notevolmente ridotto, così come anche i tempi di esecuzione.

Un'eventuale adozione di Alfresco da parte di TLogic, dovrà tener conto di questo

fatto importante. L'utilizzo del programma da parte di più utenti sarà certamente condizionato dalle prestazioni del sistema, le quali potranno essere aumentate grazie a questa tecnica ottimizzante.

4.7 Possibile strategia d'uso in azienda

Sia nei test, sia nella carta, Alfresco si è dimostrato un valido prodotto, tanto da pensarne un possibile uso in azienda.

Adottare Alfresco come repository documentale per HotStudio, potrebbe portare grandi migliorie e grande pregio al prodotto di TLogic.

L'idea di Tlogic però, va ben oltre l'integrazione in HotStudio; la software house, sta da tempo pensando di creare un framework di aggancio per Alfresco. Tale framework potrebbe essere utilizzato da una molteplicità di applicazioni (quindi non solo da HotStudio) per interagire con il repository.

Lo schema seguente, da un idea di quello che potrebbe essere un possibile uso di Alfresco nella logica di TLogic:

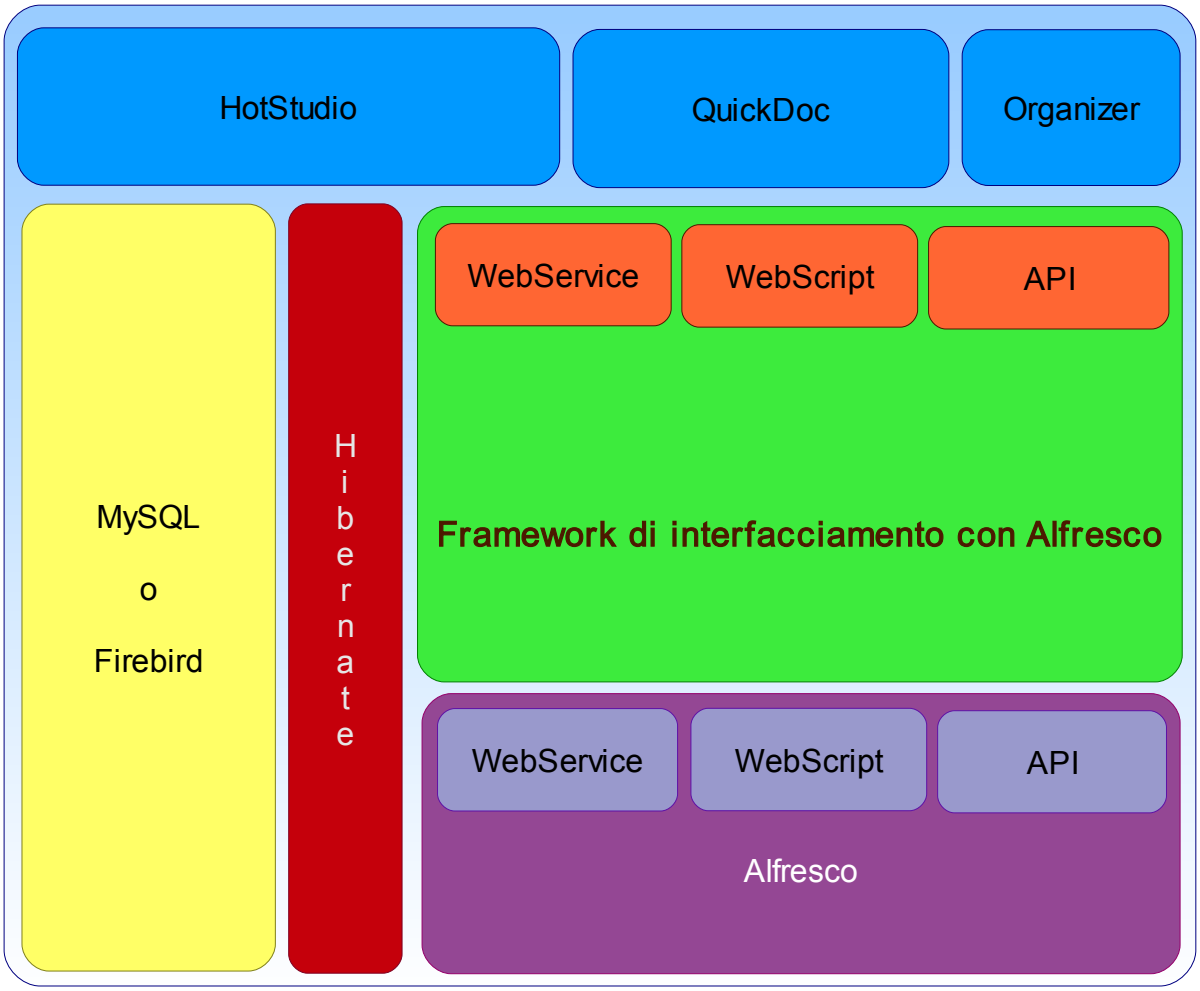

Figura6: possibile architettura TLogic

4.8 Conclusioni

Lo stage è stato sicuramente d'aiuto a TLogic, la quale sta valutando in questo periodo le possibili strade per investire in modo produttivo ed efficiente su Alfresco.

Attualmente Alfresco è in pieno sviluppo, e molte società (anche importanti) stanno investendo su questo prodotto. Per TLogic il futuro è quasi sicuramente basato su Alfresco.

5 Conclusioni sull'attività di stage

L'esperienza di stage effettuata presso EniData è stata sicuramente un'esperienza importante.

L'aspetto che più di altri è stato risaltato dallo stage, è la difficoltà di gestione delle informazioni a livello aziendale. Le informazioni sono un bene prezioso per un'azienda, e per questo motivo devono essere gestite in maniera corretta ed efficiente, garantendo la disponibilità nel tempo, la facilità d'accesso e il controllo globale.

Il concetto di informazione, è un concetto molto ampio, che non si limita ai soli documenti cartacei e/o digitali, ma comprende molti altri aspetti di importanza non inferiore.

Lo stage ha permesso ad EniData, E-Masterit e TLogic, di progredire sotto questo aspetto. I vantaggi in termini di produttività ed efficienza saranno sicuramente evidenti sia nel breve che nel medio/lungo periodo.

Senza dubbio le attività svolte nello stage, non sono che l'inizio (o forse la continuazione) di una serie di processi di innovazione ed ammodernamento, i quali potranno (se applicati correttamente all'attività aziendale) portare frutti significativi.

Il livello di soddisfazione dimostrato dalle aziende è stato elevato, segno che il lavoro svolto è stato utile ed apprezzato. Gli strumenti creati sono tutt'ora utilizzati in modo attivo nella realtà aziendale.

5.1 Conoscenze possedute e acquisite

Le conoscenze possedute ad inizio stage, hanno sicuramente contribuito al conseguimento del risultato finale. Le competenze in ambito Java e riguardo i sistemi operativi, hanno permesso di affrontare i problemi incontrati in maniera sicura e veloce, senza necessitare particolari attività di formazione.

L'elevata disponibilità di documentazione nel web, ha permesso di chiarire ed approfondire alcuni aspetti dei prodotti utilizzati.

Di enorme aiuto è stato il wiki di Alfresco il quale, seppur carente in qualche aspetto, ha garantito lo studio e l'usabilità del prodotto.

Tra le conoscenze acquisite nello stage, voglio menzionare:

- il concetto di web service (creazione, uso e utilità)
- il concetto di software di gestione documentale (tipologie, utilizzi, vantaggi e svantaggi)
- il concetto di repository documentale
- la conoscenza basilare del linguaggio di programmazione Delphi

5.2 Piano di lavoro

Il piano di lavoro definito inizialmente è stato rispettato al 70%. Le motivazioni che hanno comportato la modifica o la cancellazione di alcune attività, sono state principalmente:

- mutamento dello scenario aziendale: alcuni aspetti dell'azienda sono cambiati (es. assunzione di nuovo personale, mutamento delle esigenze del mercato, scoperta di nuove tecnologie, ecc.) comportando l'annullamento o la modifica di alcune attività previste dal piano di lavoro
- prolungamento di alcune attività per permettere l'approfondimento di alcune tematiche importanti ai fini dello stage: in accordo con l'azienda si è scelto di prolungare alcune attività di analisi, per permettere l'ottenimento di risultati più accurati, utili all'azienda ma anche allo stagista
- sottostima del tempo per alcune attività: certe attività hanno richiesto più tempo del previsto.

La principale causa che ha comportato l'allungamento dei tempi di svolgimento di una o più attività, è stata la necessità di documentarsi in modo più approfondito e rigoroso su aspetti come: funzionalità avanzate dei sistemi di versionamento, le possibilità offerte agli sviluppatori di Alfresco, ecc.

Il diagramma di Gantt della figura7, da un'idea della disposizione temporale di

ciascuna attività. E' doveroso precisare, che tale diagramma illustra una visione di massima della realtà: alcune attività sono state riprese nel tempo per periodi limitati (dell'ordine di alcune ore).

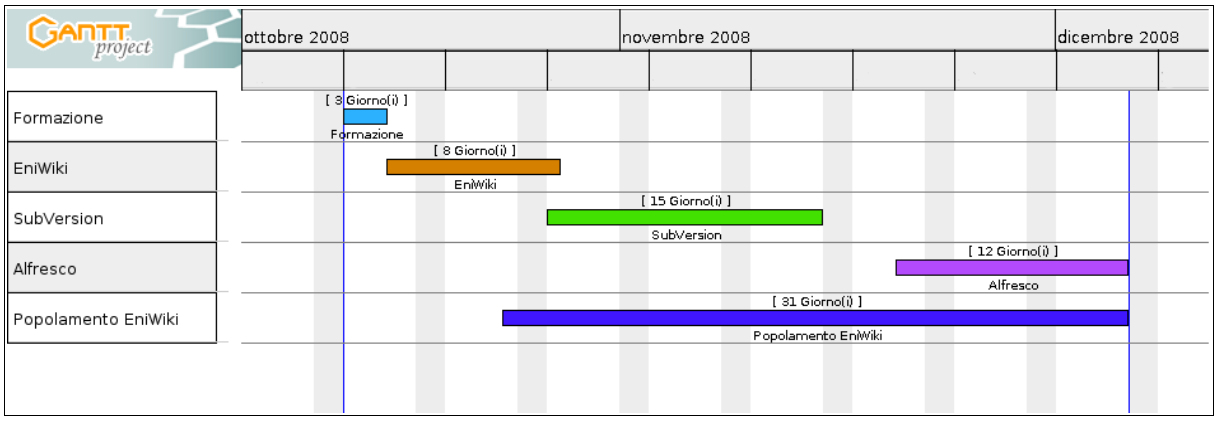

Figura7: diagramma di Gantt; attività stage.

6 Glossario

API: (Application Programming Interface) sono un insieme di procedure messe a disposizione del programmatore.

Data-entry: attività di immissione di dati relativi ad un certo contesto.

DBMS: (DataBase Management System) è un sistema software progettato per permettere la creazione e la manipolazione di database.

Firewall: (in italiano si traduce “muro di fuoco”) è un componente di difesa per una rete informatica.

ICR: (Intelligent Character Recognition) riconoscimento intelligente dei caratteri; permette la conversione di un'immagine contenente testo in testo modificabile. Vedere anche OCR.

Metadato: (letteralmente “dato su un altro dato”) è l'informazione che descrive un altro dato.

Mirroring: attività che mirano a mantenere copie esattamente uguali di uno o più dischi rigidi. Corrisponde al concetto di RAID 1.

NAS: (Network Attached Storage) dispositivo collegato ad una rete di computer, il cui compito è quello di permettere la condivisione di file tra gli utenti della rete.

OCR: (Optical Character Recognition) tipologia di programmi che permettono di convertire in testo digitale, immagini contenenti testo.

Repository: la parte di un sistema informativo in cui vengono memorizzati e gestiti i metadati.

VPN: (Virtual Private Network) è una rete privata instaurata tra due soggetti che comunicano attraverso un mezzo pubblico (come ad esempio internet).

Workflow: schema che promuove il lavoro di gruppo secondo la definizione di un modello che prende il nome di “workflow model”.

7 Riferimenti

[1] Sito ufficiale MediaWiki: http://www.mediawiki.org

[2] Wikipedia: http://it.wikipedia.org

[3] Sito ufficiale VMware: http://www.vmware.com/

[4] Sito ufficiale Sun Microsystems: http://it.sun.com/

[5] Sito ufficiale Tigris: http://subversion.tigris.org/

[6] Sito ufficiale CollabNet: http://www.collab.net/

[7] Sito ufficiale Alfresco: http://www.alfresco.com/

[8] Wiki Alfresco: http://wiki.alfresco.com

[9] Sito ufficiale NetBeans: http://www.netbeans.org/

[10] Libro “Alfresco Enterprise Content Management Implementation” di “Munwar Shariff”: http://www.packtpub.com/alfresco/book

[11] Sito Ufficiale Wikka: http://wikkawiki.org