Alma Mater Studiorum · Università di Bologna

FACOLTÀ DI SCIENZE MATEMATICHE, FISICHE E NATURALI
Corso di Laurea Specialistica in Informatica

Progettazione e Realizzazione di un sistema web per l'automazione dei processi di gestione delle risorse umane in Dallara Automobili

Tesi di Laurea in Sistemi ed Applicazioni Multimediali

Relatore:
Chiar.mo Prof.
MARCO ROCCETTI

Presentata da:
DANIELE BOTTILLO
Matricola 0000310588

Correlatore:
Chiar.mo Dott.
FABRIZIO ARBUCCI

Sessione II
Anno Accademico 2009/2010

Indice

Capitolo 1

Introduzione

Oggigiorno l'uso dei sistemi informativi pervade la quotidianetà delle persone, dal lavoro al divertimento, dalle connessioni sociali alla cultura, quasi ogni ambito della vita di ogni persona è legata ad un qualche tipo di sistema informativo che ne permette la gestione.
Sistemi informativi e tecnologie informatiche sono da molti anni ormai un connubio solido, nato dalla semplicità con cui le tecnologie informatiche permettono di gestire carichi di lavoro. Queste tecnologie infatti permettono facilmente di automatizzare tali processi di gestione, con la conseguenza di facilitare il lavoro e permettere di sostenere un alto numero di 'elementi' che altrimenti manualmente sarebbe impossibile da trattare.
Nel mondo del lavoro, tali sistemi sono ampiamente utilizzati: si parte dal software per la ricerca di lavoro, al software per la gestione del personale, al programma di gestione delle risorse (sia umane che materiali), ai programmi per l'uso di macchine automatizzate e molti altri ancora.
Lo scopo di tali servizi è, come spesso capita agli strumenti informatici, di velocizzare i processi di sviluppo, di migliorarne la qualità oltre che di minimizzarne gli errori e di tenere traccia di statistiche ed andamenti. Inoltre tali vantaggi sono amplificati dal numero di 'elementi' che vengono gestiti da questi sistemi: per esempio sostenere il carico di poche decine di ordini in un magazzino cambia radicalmente la necessità rispetto ad una gestione nel-

l'ordine delle centinaia. Viceversa, nel caso in cui si progetti e si utilizzi un sistema informatico sovradimensionando il problema, molto probabilmente si ottiene esattamente il risultato contrario, cioè un sistema che complica la gestione dei dati, allunga i tempi e aumenta lo sforzo delle persone che lo utilizzano.

Per questo è giusto progettare ad hoc il sistema per l'esigenze degli utilizzatori: l'insieme delle funzioni disponibili del sistema devono rispecchiare i reali bisogni e le reali necessità, non è necessario sovraccaricare il sistema di funzionalità ma semmai lasciare la possibilità al sistema di evolvere nel caso in cui le necessità aumentino; altro elemento chiave è l'interfaccia del sistema, che deve essere adatta alla tipologia di utilizzatori (eg. una segretaria ed un gestore delle risorse umane hanno conoscenze e atteggiamenti diversi rispetto l'uso del software), solitamente la più semplice possibile e adatta al sistema che deve supportare, per esempio un sistema di ricezione dei dati deve favorire tale meccanismo, aiutare l'utente durante l'inserimento, velocizzare alcune operazioni e predisporre alcuni strumenti di supporto (eg. copia veloce dei dati, inserimento date attraverso calendari, etc..). In definitiva un sistema informatico deve essere una giusta sintesi tra funzionalità, interfaccia e usabilità.

Per quanto riguarda questo lavoro di tesi, quindi verrà esposto un problema reale relativo ad un'azienda italiana e verrà descritta la soluzione che ha portato alla creazione di un sistema informatico specializzato.

1.1 Dallara Automobili

La Dallara Automobili è un' azienda italiana costruttice di automobili da competizione, fondata nel 1972 a Varano de' Melegari dall' ing. Gian Paolo Dallara.
Le attività principali della Dallara Automobili riguardano le competizioni sportive a ruote scoperte, dalle formule minori fino alla Formula 1 in cui ha gareggiato dalla stagione 1988 a quella del 1992, quale fornitrice dei telai della Scuderia Italia. Proprio quest'anno la Dallara Automobili ha fatto il suo ritorno in Formula Uno gareggiando nelle prime sei gare con il team HRT Racing.
La grandezza e la continua espansione dell'azienda, hanno portato ad una nuova esigenza nel settore delle risorse umane: gestire l'intera pratica delle candidature, dal primo contatto alla vera e propria assunzione. Non essendo presente nessun sistema informatizzato all'interno dell'azienda, è nata così l'esigenza di ricercare un nuovo software.
La mole di dati è abbastanza consistente, si parla di migliaia di candidature già ricevute, in formato cartaceo ed un flusso costante di candidature giornaliere sull'ordine delle decine.
La stessa Dallara Automobili ha imposto diversi vincoli nella definizione del problema e quindi della sua risoluzione: non utilizzare software proprietario ma open source e cercare di utilizzare o di creare ad hoc uno strumento il più possibile modellato e adatto a tale esigenza.
Sono stati affrontati diversi temi su come impostare ed organizzare questa gestione, dalla sicurezza di esporre i dati sensibili sul web all'installare un'architettura su più livelli per permettere ad altri software di accedere ai dati raccolti; sono stati inoltre trattati a lungo i meccanismi di inserimento per facilitare chi deve presentarsi all'azienda, in modo da mirare direttamente ai dati più importanti per l'azienda e accelerare il processo di reclutamente di nuove figure professionali.

1.2 ERP

L'acronimo ERP significa *Enterprise Resource Planning* (letteralmente 'pianificazione delle risorse d'impresa').
Si tratta di un sistema di gestione, quindi un sistema informativo, che integra tutti i processi di business rilevanti di un'azienda: vendite, acquisti, gestione magazzino, contabilità come funzionalità principali. Con l'aumento della popolarità dell'ERP e la riduzione dei costi per l'ICT (Information and Communication Technology), si sono sviluppate applicazioni che aiutano i business manager ad implementare questa metodologia nelle attività di business come: controllo di inventari, tracciamento degli ordini, servizi per i clienti, finanza e risorse umane.
La prima versione dell'ERP metteva in collegamento diretto le aree di gestione contabile con l'area di gestione logistica come magazzini ed approvvigionamento, successivamente si sono iniziate ad implementare le relazioni interne anche con le aree di vendita, distribuzione, produzione, manutenzione impianti e gestione dei progetti. Di grande importanza è il sistema di Pianificazione Fabbisogno Materiali o *Materials Requirements Planning* (MRP) la sua evoluzione MRP II (integrati nel sistema ERP) che permettono di programmare logiche di ordini automatici ai fornitori veramente sofisticate, tanto da tener conto dei tempi di consegna e di messa in produzione del prodotto. Questa metodologia permette di ottimizzare la rotazione dei materiali nei magazzini e la minimizzazione delle giacenze che impattano a livello contabile e fiscale.
A tutt'oggi i moderni sistemi di ERP coprono tutte le aree che possano essere automatizzate e/o monitorate all'interno di un'azienda, permettendo così agli utilizzatori di operare in un contesto uniforme ed integrato, indipendentemente dall'area applicativa.
I più grandi produttori di sistemi ERP sono SAP, Orale, Peoplesoft e Computer House, che dominano il mercato delle multinazionali e delle grandi imprese nazionali. Tali sistemi prevedono quindi moduli per la gestione delle risorse umane, ma richiedono l'uso di uno strumento complesso e pesante,

difficilmente adattabile al contesto di richiesta da parte della Dallara Automobili; la richiesta prevede infatti uno strumento adatto e adattato alle esigenze dell'azienda, non uno strumento completo di cui viene realmente utilizzato solo un ipotetico sottoinsieme delle funzionalità, creando un inutile complessità. Considerando inoltre il vincolo dell'utilizzo di tecnologie open source, quando qualsiasi sistema ERP in commercio è basato su software proprietario, la scelta dell'utilizzo di un sistema ERP è stata scartata.

1.3 CMS

E' possibile astrarre il problema affrontato, cioè quello dell'acquisizione di dati curriculari, parlando semplicemente di un software di acquisizione dati e relativa visualizzazione: un'altra possibile soluzione è quindi quella di utilizzare un qualsiasi CMS (*Content Management System*), cioè un sistema di gestione di contenuti, uno strumento software studiato per facilitare la gestione dei contenuti, svincolando l'amministratore da conoscenze tecniche di progettazione.
Esistono diversi CMS specializzati, cioè appositamente progettati per un tipo preciso di contenuti (eg. blog, forum, etc..) e al tempo stesso sono presenti CMS generici, che tengono ad essere più flessibile per consentire la pubblicazione di diversi tipi di contenuti. Tecnicamente un CMS è un'applicazione lato server che si appoggia su un database persistente per la memorizzazione dei contenuti, solitamente è composto da due componenti principali: un back end che gestione la sezione di amministrazione (organizzazione e supervisione della produzione dei contenuti) e la sezione applicazione di front end, che l'utente usa per inserire e fruire i contenuti.
Il vantaggio principale derivante dall'uso di tali strumenti è la facilità della costruzione e dell'aggiornamento di un sito web: senza la necessità di saper scrivere righe di codice HTML e senza la conoscenza dei linguaggi di programmazione lato server (come PHP o APS) è possibile creare e gestire un intero sito web.
Ovviamente la generalità di tali strumenti si porta dietro una serie di svantaggi che hanno precluso tale scelta come possibile soluzione al problema della Dallara Automobili: scarsa efficienza e rigidità della struttura.
Un CMS è tanto più efficiente quanto più è specializzato: per quanto un CMS possa essere flessibile, un sito basato su questa struttura in genere presenta un aspetto poco personalizzato se non è possibile intervenire direttamente sul codice sorgente del prodotto per modificarlo. Analogamente i contenuti saranno sempre ancorati a quanto previsto da chi ha progettato il CMS e non all'esigenze di chi lo utilizza.

La struttura del CMS è rigida, per quanto sia possibile modificare parte dello stile dell'interfaccia, la struttura rimarrà sempre quella ideata (ed imposta) dal creatore del CMS: questo è un limite strettamente connesso al vantaggio primario dei CMS, cioè quello di pubblicare un portale senza doverne progettare la struttura o senza possedere la conoscenze tecniche.
Non mancano però casi in cui grandi aziende o società si sono affidati a CMS open source come la Sampdoria calcio o la Foppapedretti.
Nonostante i limiti di tali strumenti, che ne hanno impedito la scelta in fase di progettazione, i CMS sono comunque alla base dell'idea che verrà presa in considerazione come soluzione al problema della Dallara Automobili:

> ***User Generated Content (UGC)**. In the early years of the web, content publishing was limited to people and institutions with access to a web server and with the skills required to build a HTML page and upload it. Content Management Systems (CMS), and most specifically blog-oriented, server-based CMS allowing users to easily write and publish their posts, were but the first step toward a new era: an era of easy content production and sharing. Tools and platforms for image sharing (such as Flickr or Picasa), video sharing (such as YouTube), audio sharing (such as podcasts) were further steps in the same direction. The new web is not just a tool for accessing information produced by institutional entities and by power users: it is an environment in which every single user can publish and share self-produced content. UGC is the core of web 2.0, and most of its tools try to address the obvious problems of sheer volume, organization, classification, evaluation, selection, retrieval, social use and preservation of such a huge amount of information. In the field of academic research, UGC implies a shift in the direction of a strictly interconnected research community, oriented not only toward the individual production of research content, but also toward its active and collaborative*

dissemination and evaluation. [1]

I CMS sono quindi rappresentativi dei primi software di condivisione delle conoscenze tra gli utenti, il cui stato passa dalla passività di ricezione delle informazioni all'attivilità della partecipazione e condivisione delle conoscenze. Anche se tale novità non riguardano direttamente il problema trattato, sono alla base dello sviluppo delle web application che sono la soluzione utilizzata per la Dallara Automobili.

[1] Gino Roncaglia, *Web 2.0 and the Future of Research: New Tools for Research Networks*, Luxembourg, 8pp, 15-16 October 2009

1.4 Web 2.0 e le Web Application

Il web 2.0 è forse uno dei termini più sfruttati e meno definiti che si possono trovare in informatica: a volte viene usato per indicare un aspetto tecnologicamente avanzato (quando magari tecnicamente è irrilevante), a volte è sinonimo di innovazione, a volte viene usato con uno scopo puratamente commerciale. Dal punto di vista prettamente tecnico, il web 2.0 non esiste in quanto è l'uso contemporaneo di una serie di tecnologie già note: HTML, Javascript, DOM, CSS ed XML che permettono la creazione di siti web in Ajax (Asynchronous JavaScript and XML), un sito web in Ajax non è altro che un sito web standard tranne che in alcune (o tutte) sue parti le richieste verso il server non sono sincrone (impone quindi il caricamento della pagina intera) ma sono asincrone, principalmente associabili a determinati eventi, e permettendo così l'interazione con il sito web senza dover ad ogni azione ricaricare la pagina intera. Quindi il Web 2.0 è un nuovo modo di realizzare siti web utilizzando tecnologie che sono sempre esistite.

1.4.1 Ajax

> *Ajax is not a technology and not a new programming languages, Ajax is a new way of think, design, develop, and a new style to programmer Web applications.* [2]

Ajax, Asynchronous Javascript and XML, è una tecnica di sviluppo per la realizzazione di siti web interattivi (Rich Internet Application). Lo sviluppo di applicazioni web-based con Ajax si basa su uno scambio di dati in background fra web browser e server, che consente l'aggiornamento dinamico di una pagina web senza esplicito ricaricamento da parte dell'utente. Ajax è quindi uno strumento asincrono, nel senso che i dati sono richiesti al server e caricati in background senza interferire con il comportamento della pagina

[2] J. Sergio Zepeda, Sergio V. Chapa, *From Desktop Applications Towards Ajax Web Applications*, 5-7 Sept. 2007 , 193 - 196, Mexico City

che ha effettuato tale richiesta.
La tecnica Ajax è quindi una combinazione di:

- *Hypertext Transfer Markup Language*, HTML/XHTML/XML
- *Cascading Style Sheets*, CSS
- *Document Object Model*, DOM
- linguaggio *client-side*, Javascript
- *XMLHttpRequest object*, XHR per l'interscambio asincrono dei dati tra il browser e il server web
- linguaggi per la manipolazione, trasformazione e scambio di dati: XML/XSLT/HTML/JSON
- protocollo di transferimento, HTTP/HTTPS
- linguaggio *server-side*, PHP/JSP/Ruby/Perl/ASP

Un applicazione web ajax usa quindi una combinazione di tecnologie già affermate nel panorama del web. E' la combinazione di queste tecnologie che determina l'unicità e la potenza di Ajax sul web, come scritte Jesse James Garrett in un suo saggio:

> *Ajax isn't a tecnology. It's a really several technologies, each flourishing in its own right, coming together in powerful new ways.*

1.4.2 Web Application

La soluzione quindi proposta è una Web Application: cioè un'applicazione dedicata accessibile via web attraverso una rete, come ad esempio una intranet o la rete Internet.
Il mondo delle web application sta ormai invadendo la vita quotidiana di chi utilizza internet: i siti web intesi come semplice strumento di informazione,

sono ormai un ricordo lontano, internet e le relative web application stanno trasportando le classiche funzionalità di un computer desktop sopra il web.

> *However, the potential of the web to deliver full scale applications didn't hit the mainstream untile Google introduced Gmail, quickly followeb by Google Maps, web-base applications with rich user interfaces and PC-equivalent interactivity* [3]

Esempio quindi rappresentativo di tale passaggio è l'evoluzione delle suite di produttività personale (eg. Microsoft Office): Google con la sua piattaforma Google Documents permette agli utenti di creare, visualizzare, salvare, stampare documenti di tipo testuale, tabellare o presentazionale. Tali operazioni per molti anni appannaggio delle applicazioni desktop, sono ora disponibili come web application online, con i relativi vantaggi: è possibile accedere da qualsiasi computer connesso ad internet, oltre che condividere i documenti in maniera praticamente istantanea e tutto questo senza nessun tipo di installazione. Le web application infatti sono applicazioni web-based, cioè applicazioni che vengono eseguite all'interno dei browser.
Rispetto quindi alle applicazioni desktop, dove solitamente gli amministratori utilizzano un programma di gestione dei dati e l'utente invece utilizza un'interfaccia (non necessariamente la stessa del programma di gestione) che può essere sia desktop che web, nelle web application tutto viene effettuato sulla rete: gli amministratori e gli utenti utilizzano gli stessi software, gli utenti inseriscono e visualizzano i dati mentre gli amministratori gestiscono sia i dati che la parte d'interfaccia, regolandone le funzionalità.
Ovviamente, le applicazioni web che usano Ajax richiedono browser che supportano tale tecnologie: Mozilla, Firefox, Opera, Konqueror, Safari, Internet Explorer(7+) e Chrome sono i principali browser che permettono l'uso delle web application.
La asincronicità di Ajax è la caratteristica principale che ha permesso lo

[3]Tim Oreilly , *What is Web 2.0: Design Patterns and Business Models for the Next Generation of Software*, O'Reilly Media

sviluppo delle web application: permette infatti di gestire in modo separato il caricamento delle pagine, offrendo così un'esperienza tipica dell'uso comune dei computer desktop, come un basso tempo di risposta o la modifica di alcune parti dello strumento utilizzato, all'interno di una normale browser web. Altro vantaggio fondamentale delle web application, è che sono indipendenti dal sistema operativo: esse vengono eseguite su un qualsiasi browser web che implementa le caratteristiche sopra descritte, annullando così qualsiasi differenza tra i sistemi operativi sottostanti.

1.5 La soluzione proposta

L'analisi del tipo di applicazione da utilizzare per la gestione delle risorse umane in Dallara Automobili ha toccato quindi diverse possibili soluzioni.
I sistemi ERP sono stati scartati in quanto proprietari e troppo complicati per una gestione più diretta e controllata del problema. Non si è voluto scegliere un software che gestisse un numero troppo grande di funzionalità non richieste in fase di definizione dei requisiti, sarebbe infatti stata una scelta di sovradimensionamento del problema che avrebbe portato ad una difficile gestione del processo di selezione delle risorse umane.
Al tempo stesso, i sistemi CMS non sono stati scelti in quanto meccanismi anch'essi troppo generici e rigidi, per quanto più vicini alle tecnologie richieste. Molto probabilmente tale scelta avrebbe comportato un impatto iniziale migliore, in quanto la struttura già pronta permette di concentrare i primi sforzi sulla definizione del modello dei dati, ma appena si fosse cercato di effettuare modifiche strutturali e di presentazione, il CMS avrebbe impedito tali modifiche o le avrebbe permesse a patto di una mole di lavoro non assolutamente proporzionata al risultato finale.
I CMS sono comunque alla base delle web-application: essi sono i primi strumenti che hanno permesso l'interazione tra gli utenti e che hanno quindi portato alla creazione delle prime web application (YouTube, Flickr, Picasa, etc...).
Nonostante per quanto riguarda l'esigenza della Dallara Automobili non ci sia bisogno, almeno in prima istanza, di creare una piattaforma di condivisione dei contenuti, si è scelto di utilizzare una web application. Tale scelta è stata effettuata nell'ottica di ottenere i vantaggi tipici di una web application: di usufruire quindi sia dell'indipendenza dalla piattaforma da cui l'utente dovrà accedere al sistema sia di tutte quelle funzionalità che rendono l'applicazione più simile ad un'applicazione di tipo desktop; fornendo così un'esperienza migliore di utilizzo del software e una conseguente facilità di gestione ed inserimento dei dati all'interno del sistema. In definitiva, si è scelto quindi di progettare un sito web completamente basato su tecnologie Ajax, in modo

da fornire un'esperienza di utilizzo moderna e innovativa.

Capitolo 2

Progettazione

2.1 Descrizione generale

Si intende realizzare un software per la gestione delle candidature in Dallara automobili, tale software oltre a fornire le più comuni funzionalità di un form di autocandidatura ordinario, deve soddisfare in pieno le esigenze dell'azienda, essere focalizzato sulla ricerca di personale richiesto dalla Dallara Automobili.
Il sistema deve consentire la gestione dei curriculum vitae attraverso un'interfaccia web, sia per gli utenti esterni sia per i gestori delle risorse umane interni all'azienda Dallara Automobili.
L'esigenza principale del sistema consiste nell'inserimento da parte di utenti esterni delle loro candidature e la gestione di tali candidature da parte degli amministratori. La gestione delle candidature include diversi aspetti:

- visualizzazione della candidatura
- aggiornamento sulla situazione della candidatura
- ricerche di candidature specifiche
- statistiche sulle candidature

2.1.1 Tipologia di utenti

Vi sono due grandi categorie di utenti del sistema:

- Utenti non registrati
- Utenti registrati

I primi potranno solo inviare una candidatura e contemporaneamente registrarsi nel sistema, avendo accesso solamente alla pagina di inserimento della candidatura. Non è dunque richiesta alcuna abilità particolare se non la dimestichezza nella navigazione internet attraverso un comune browser web.
Gli utenti registrati, invece, possono sfruttare il sistema per rivedere la propria candidatura, effettuare modifiche e controllare lo stato della propria candidatura: cioè se l'utente è stato scartato, oppure contattato, o se è stato fissato un colloquio. Anche in questo non è richiesta nessuna abilità particolare oltre la dimestichezza con l'uso di un brower web.

2.1.2 Vincoli

L'azienda in carico, la Dallara Automobili, richiede l'uso di tecnologie open source: sarà dunque necessario operare sfruttando il linguaggio Java, PHP e javascript, nonchè dover utilizzare come database MySQl e come server web sia Apache che Tomcat.
Ulteriore vincolo è dato dalla richiesta che i dati salvati e la macchina che li espone (interfaccia) debbono essere due entità separate.

2.2 Specifica dei Requisiti

In questa sezione vengono illustrati i requisiti richiesti dal sistema, sono suddivisi in requisiti funzionali e non.

2.2.1 Requisiti non funzionali

I requisiti non funzionali sono proprietà del sistema visibili all'utente e non correlate direttamente al suo comportamento funzionale:

- Il sistema deve permettere l'accesso almeno tramite un comune browser grafico (Chrome, Firefox, Internet Explorer, Safari) per utenti normodotati che funzioni su PC
- Eseguibilità su almeno una piattaforma
- Gestione concorrente di un numero di utenti dell'ordine delle centinaia attraverso il modello CREW [1]
- Tempi di risposta accettabili nell'ordine dei secondi
- Disponibilità 24 ore al giorno 7 giorni su 7

2.2.2 Requisiti funzionali

I requisiti funzionali sono le funzionalità che il sistema propone all'utente, sono così riassunte:

- Il sistema deve consentire l'accesso a tre tipologie di utenti: amministratore (interno all'azienda), utente registrato (interno ed esterno all'azienda), utente non registrato (esterno all'azienda)
- Il sistema deve permettere agli utenti non registrati, l'invio di una candidatura tramite la compilazione di un form senza ulteriore intervento da parte degli amministratori

[1] Concurrent Read, Exclusive Write

- Deve essere possibile modificare la password da parte di qualsiasi utente registrato in modo automatizzato, senza che sia richiesto l'intervento da parte dell'amministratore
- Deve essere possibile visualizzare la propria candidatura e relativo stato, da parte di qualsiasi utente registrato
- Deve essere possibile modificare la propria candidatura da parte di qualsiasi utente registrato
- Il sistema deve permettere agli utenti amministratori di creare nuovi utenti amministratori
- Il sistema deve permettere agli utenti amministratori di ricercare candidature inserite da utenti registrati
- Deve essere possibile per gli utenti amministratori salvare le candidature ricercate
- Deve essere possibile per gli utenti amministratori aggiornare lo stato della candidatura di un utente registrato

2.3 Architettura del sistema

In questa sezione viene brevemente descritta l'architettura scelta per la progettazione della web application. Decidendo di attenersi, visto la popolarità, per quanto possibile, a prodotti già esistenti e visto il vincolo della separazione dei dati dall'interfaccia, l'architettura della web application segue il noto pattern MVC.
Model View Controller è un pattern architetturale molto diffuso nell'ambito web, per esempio è alla base dell'intero meccanismo di sviluppo di Ruby on Rails; esso consta di tre tipologie di oggetti:

- Model: oggetto della web application
- View: visualizzatore dei dati contenuti nel Model
- Controller: gestore dei comandi ricevuti dall'utente attraverso il View

In pratica il Controller esegue una sorta di interfacciamento fra il Model e il View.
Essendo questa tipologia di web application piuttosto incentrata sui dati, infatti gran parte dell'elaborazione è legata alla richiesta e alla sottomissione dei suddetti dati da parte dell'utente, si è deciso di optare per questa scelta di progettazione, forti anche del largo utilizzo nel campo delle web application. Viene ora data una breve descrizione della struttura specifica della web application, in accordo con il pattern appena citato.

Model rappresenta, come già accennato, l'insieme di dati su cui è incentrata l'applicazione. Darà quindi una rappresentazione persistente a Candidature e Utenti, ovvero a tutte le entità significative legate alla web application.

View è incentrato sull'utilizzo della tecnologia Ajax: tutte le pagine della web application sono caricate dinamicamente attraverso la manipolazione del Dom da parte del Javascript, inoltre viene fatto uso delle

sessioni in Php per ovviare al problema della caratteristica *stateless* del protocollo Http.

Controller è realizzato attraverso un back_end in java messo a disposizione attraverso Apache Tomcat.
La connessione tra l'interfaccia Ajax e tale back_end è delegata ad un componente di nome PHP/Java Bridge [15]. È stato ideato seguendo fedelmente le indicazioni ricavate dalla stesura dei casi d'uso.

2.4 Strumenti e linguaggi utilizzati

Uno degli aspetti fondamentali del processo di creazione del software è dato dalla scelta degli strumenti di supporto allo sviluppo, quindi è vitale trovare gli strumenti giusti che permettano una buona riuscita del software. In questa sezione oltre ai software utilizzati vengono riassunte le principali tecnologie alla base della web application.

2.4.1 Ambiente di sviluppo

Per quanto riguarda l'ambiente di sviluppo per il back_end dai dati (il model) è stato scelto di utilizzare Eclipse [16].
Eclipse è un progetto open source legato alla creazione di una piattaforma di sviluppo ideata dalla Eclipse Foundation e fondata su una comunità strutturata sullo stile dell'open source. Pur essendo orientato allo sviluppo del progetto stesso, questo IDE è utilizzato anche per la produzione di software di vario genere. Si passa infatti da un completo IDE per il linguaggio Java (JDT, Java Development Tools) ad un ambiente di sviluppo per il linguaggio C++ (CDT, C/C++ Development Tools), con la possibilità di sfruttare plug-in che permettono di gestire, ad esempio, XML e PHP. Eclipse è scritto completamente in Java. La piattaforma di sviluppo è incentrata sull'uso di plug-in: delle componenti software ideate per fornire specifiche funzionalità. Lo stesso nucleo fondamentale dell'applicazione è fortemente modularizzato, ed è formato interamente da plugin. Nella stessa versione base è possibile programmare in Java, usufruendo di comode funzioni di aiuto quali: completamento automatico (*Code completion*) e suggerimento dei tipi di parametri dei metodi.
Essendo scritto in Java, Eclipse è disponibile per le piattaforme Linux, MacOS X e Windows. La licenza di riferimento per Eclipse è la Common Public License, che permette di creare prodotti derivati ridistribuibili gratuitamente.

Per quanto riguarda invece lo sviluppo dell'interfaccia grafica e del controller

è stato utilizzato notepad++ [17].
Notepad++ è un editor di testo libero per Windows, che supporta diversi linguaggi di programmazione. Su SourceForge, a giugno 2010, risulta essere stato scaricato più di 25 milioni di volte.
Ha un'interfaccia personalizzabile, ed è possibile aprire più documenti all'interno della stessa finestra di programma tramite l'uso delle tab.
Il progetto è basato sul componente open source Scintilla ed è scritto in C++, facendo uso esclusivamente delle API Win32, quindi senza impiegare Microsoft Foundation Classes (MFC) o librerie simili, assicurando una minore dimensione del programma e quindi un caricamento più veloce. È distribuito secondo la licenza GPL.
Notepad++ supporta l'autocompletamento, la ricerca/sostituzione tramite espressioni regolari, la scrittura a schermo diviso, il code folding, i segnalibri, l'evidenziazione delle parentesi e dell'indentazione. Supporta anche l'aggiunta di macro e plugin.

Macchina di sviluppo

La Dallara Automobili ha predisposto una macchina di sviluppo sia per l'ambiente di back_end che quello di front_end, infatti in fase di sviluppo per facilitare le operazioni si sono riunite le due componenti su una stessa macchina.
Tale macchina è stata equipaggiata con sistema operativo Ubuntu e sono stati installati i seguenti software:

- Server web Apache (con modulo php)
- Database MySql
- PhpMyAdmin
- Server Tomcat
- PHP/Java Bridge
- Firefox (con estensione Firebug)

2.4.2 PHP/Java Bridge

PHP/Java Bridge è un software che permette di il passaggio di un flusso di dati tra un motore di script, come ad esempio PHP o scheme o Python, con una *Java Virtual Machine*, attraverso un protocollo di rete basato su XML. E' 50 volte [2] più veloce che una chiamata RPC attraverso SOAP e richiede meno risorse dal punto di vista del server web. Garantisce miglior velocità e affidabilità rispetto ad una comunicazione diretta attraverso la *Java Native Interface*, e inoltre non richiede nessun componente addizionale per invocare le procedure Java dal PHP e viceversa.

[2]Dati riportati dal sito http://php-java-bridge.sourceforge.net/pjb/

2.5 Schema del sito

Figura 2.1: *Unico punto di accesso al sistema*

In questa sezione viene presentato uno schema riassuntivo delle pagine principali del sito.
L'unico punto di accesso è l'*Index.php* da cui è possibile effettuare due operazioni: il login se si è già registrati, oppure l'inizio di una registrazione che porterà sia alla creazione dell'utente all'interno del sistema che all'invio della candidatura (vedere figura 2.1).

2.5.1 Registrazione

La registrazione inizia direttamente nella pagina *Index.php*, dove all'utente viene richiesto l'inserimento di alcuni dati all'interno del sistema:

- Cognome
- Nome

- Password
- Password ReType (per controllo che la password sia stata digitata correttamente)
- Email
- Email ReType (per controllo che l'email sia stata digirata correttamente)
- CAPTCHA (controllo che l'utente sia effettivamente umano e non un bot)

Figura 2.2: *Pre-registrazione e registrazione*

Questa prima pre-registrazione serve per creare l'utente all'interno del sistema, a fine registrazione verrà inviata in automatico un'email all'indirizzo specificato contentente un link (e un codice univoco) per attivare la registrazione. L'attivazione della registrazione può essere anche effettuata facendo un primo login (ma sarà necessario riportare il codice univoco spedito via email). Ogni utente è identificato all'interno del sistema dal suo indirizzo email.
Successivamente all'inserimento della pre-registrazione, il sistema propone all'utente subito la sottimissione della sua candidatura. La prima tipologia di dati richiesti è quella anagrafica, affianco a tali dati viene anche richiesta la tipologia di candidatura che si sta sottomettendo, i valori possibili sono:

- Candidatura spontanea
- Candidatura in risposta ad un Annuncio
- Stage (in tal caso bisogna anche specificare per cosa l'utente si propone)
- Tesi (in tal caso bisogna anche specificare per cosa l'utente si propone)
- Tirocinio Formativo (in tal caso bisogna anche specificare per cosa l'utente si propone)
- Alternanza Scuola Lavoro (in tal caso bisogna anche specificare per cosa l'utente si propone)

La vera e propria candidatura è suddivisa in sei sezioni:

- Anagrafica
- Studi e Formazione
- Esperienze Professionali
- Conoscenze
- Interessi
- Upload CV

Anagrafica

Per quanto riguarda la sezione Anagrafica, l'utente deve inserire i seguenti campi:

- Cognome * [3]
- Nome *
- Data di nascita *
- Luogo *
- Provincia *
- Nazione *
- Cittadinanza *
- Seconda cittadinanza
- Stato civile * (scelta da elenco: Celibe, Nubile, Coniugato, Divorziato, Separato)
- Come è entrato in contatto con Dallara (più eventuale testo da specificare) *
- Tipologia candidatura (più eventuale testo da specificare) *
- Indirizzo email *
- Recapito telefonico *
- Ulteriore recapito telefonico
- Flag categoria protetta L. 68/99
- Percentuale invalidità

[3] * = obbligatori

- Flag categoria protetta Art. 18 Comma 2
- Indirizzo Residenza *
- Civico Residenza *
- CAP Residenza *
- Comune Residenza *
- Flag da usare nel caso in cui la residenza e il domicilio coincidano
- Indirizzo Domicilio *
- Civico Domicilio *
- CAP Domicilio *
- Comune Domicilio *
- Se il soggetto è automunito
- Patente (da scegliere: A, B, C, D, BE, CE, DE, Internazionale)

I campi Cognome, Nome ed Indirizzo Email sono riportati dalla fase di pre-registrazione e come tali non andranno re-inseriti. Il domicilio è possibile ricopiarlo dalla residenza senza doverlo inserire due volte. Se si seleziona la categoria protetta L. 68/99 è obbligatorio specificare la percentuale di invalidità, se si seleziona Art. 18 Comma 2 il sistema automaticamente seleziona il flag L.68/99 ma il campo della percentuale di invalidità viene disabilitato. In *Come è entrato in contatto con Dallara* è possibile specificare il valore con un campo di testo libero, mentre in *Tipologia candidatura* se l'utente si propone per una richiesta di collaborazione con l'azienda, deve specificare che tipo di collaborazione, sempre con un campo di testo libero.

Studi e Formazione

All'interno della sezione Studi e Formazione, il candidato inserisce tutte le informazioni rigurdanti le sue esperienze scolastiche e universitarie. Inizialmente viene richiesto se il candidato ha conseguito o sta conseguendo un Phd o un Master o altre specializzazioni di questo tipo. Le informazioni richieste sono:

- Tipologia di specializzazione *
- Struttura *
- Facoltà *
- Se conseguita o meno *
- (Se il campo precedente è stato selezionato) Anno di conseguimento *
- (Se il campo precedente è stato selezionato) Votazione *
- (Se il campo precedente è stato selezionato) Titolo e argomento della tesi *
- Media esami
- Città *
- Stato *

E' possibile specificare anche attività formative o tesi in azienda o tirocini formativi:

- Tipologia * (scegliere tra: stage, tesi in azienda, tirocinio formativo, alternanza scuola/lavoro)
- Titolo attività*
- Azienda *

- Durata *
- Data *
- Città *
- Stato *

Successivamente viene richiesto di specificare la laurea (cinque anni o 3+2), i campi da specificare sono i medesimi della sezione Phd-Master (nel caso della laurea 3+2 sarà necessario compilarne due separati) con la differenza che non sarà presente la 'Tipologia di specializzazione' e il campo Struttura è sostituito da 'Ateneo'.
Infine sono richieste le informazioni riguardo al diploma:

- Nome istituto *
- Indirizzo Tipologia istituto *
- Votazione Finale *
- Data conseguimento *
- Città *
- Stato *

Esperienze Professionali

In questa sezione viene chiesto al candidato di inserire le informazioni riguardo le sue esperienze professionali passate e presenti, sono chieste le ultime due esperienze professionali (se esistono) e nel caso in cui sono presenti ulteriori esperienze professionali è richiesto la selezione di un flag specifico (in tal caso saranno poi da verificare all'interno dei Curriculum di cui verrà fatto upload alla fine della procedura). Le informazioni richieste saranno:

- Data inizio *

- Data fine / Attuale *
- Azienda *
- Ruolo *
- Settore * (Scelta tra: Aeronautica-aerospaziale, Agenzie viaggi-tour operator, Comunicazione, Agricoltura-allevamento, Ambiente, Alberghiero, Alimentare, Assicurazioni-banche-finanziarie, Auto, Calzaturiero, Carta, Cemento-Laterizi-Ceramica, Chimica, Cinema, Commercio, Corrieri-trasportatori-logistica, Edilizia, Editoria-grafica, Elettromeccanica, Energia-acqua-gas, Engineering, Fabbricazione macchine ed apparecchi meccanici, Farmaceutica, Grande distribuzione organizzata, Industria del mobile, Industria dell'abbigliamento, Industria petrolifera, Informatica-elettronica-automazione, Navale, Ottica-occhiali, Plastica e gomma, Pubblici esercizi, Radio e televisione, Siderurgico-fonderie, Telecomunicazioni, Tessile, Trasporti, Vetro, Altro con campo di testo da specificare)
- Area * (scelta tra: Acquisti-logistica-magazzino, Amministrazione-finanza-controllo, Assistenza clienti, Comunicazioni e pubbliche relazioni, Direzione generale, Sistemi informativi, Legale, Manutenzione, Vendite e marketing, Risorse umane-organizzazione, Produzione, Qualità-sicurezza-ambiente, ReS-area tecnica, Segreteria, Servizi generale, Altro con campo di testo da specificare)
- Tipologia Contrattuale * (Sceltra tra: Contratto a tempo indeterminato, Contratto a tempo determinato, Somministrazione lavoro-interinale, Apprendistato, Collaborazione occasionale, Co.Co.Co., Co.Co.PRO., Consulenza, Altro con campo di testo da specificare)
- Motivo licenziamento o motivo ricerca di un nuovo impiego *
- RGA *

- Benefit (voci da selezionare: Acqusiti agevolati, Asili infantili, Borse di studio, Carta di credito, Check up medico, Fitness, Mutui agevolati, Polizze assicurative, Rimborso spese viaggo casa-lavoro, Spese mediche, Alloggio, Autovettura, Carburante, Cellulare, Computer portatile, Mensa/buoni pasto, Partecipazioni azionarie, Previdenza integrativa, Scuola per i figli, Altro con campo di testo da specificare)
- Città *
- Stato *

Conoscenze

Nella sezione Conoscenze, l'utente può inserire le informazioni riguardanti le sue conoscenze linguistiche e informatiche; per quanto riguarda le conoscenze linguistiche è possibile inserire fino a cinque lingue specificando i seguenti campi:

- Lingua * (menù a tendina)
- Valutazione parlato * (valori possibili: Base, Intermedio, Avanzato, Madre Lingua)
- Valutazione scritto * (valori possibili: Base, Intermedio, Avanzato, Madre Lingua)

Per quanto riguarda le conoscenze informatiche invece viene richiesto all'utente di specificare la conoscenze dei seguenti software:

- Word
- Excel
- Access
- PowerPoint
- Posta Elettronica

- Altro (con campo di testo da specificare)

Infine l'utente può specificare se ha maturato altre conoscenze usando i seguenti flag:

- Formula SAE
- Esperienze di Pista

In caso di specifica dei due precedenti campi, il sistema richiede per ognuno dei flag:

- Data inizio
- Data Fine
- Università/Azienda
- Attività svolta
- Città
- Stato

Interessi

In questa penultima sezione, l'utente inserisce le aree di interesse per cui è disposto a lavorare in Dallara Automobili. E' possibile specificare fino a quattro aree di interesse; è necessario specificare almeno la prima area di interesse che è obbligatoria mentre le restanti tre sono opzionali. Le aree di interesse sono le seguenti:

- Progettazione (Analisi Strutturali, Progettazione Meccanica, Progettazione Carrozzeria, Progettazione Parti in Materiale Composito, Progettazione Modelli di Galleria del Vento)
- Aerodinamica (Aerodinamica Sperimentale, Aerodinamica Numerica, Galleria del Vento - Operazioni, Modelleria, Prototipazione Rapida)

- ReS (Dinamica del veicolo, Progettazione, Sperimentazione Outdoor)
- Produzione (Programmazione Produzione, Macchine Utensili, Carpenteria, Compositi - Laminazione, Carrozzeria - Finitura, Compositi, Montaggio Vetture)
- Operazioni (Acquisti, Logistica, Magazzino)
- Qualità (Sistema qualità, Controllo qualità)
- Amministrazione-finanza-controllo (Amministrazione e contabilità, Finanza, Controllo di gestione)
- Risorse Umane
- Legale
- Stile
- Sistemi informativi
- Vendite e Marketing (Vendite, Marketing)
- Comunicazione e Pubblice Relazioni
- Segreteria (Assistenza alla direzione, Reception)

Inoltre l'utente deve anche specificare le disponibilità per il raggiungimento dell'azienda, la disponibilità ad effettuare trasferte all'estero e la disponibilità di trasferirsi all'estero (selezionare: USA, Altro specificando con un campo di testo). Per ognuna di queste disponibilità l'utente è obbligato a selezionare 'si' o 'no', non può non specificare la scelta.

Upload CV

L'ultima sezione obbliga l'utente ad accettare l'informativa sulla privacy (per permettere l'uso dei dati salvati) e a specificare opzionalmente un documento di cui fare l'upload, il documento deve avere una dimensione massima di 2MB e avere estensione .doc o .pdf.

2.5.2 Gestione da parte degli utenti

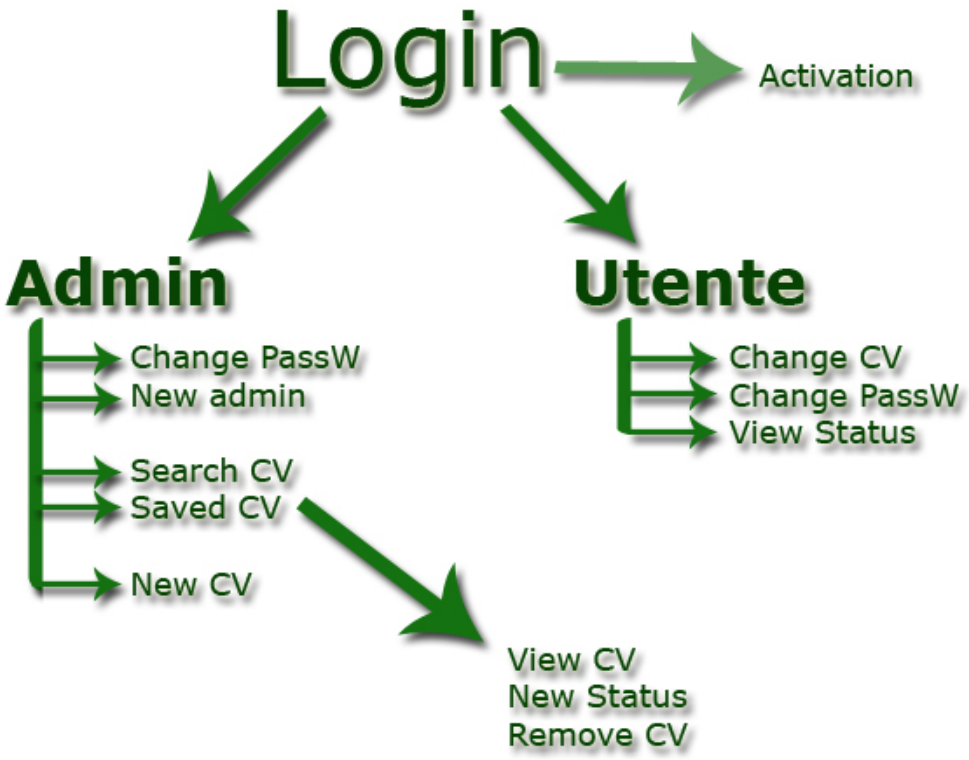

Figura 2.3: *Gestione utente*

Una volta effettuato il login, il sistema in base alle credenziali indirizza l'utente in due sezioni differenti: l'amministratore del sistema e il candidato. Se l'utente non è stato ancora attivato allora viene reindirizzato in un'area di attivazione dell'account, dove l'utente deve inserire il codice che ha ricevuto nell'email; senza l'attivazione non è possibile proseguire.

Amministratori di sistema

Gli amministratori di sistema hanno a disposizione cinque funzionalità:

- cambio password

- creazione di un nuovo amministratore di sistema
- ricerca delle candidature
- gestione delle candidature salvate
- inserimento nuova candidatura

Attraverso la funzionalità 'cambio password', l'amministratore può cambiare la propria password personale, deve inserire due volte (per sicurezza) la password attuale e una volta la nuova password.
Ogni amministratore inoltre può creare altri amministratori di sistema, per fare ciò bisogna inserire un nome utente ed un'email; il sistema in automatico genererà una password che viene inviata via email al nuovo amministratore. Non c'è bisogno di attivare gli account degli amministratori.
Con la pagina delle ricerche, gli amministratori invece possono ricercare attraverso le candidature memorizzate all'interno del sistema, ci sono tre modalità di ricerche:

- Base
- Avanzata
- Libera

Con la modalità Base le ricerche sono effettuate esclusivamente attraverso Nome, Cognome ed Email (con almeno uno dei tre valori) mentre con la modalità Avanzata è possibile fare ricerche attraverso ogni singolo parametro inserito dai candidati al momento dell'inserimento della candidatura; quindi si potranno fare ricerche su ogni criterio delle sezioni Anagrafica, Studi e Formazione, Esperienze professionali, Conoscenze ed Interessi. E' possibile anche fare ricerche con la modalità Libera, dove specificando del testo il sistema ricerca automaticamente all'interno di tutti i criteri il testo inserito.
Una volta ricercate le candidature, l'amministratore di sistema, oltre che visualizzarle, può anche salvarle nelle proprie candidature, in modo da raggiungerle direttamente dal proprio pannello di controllo. Nella sezione delle

candidature salvate è possibile sia aggiornarne lo stato (quindi inserire un nuovo evento per il candidato) che eliminarle.
Ogni candidato assume un diverso stato in base agli eventi a cui viene associato, gli stati sono identificati da diversi colori:

- Grigio: appena inserito nel sistema
- Giallo: contattato dalla Dallara
- Arancione: in colloquio
- Verde: assunto
- Rosso: scartato

Gli eventi possibili da associare ai candidati sono:

- Inserito da Form Autocandidatura, risposta automatica in Data - Grigio
- Inserito da Amministratore - Grigio
- Contattato telefonicamente - Giallo
- Contattato via Email - Giallo
- In attesa di Colloquio - Giallo
- Colloquiato in Data - Arancione (è possibile aggiungere un 'Documento Scheda Colloquio')
- Assunto - Verde
- Scartato - Rosso

Ovviamente ognuno di questi eventi sarà associato ad un amministratore e ad una data.
Infine l'ultima funzionalità del sistema per gli amministratori è la possibilità di inserire candidature dal proprio account.

Candidati

Il candidato che effetta il login nel sistema può effettuare le seguenti operazioni:

- cambio password
- modifica la candidatura
- visualizzare lo stato della propria candidatura

Attraverso la funzionalità 'cambio password', il candidato può cambiare la propria password personale, deve inserire due volte (per sicurezza) la password attuale e una volta la nuova password.
Appena effettuato il Login, il candidato si trova di fronte ad un riepilogo della sua candidatura, con l'elenco degli stati associati.
Il candidato può visualizzare e aggiornare la propria candidatura in ogni momento, per ogni sezione è possibile cambiare i valori o compilare i campi che non erano obbligatori al momento della compilazione.

2.6 Schema ER del database

In questa sezione viene presentato lo Schema ER del database a partire dalle specifiche della sezione precedente.
Lo schema è concettualmente suddiviso come l'inserimento della candidatura dell'utente, quindi nelle sei sezioni: Anagrafica, Studi e Formazione, Esperienze lavorative, Conoscenze, Interessi e Upload CV; in realtà dal punto di vista del database la prima e l'ultima sezione, cioè Anagrafica e Upload CV, sono accorpate.

2.6.1 Anagrafica e Upload CV

Il fulcro principale del database è l'entità *candidatura*, tale entità contiene tutte le candidature presenti nel sistema, ogni candidatura è associata

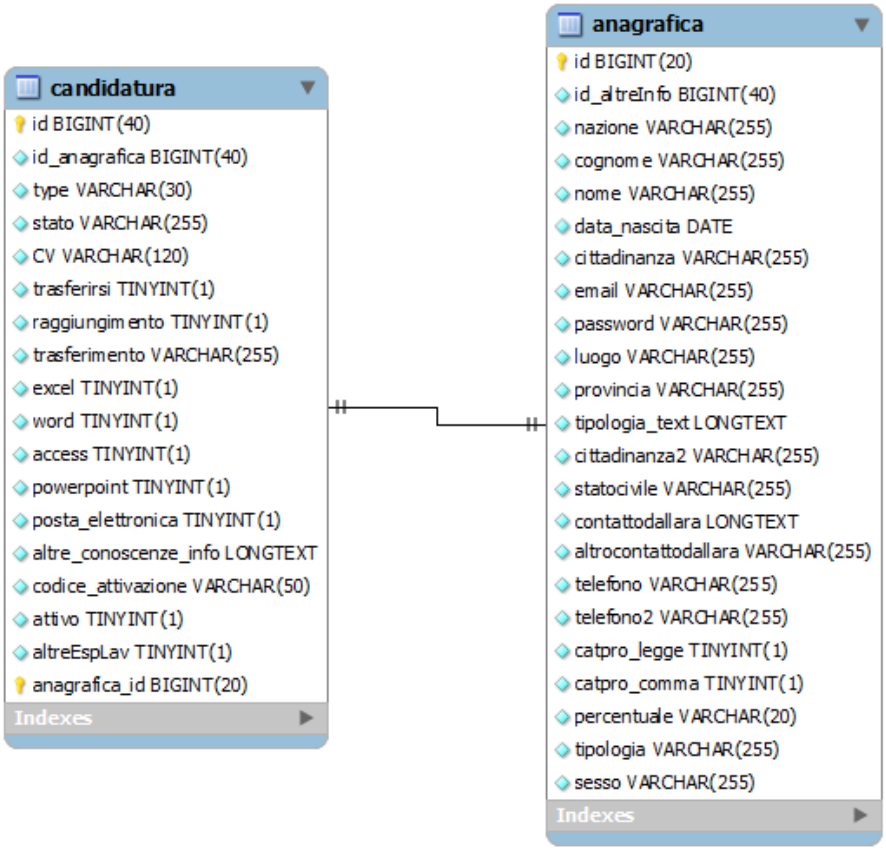

Figura 2.4: *Schema ER Anagrafica e Upload CV*

in modo univoco ad un elemento dell'entità *anagrafica* che invece rappresenta l'utente che ha inserito la candidatura.
Concettualmente la due entità, *candidatura* e *anagrafica*, potrebbero essere un'unica entità ma per ragioni di comodità e di perfomance (evitare l'uso di tabelle con un alto numero di colonne) è stato deciso di separare le informazioni per tipologia, quindi nell'entità *candidatura* sono presenti tutte le informazioni della candidatura e nell'entità *anagrafica* invece le informazioni anagrafiche del soggetto; per questo motivo nello schema in figura 2.4 l'entità *candidatura* risulta una sotto-entità dell'entità *anagrafica*.

2.6.2 Studi e formazione

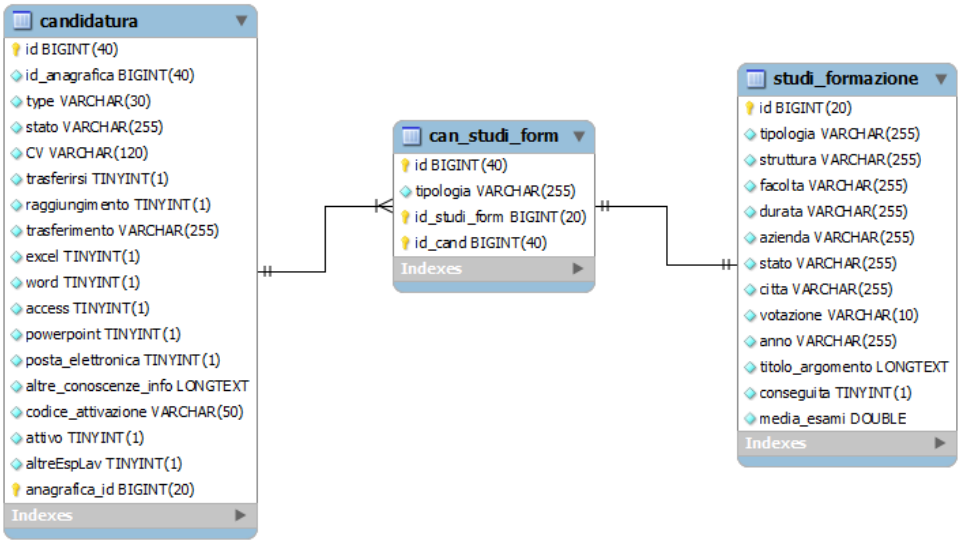

Figura 2.5: *Schema ER Studi e Formazione*

Come si può vedere in figura 2.5, tutte le informazioni riguardo attività formative, master, lauree e diplomi sono state accorpate in un'unica entità di nome *studi_formazione*, in questo modo ogni candidatura è legata da una relazione 1 a N con l'entità *studi_formazione* e attraverso questo legame viene specificata la tipologia: se master o attività formativa o laurea o infine diploma.
Di conseguenza ogni elemento di *studi_formazione* non avrà tutte le colonne compilate in quanto i vari titoli di studi non si sovrappongono completamente, ma solo per alcuni campi.

2.6.3 Esperienze Lavorative

Come per la sezione 'Studi e Formazione', anche per le esperienze lavorative l'entità *candidatura* si ritrova in una relazione 0 a N con l'entità *esperienze_lavorative*, anche in questo caso la relazione sarebbe più specificatamente da 0 a 2, in quanto ogni candidato può specificare fino a due

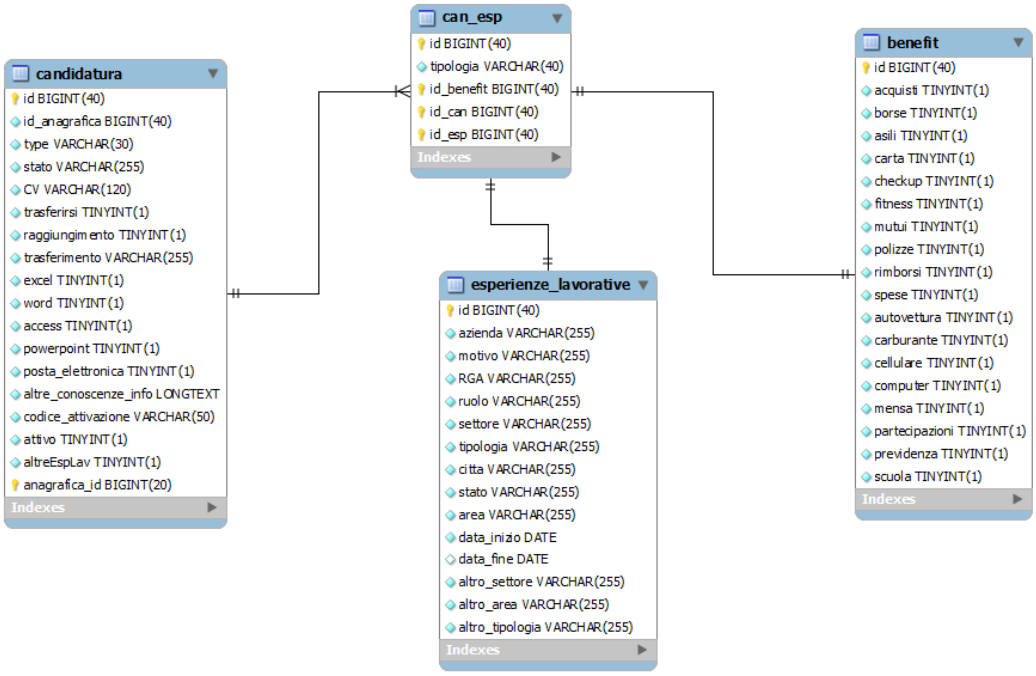

Figura 2.6: *Schema ER Esperienze Lavorative*

esperienze lavorative.
Per quanto riguarda i Benefit, per semplicità sono stati salvati in una tabella a parte per evitare di rendere troppo complessa la tabella *esperienze_lavorative*.

2.6.4 Conoscenze

Le conoscenze linguistiche ed informatiche sono immagazzinate in modo diverso: le conoscenze linguistiche sono in un'entità *lingue_conosciute* in relazione 1 a 1 (e 1 a N nell'altro verso) con l'entità *candidatura* mentre per quanto riguarda le informazioni sulle conoscenze informatiche, sono salvate all'interno dell'entità *candidatura*. Nella figura 2.7 si può infine notare come le conoscenze della formula SAE e le eventuali esperienze in Pista sono racchiuse nell'entità *sae_pista* in relazione 1 a 1 (e 0 a N nell'altro verso) con l'entità *candidatura*, gli attributi dell'entità sono in comune, la discriminazione tra

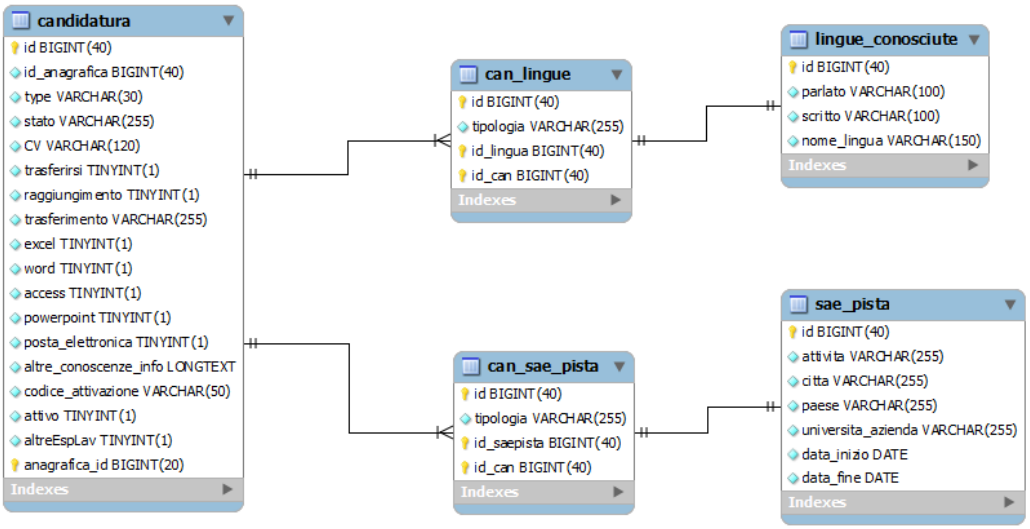

Figura 2.7: *Schema ER Conoscenze*

le due sezioni è effettuata nella relazione *can_sae_pista* .

2.6.5 Interessi

Per quanto riguarda l'ultima categoria, le aree di interesse sono salvate in un'entità di nome *interessi*, quest'entità ha il solo attributo *interesse*, che corrisponde al dato scelto del candidato al momento dell'invio della candidatura. Le informazioni relative alle disponibilità invece sono salvate direttamente nell'entità *candidatura* come si può vedere in figura 2.8 .

2.6.6 Amministratori

Gli amministratori sono entità separate da quelle degli utenti registrati: sono salvati in un'entità di nome 'Admin' con gli attributi Email e Password. Gli amministratori potendo salvare le candidature, sono collegati in una relazione N a N con le istanza della tabella candidatura.

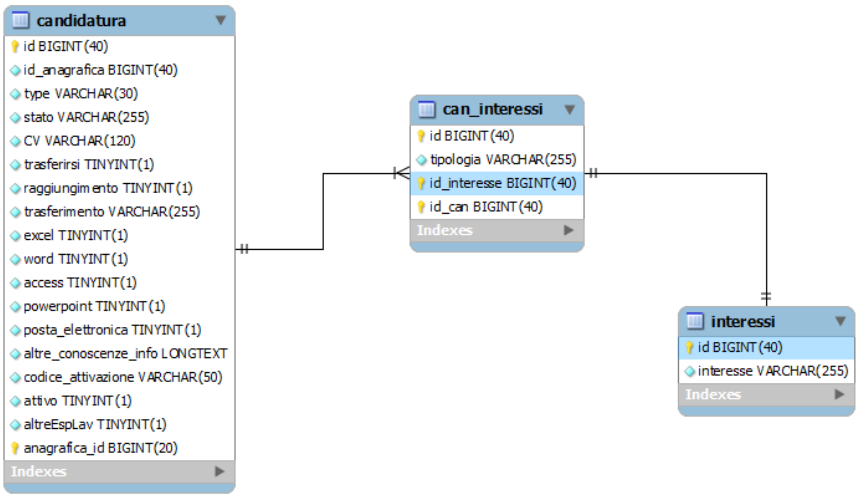

Figura 2.8: *Schema ER Interessi*

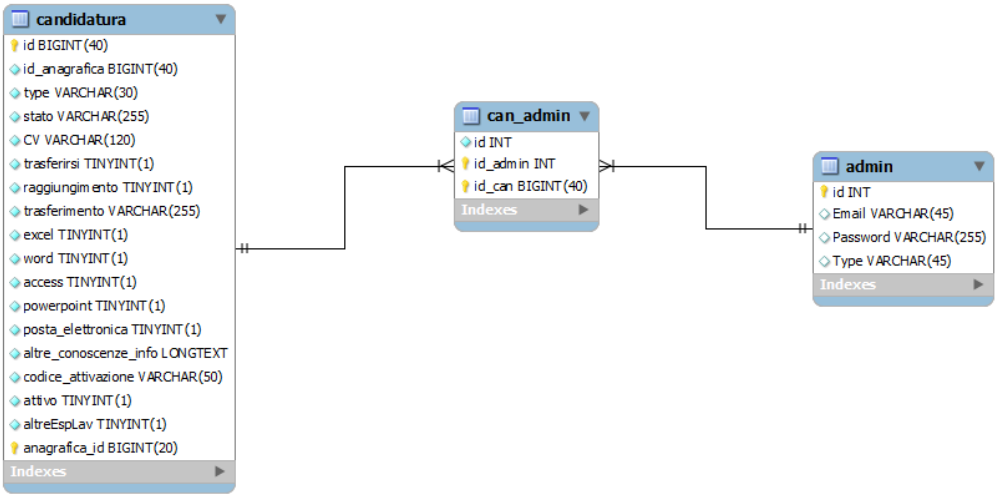

Figura 2.9: *Schema ER Amministratori*

Capitolo 3

Sviluppo e implementazione

Lo sviluppo dell'applicazione è stato completamente eseguito su piattaforma Windows a causa della necessità di garantire il supporto per Internet Explorer (dalla versione 8). Tale supporto è garantito inoltre per Chrome, Safari e Firefox.
Lo sviluppo al momento della stesura di questa Tesi di Laurea è arrivato a coprire il 60% del lavoro descritto nel capitolo di Progettazione: manca la gestione degli utenti amministrativi e la doppia lingua in inglese.
In fase di sviluppo, vista la struttura adottata, si è deciso di spostare alcune parti di logica all'interno dei file .php responsabili dell'interazione con il modello in Java. In questo modo alcuni controlli o più generalmente parti logiche sono presenti in questi file con la funzione di collante tra il modello dei dati e il controller.
Il back_end in java quindi effettua operazioni esclusivamente di salvataggio e recupero dei dati, vengono lanciati errori solo nel caso in cui il database risulti inaccessibile, in quanto tale back_end presuppone che i dati ricevuti siano sempre corretti.
Gli strumenti utilizzati per lo sviluppo sono i seguenti:

- Notepad++: editor per la scrittura sia dei file php/html che degli script javascript
- Eclipse: software per la scrittura delle pagine in java del back-end

- Photoshop: software di fotoritocco usato per la creazione della veste grafica

3.1 Configurazione ambiente

L'ambiente di produzione del software è composto da due macchine separate: una macchina si occupa di gestire il front-end (html, css, php, javascript) e la seconda di gestire i dati con il back-end (mysql e java). Il punto di unione tra le due macchine è un bridge di nome Php/Java Bridge.
Tale software permette la chiamata a metodi java attraverso specifiche chiamate in php. Requisiti necessari sono un server Apache che interpreta gli script php e un server Tomcat che mette a disposizione un servizio java all'indirizzo della macchina su quale gira. Le pagine php in questo modo definendo le seguenti variabili ed includendo un file specifico di configurazione:

```
define("JAVA_HOSTS", "INDIRIZZO:PORTA");
define("JAVA_SERVLET", "/bridge/NOME_SERVIZIO");
require_once("Java.inc");
```

possono connettersi all'INDIRIZZO:PORTA sul quale tomcat dovrà eseguire il bridge di nome NOME_SERVIZIO, a tal punto sarà possibile chiamare metodi propri del NOME_SERVIZIO direttamente come chiamata php ed ottenere i relativi risultati, ad esempio:

```
java_context()->getServlet()->RemoteMethod($param1,$param2,...));
```

Tutte le problematiche relative al passaggio dei parametri e alla consistenza delle chiamate è gestito dal PHP/Java Bridge.

3.2 Organizzazione strutturale

Lo Sviluppo segue ovviamente la progettazione e come tale rispecchia chiaramente il pattern Model View Controller: il Model è rappresentato dal back-end in java che salva e prende i dati dal database, mentre per quanto riguarda View e Controller, essi risiedono sulla medesima macchina, cioè il front-end. L'organizzazione del front-end è così formata:

- Cartella principale del sito
- Cartella css
- Cartella controller
- Cartella model
- Cartella image

3.2.1 Cartella principale del sito

E' la cartella di accesso al sito, contenente tutte le altre cartelle elencate precedentemente, all'interno sono presenti i seguenti file in php:

- Index.php: rappresenta la pagina principale del sito dove è possibile inserire le candidature ed effettuare il login
- Activate.php: pagina che permette l'attivazioni dei profili utenti
- User.php: rappresenta la pagina di gestione degli utenti
- Admin.php: pagina di gestione per gli amministratori del sito

Tali pagine sono completamente vuote: la creazione del codice Html di tali pagine è delegata al controller in javascript che riempe il DOM delle singole pagine in base al flusso determinato dalle scelte dell'utente. La differenza tra le varie pagine è data dall'inclusione di file javascript diversi e dal passaggio di parametri al controller.

Importante sottolineare come l'unico punto di accesso rimane comunque la pagina Index.php, infatti se un utente provasse ad accedere ad una delle altre pagine (senza aver ancora effettuato il login) allora verrebbe indirizzato alla pagina Index.php stessa.

3.2.2 Cartella css

La cartella css contiene i file di stile che vengono utilizzati dalle pagine php della sezione precedente.
E' presente un unico file di stile per la presentazione, affiancato da un file di stile specifico per le versioni di Internet Explorer precedenti alla 8.

3.2.3 Cartella controller

Nella cartella controller sono contenuti tutti i file che implementano la logica del sistema, cioè i file javascript.
E' presente un file principale di nome *lib.js* che racchiude il cuore vero e proprio della logica: infatti all'interno di questo file vengono prese le scelte riguardo quale pagina mostare e quindi quale funzione chiamare. Il file *lib.js* richiama funzioni presenti in tutti gli altri file *.js*:

- libAnagrafica.js: file che si occupa sia di riempire il DOM della pagina con il form per l'inserimento (o modifica) della prima parte dell'anagrafica sia di inviare il salvataggio di tali dati al back-end attraverso una chiamata Ajax

- libAnagrafica2.js: file che si occupa sia di riempire il DOM della pagina con il form per l'inserimento (o modifica) della seconda parte dell'anagrafica sia di inviare il salvataggio di tali dati al back-end attraverso una chiamata Ajax

- libConoscenze.js: file che si occupa sia di riempire il DOM della pagina con il form per l'inserimento (o modifica) delle conoscenze sia di inviare il salvataggio di tali dati al back-end attraverso una chiamata Ajax

- libEspeLav.js: file che si occupa sia di riempire il DOM della pagina con il form per l'inserimento (o modifica) delle esperienze lavorative sia di inviare il salvataggio di tali dati al back-end attraverso una chiamata Ajax

- libFormazione.js: file che si occupa sia di riempire il DOM della pagina con il form per l'inserimento (o modifica) della formazione sia di inviare il salvataggio di tali dati al back-end attraverso una chiamata Ajax

- libInteressi.js: file che si occupa sia di riempire il DOM della pagina con il form per l'inserimento (o modifica) degli interessi sia di inviare il salvataggio di tali dati al back-end attraverso una chiamata Ajax

- libRegistration.js: file che si occupa dell'inizio e della fine della registrazione, permette di riempire il DOM della pagina con l'inizio della registrazione e la conclusione (che porta all'attivazione)

- libUser.js: file che si occupa dell'intera gestione (sia DOM che di salvataggio) della sezione dell'utente

3.2.4 Cartella model

La cartella model è composta da un insieme di file con estensione php che permette di comunicare con il back_end attraverso il PHP/Java Bridge. E' proprio in questi file che vengono incluse le righe descritte nella sezione precedente *Configurazione ambiente*.
Tali file sono quindi responsabili del passaggio di informazioni tra il javascript (il controller e quindi l'utente) e il back_end in java (il modello vero e proprio):

- activate.php: file che prende in input il codice di attivazione inserito dall'utente e controlla se è equivalente a quello generato al momento del salvataggio, in caso positivo attiva l'account dell'utente

- cancelRegistration.php: file che gestisce l'operazione di annullamento durante l'inserimento della registrazione, cancella quindi a cascata tutte

le informazioni inserite fino a quel momento, è utilizzato inoltre anche quando l'utente non effettua l'attivazione entro un determinato periodo

- changePw.php: permette di modificare la password dell'utente
- checkEmail.php: controlla se l'email è già stata utilizzata o meno all'interno del sistema
- checkFile.php: controlla se ad una determinata candidatura è stato già associato un file o meno
- dati.php: in questo file sono presenti le tre righe di inclusione necessarie per connettersi al back_end, tale file infatti è incluso da tutti gli altri file .php presenti nella cartella model
- endRegistration.php: file che si occupa di generare il codice di attivazione per una determinata candidatura
- GetDati.php: uno dei file principali della cartella model, in quanto permette di acquisire dati riguardo una determinata candidatura, per ottenere tali dati bisogna necessariamente specificare un parametro *$type* che può assumere i seguenti valori: *anagrafica1, anagrafica2, formazione, espelav, conoscenze, interessi e file*
- GetUser.php: restituisce Nome e Cognome di una determinata candidatura
- login.php: funzione che esegue il login a partire dai dati inseriti, gestisce sia il login errato che il login corretto ma anche il caso in cui il login è corretto ma l'utente debba ancora effettuare l'attivazione
- logout.php: distrugge la sessione php
- newRegistration.php: crea una nuova riga nelle tabelle Candidatura e Anagrafica del database

- saveAnagrafica.php: permette l'inserimento o la modifica della prima parte di anagrafica di una determinata candidatura
- saveAnagrafica2.php: permette l'inserimento o la modifica della seconda parte di anagrafica di una determinata candidatura
- saveConoscenze.php: permette l'inserimento o la modifica delle conoscenze di una determinata candidatura
- saveEspeLav.php: permette l'inserimento o la modifica delle esperienze lavorative di una determinata candidatura
- saveFormazione.php: permette l'inserimento o la modifica della formazione di una determinata candidatura
- saveInteressi.php: permette l'inserimento o la modifica degli interessi di una determinata candidatura
- updateSession.php: aggiorna la sessione php
- verifyCaptcha.php: verifica la correttezza del codice CAPTCHA inserito dall'utente

3.2.5 Cartella image

Cartella che contiene tutti le immagini utilizzati dall'interfaccia grafica web. Tali file sono tutti in formato .png.

3.3 Back_end

Il back_end è stato sviluppato in java ed è un servizio messo a disposizione del server Tomcat per il linguaggio server-side PHP, attraverso il PHP/Java Bridge.
Il php in questo modo può effettuare delle chiamate attraverso l'istanza *java_context()->getServlet()*, tale istanza ha tutti i metodi resi pubblici dalla classe java, Action.java, del bridge definito all'interno di tomcat. In questo modo basta aggiungere un metodo pubblico all'Action.java, che dal php sarà possibile richiamare la funzione con i corrispondenti parametri, la gestione del passaggio dei parametri è completamente delegata al PHP/Java Bridge, sia il php che il java sono completamente trasparenti a questo passaggio.
I metodi messi a disposizione da parte del back_end sono i seguenti:

- *ChangePw(Integer id_candidatura, String oldpass, String pass)*: permette di modificare la password associata ad una candidatura
- *Login(String user, String pass)*: controlla se il login è corretto o meno
- *checkEmail(String email)*: controlla se l'email è già presente nel sistema
- *checkFile(Integer id_candidatura)*: controlla se l'utente ha già associato un file o meno alla propria candidatura
- *getFieldCandidatura(Integer id_candidatura, String field)*: restituisce il valore del campo *field* di tipo intero a partire da una candidatura
- *getFieldStringCandidatura(Integer id_candidatura, String field)*: restituisce il valore del campo *field* di tipo string a partire da una candidatura
- *updateFieldCandidaturaString(Integer id_candidatura, String field, String value)*: aggiorna il valore del campo *field* di tipo string a partire da una candidatura

- *updateFieldCandidatura(Integer id_candidatura, String field, String value)*: aggiorna il valore del campo *field* di tipo intero a partire da una candidatura

- *saveInteresse(Integer idcandidatura, String value, String type)*: permette di salvare un'interesse legato ad una specifica candidatura

- *saveLanguage(Integer id_candidatura, String lingua, String parlato, String scritto, String type)*: permette di salvare una lingua legata ad una specifica candidatura

- *saveSaePista(Integer id_candidatura, String inizio, String fine, String azienda, String attivita, String citta, String paese, String type)*: permette di salvare sia un'esperienza di pista che un'esperienza SAE legate ad una specifica candidatura

- *saveBenefit(Integer id_candidatura, Integer acquisti, Integer borse, Integer asili,Integer carta, Integer checkup, Integer fitness, Integer mutui, Integer polizze, Integer rimborsi, Integer spese, Integer autovettura, Integer carburante, Integer cellulare, Integer computer, Integer mensa, Integer partecipazioni, Integer previdenza, Integer scuola, String type)*: inserisce/modifica i benefit associati ad una candidatura

- *saveEspLav(Integer id_candidatura,String azienda, String motivo, String RGA, String ruolo, String settore, String tipologia, String citta, String stato, String area, String data_inizio, String data_fine, String settoreAltro, String areaAltro, String tipologiaAltro, String type)*: inserisce/modifica le esperienze lavorative associate ad una candidatura

- *saveFormazione(Integer id_candidatura,String specializzazione, String struttura, String facolta, String durata, String azienda, Integer conseguita, String anno, String votazione, String titolo, Double esami, String citta, String stato, String type)*: inserisce/modifica la formazione associata ad una candidatura

- *removeLanguage(Integer id_candidatura,String type)*: rimuove una lingua associata ad una candidatura
- *removeInteresse(Integer id_candidatura,String type)*: rimuove un interesse associato ad una candidatura
- *removeSaePista(Integer id_candidatura,String type)*: rimuove un'esperienza di pista/SAE associata ad una candidatura
- *removeFormazione(Integer id_candidatura,String type)*: rimuove un'esperienza formativa associata ad una candidatura
- *removeEspeLav(Integer id_candidatura,String type)*: rimuove un'esperienza lavorativa associata ad una candidatura
- *updateAnagraficaOne(Integer id_candidatura,String cognome, String nome, String datanascita, String luogo, String provincia, String nazione, String cittadinanza, String cittadinanza2, String statocivile, String contattodallara, String email, String telefono, String telefono2, String categoria, String to182, String percentuale, String tipologia, String tipologiaText, String altrocontattodallara, String sesso)*: aggiorna la prima parte dell'anagrafica associata ad una candidatura
- *updateAnagraficaTwo(Integer id_candidatura,String indirizzoRES, String civicoRES, String capRES, String comuneRES, String indirizzoDOM, String civicoDOM, String capDOM, String comuneDOM, Integer automunito, Integer pat_a, Integer pat_b, Integer pat_c, Integer pat_d, Integer pat_be, Integer pat_ce, Integer pat_de, Integer pat_int)*: aggiorna la seconda parte dell'anagrafica associata ad una candidatura
- *getDati(Integer id_candidatura)*: restituisce Nome, Cognome ed Email associati ad una candidatura
- *getAnagrafica1(Integer id_candidatura)*: restituisce le informazioni della prima parte di anagrafica associata ad una candidatura

- *getAnagrafica2(Integer id_candidatura)*: restituisce le informazioni della seconda parte di anagrafica associata ad una candidatura
- *getLingue(Integer id_candidatura)*: restituisce le lingue associate ad una candidatura
- *getInteressi(Integer id_candidatura)*: restituisce gli interessi associati ad una candidatura
- *getFormazione(Integer id_candidatura, String type)*: restituisce una delle esperienze formative (in base al parametro *type*) associate ad una candidatura
- *getEspLav(Integer id_candidatura, String type)*: restituisce una delle esperienze lavorative (in base al parametro *type*) associate ad una candidatura
- *getBenefit(Integer id_candidatura, String type)*: restituisce i benefit associati ad una delle esperienze lavorative (in base al parametro *type*) associate ad una candidatura
- *getSaePista(Integer id_candidatura, String type)*: restituisce una delle esperienze di pista/SAE (in base al parametro *type*) associate ad una candidatura
- *newAnagraficaInit(String sCognome, String sNome, String sEmail, String sPassword)*: inserisce un nuovo record nella tabella dell'anagrafica
- *newCandidatura(Integer id_anagrafica)*: inserisce un nuovo record nella tabella candidatura, associandola ad un elemento della tabella anagrafica
- *cancelRegistration(Integer id_candidatura)*: cancella la registrazione da dentro il database, si occupa di cancellare a cascata tutte le informazioni associate alla candidatura

Come è facile intuire, la struttura logica della candidatura è ripetuta per ogni sezione: è infatti presente una funzione di inserimento/salvataggio ed una di cancellazione per tutte e sei le sezioni della candidatura.
Tutte queste funzioni non effettuano nessun tipo di controllo sui dati che ricevono, infatti si presuppone che la logica abbia già scremato e controllato i dati e che quindi il back_end li riceva corretti.
A titolo di esempio viene mostrato il codice della funzione *saveFormazione(...)*, tale funzione si occupa di inserire o modificare i dati relativi ad un'esperienza formativa associata ad una candidatura:

```
public String saveFormazione(String id_candidatura,
String specializzazione, String struttura, String facolta,
String durata, String azienda, Integer conseguita,
String anno, String votazione, String titolo,
    Double esami, String citta, String stato, String type){
Connection conn = null;
Statement stmt = null;
ResultSet rst = null;
long nID = -1 ;
String sMsg = "" ;
String result = "";
try {
    conn = Global.getConnectionDb() ;
    if( conn != null ){
        String sSQL= "SELECT id_studi_form FROM can_studi_form
        where id_cand='"+id_candidatura+"'
        and tipologia='"+type+"'";
        stmt = conn.createStatement() ;
        rst = stmt.executeQuery( sSQL ) ;
        if( rst.next() ) {
            nID = rst.getLong( "id_studi_form" ) ;
            sSQL= "UPDATE studi_formazione set
```

```
        tipologia='"+specializzazione+"',
        struttura='"+struttura+"',
        facolta='"+facolta+"',durata='"+durata+"',
        azienda='"+azienda+"',stato='"+stato+"',
        citta='"+citta+"',
        votazione='"+votazione+"', anno='"+anno+"',
        titolo_argomento='"+titolo+"',
        conseguita='"+conseguita+"',
        media_esami='"+esami+"' where id="+nID;
        stmt = conn.createStatement() ;
        stmt.executeUpdate( sSQL) ;
    }else{
        sSQL= "INSERT into studi_formazione (tipologia,
        struttura,facolta,durata,azienda,stato,citta,
        votazione,anno, titolo_argomento,conseguita,
        media_esami) " + "values
        ( '" + specializzazione + "','" + struttura + "',
        '" + facolta + "','" +durata+"', '"+azienda+ "',
        '" + stato + "','" + citta + "','"+votazione+"',
        '" + anno + "','" + titolo + "',"+conseguita+",
        " + esami + ")" ;
        stmt = conn.createStatement() ;
        nID = stmt.executeUpdate(
        sSQL, Statement.RETURN_GENERATED_KEYS ) ;
        rst = stmt.getGeneratedKeys();
        while (rst.next()){
            nID = rst.getInt(1);
        }
        sSQL= "INSERT into can_studi_form
        (id,id_studi_form,id_cand,tipologia)
        values (null,'"+nID+"','"+id_candidatura+"',
```

```
            '"+type+"')";
            stmt = conn.createStatement() ;
            stmt.executeUpdate( sSQL) ;
        }
        updateFieldCandidaturaString(id_candidatura,
        "stato","espe");
        result = "ok";
    }
}
catch( Exception e ){
    sMsg= e.getMessage();
    result = "error****"+sMsg;
}
finally{
    try{
        Global.closeCon( conn, stmt, rst ) ;
    }
    catch( Exception e ){
        sMsg= e.getMessage()  ;
    }
}
return result;
}
```

Come già descritto, non vengono effettuati nessun tipo di controlli sui dati ricevuti, la funzione semplicemente a partire dall'id della candidatura e dal tipo di esperienze formativa (se phd, o laurea 5 anni o diploma, etc...) controlla se già presente nel sistema: in caso negativo allora procede nell'inserimento, alternativamente effettua invece un aggiornamento dei dati; la clausola finally è presente in tutti i metodi del back_end è serve per forzare la chiusura della connessione verso il database, altrimenti ogni chiamata di metodo lascerebbe una connessione pendente verso la base di dati.

3.4 Front_end

Il front_end è composto da due parti: abbiamo un'insieme di librerie javascript che effettuano la creazione delle pagine Html e si occupano della logica del sistema, la seconda parte invece è composta da una serie di file in php che si occupano di dialogare con il back_end e di ripercuotere i dati verso il javascript.
L'insieme dei file in php che dialogano con il back_end si possono vedere come un'estensione del javascript, infatti tali file sono utilizzati esclusivamente come tramite tra il javascript e il back_end e quindi possono essere visti come parte integrante della logica e del front_end.

3.4.1 Pagine principali e stato del sistema

Come già descritto in precedenza, le pagine Html (Index.php, Activate.php e User.php) sono pagine vuote: il DOM di tale pagine è creato dinamicamente in base allo stato in cui si trova l'utente durante la navigazione.
Essendo il protocollo HTTP intrinsecamente *stateless* (senza stato), sono state utilizzate le funzioni di libreria di php che permettono di salvare sessioni all'interno delle pagine durante la navigazione. Quindi al momento del primo accesso al sistema, viene inizializzata una nuova sessione (attraverso la pagina Index.php): poi se l'utente effettua una nuova registrazione allora vengono create due nuove variabili di sessione *next* e *idcand* che rappresentano rispettivamente la prossima pagina da visualizzare e il nuovo id appena creato della candidatura; se si effettua il login invece *next* e *idcand* sono settate a 'user' e all'id della candidatura corrispondente al login corretto.
I valori possibili che può assumere la variabile *page* sono: index, user e activate. Tali valori corrispondono esattamente alle pagine php. Mentre la variabile *next* può assumere diversi valori:

- viewCV: la pagina richiesta è quella di visualizzazione del CV (può essere chiamata solo da user.php)

- changePw: la pagina richiesta è quella di modifica della password utente (può essere chiamata solo da user.php)
- user: la pagina richiesta è quella principale del profilo utente (può essere chiamata solo da user.php)
- activate: la pagina richiesta è quella di attivazione dell'utente (può essere chiamata solo da activate.php)
- end: la pagina richiesta è quella che porta all'attivazione dell'account (può essere chiamata solo da index.php)
- fine, inte, cono, espe, form, anag2, anag: sono le pagine che corrispondono alle fasi dell'inserimento della candidatura, possono essere chiamate sia da user.php che da index.php e in base al tipo di pagina vengono chiamate funzioni diversi (prese da due file javascript diversi)

In questo modo è la logica a decidere quale pagine inserire nel DOM e in caso in cui l'utente inavvertitamente effettui un refresh della pagina, essendo lo stato salvato nelle sessioni e propagato al javascript, la logica sarà in grado di ripristinare lo stato della pagina. Altro vantaggio derivante da questa architettura riguarda il fatto che tutti i dati passano attraverso il javascript e arrivano al php solo con richieste POST e GET, in questo modo non sono presenti querystring e viene stroncato sul nascere il problema dell'SQLInjection.
Però la logica del sistema non è implementata in php ma in javascript, quindi l'uso delle sessioni in php è solo uno strumento in favore della logica: infatti le pagine Index.php, Activate.php e User.php inviano queste variabili di sessione al javascript richiamando la funzione *init($param,$page)* che si occupa di creare le corrispondenti funzioni per la creazione del DOM in base alla combinazione parametro-pagina:

```
function init(param,page){
  switch (param) {
```

```
case "viewCV":
  if (page=='user'){
    // Imposta la larghezza della parte sinistra e destra
    setWidth('230','769');
    // Visualizzazione del CV
    ViewCV();
    // Barra degli strumenti dell'utente
    putUserBar();
  }else{
    location.href="index.php"
  }
break;
case "activate":
  if (page=='activate'){
    // Imposta la larghezza della parte sinistra e destra
    setWidth('200','799');
    // Attivazione utente
    Activate();
    // Mette il login sulla sinistra
    putLogin();
  }else{
    location.href="index.php"
  }
break;
case "anag":
  if (page=='user') {
    // Imposta la larghezza della parte sinistra e destra
    setWidth('230','769');
    // Modifica prima parte anagrafica
    canAnagrafica('user');
    // Barra degli strumenti dell'utente
```

```
    putUserBar();
    }else{
      // Imposta la larghezza della parte sinistra e destra
      setWidth('200','799');
      // Inserimento prima parte anagrafica
      canAnagrafica('registration');
      // Mette il login sulla sinistra
      putLogin();
    }
  break;
  ...
  default:
    // Imposta la larghezza della parte sinistra e destra
    setWidth('250','749');
    // Inserimento nuova registrazione
    newRegistration();
    // Mette il login sulla sinistra
    putLogin();
  }
}
```

Come si può vedere dal codice, la funzione init si occupa di gestire quale funzione chiamare al momento dell'avvio della pagina (sia nel caso del primo accesso che in quello di un eventuale refresh), le funzioni che vengono chiamate sono racchiuse all'interno degli altri file .js contenuti nella cartella controller.
Nel caso in cui la corrispondenza parametro/pagina fosse sbagliata, il sistema ripristinerebbe la configurazione di default composta dal login utente e l'inizio di una nuova registrazione.

3.4.2 Inserimento candidatura

L'inserimento della candidatura è regolato da i seguenti file .js: libRegistration.js, libAnagrafica.js, libAnagrafica2.js, libFormazione.js, libEspeLav.js, libConoscenze.js e libInteressi.js.
Ognuno di tali file contiene le seguenti tipologie di funzioni:

- una funzione che inserisce il DOM riguardo la specifica sezione da inserire
- una serie di funzioni che permettono una gestione dinamica dell'inserimento
- una funzione di controllo dei dati inseriti
- una funzione di salvataggio dei dati inseriti

Lo schema dei precedenti file è quindi sempre lo stesso: la prima funzione inserisce nel DOM della pagina il form vero e proprio della pagina di inserimento della candidatura, tale form a volte viene popolato se l'utente è tornato nella sezione, caso che si verifica sia quando l'utente durante la prima registrazione torna ad una sezione precedente e sia quando l'utente modifica tale sezione da dentro la gestione utente.
La compilazione dei campi non è mai statica, infatti l'utente con determinate scelte può far comparire a scomparire determinati campi e la gestione della dinamicità dei suddetti campi è affidata ad una serie di funzioni presenti all'interno di tali file:

```
// Funzione per l'aggiunta delle lingue
var numlingua=3;
function addLingua(){
  if (numlingua==3){
    document.getElementById("lingua"+numlingua).
            setAttribute("style","display: yes");
    document.getElementById("remLinguahref").
```

```
            setAttribute("style","display: yes");
    numlingua++;
  }
  else if (numlingua==4){
    document.getElementById("lingua"+numlingua).
            setAttribute("style","display: yes");
    numlingua++;
  }
  else if (numlingua==5){
    document.getElementById("lingua"+numlingua).
            setAttribute("style","display: yes");
    numlingua++;
    document.getElementById("addLinguahref").
            setAttribute("style","display: none");
  }
}
```

Nell'esempio precedente viene mostrata la funzione che aggiunge dinamicamente le lingue nella sezione delle conoscenze: quando l'utente clicca sul pulsante *Aggiungi Lingua* viene richiamata la funzione addLingua() che controlla il numero di lingue presente e in base a tale valore decide quale div attivare o meno (nel caso in cui si arrivi all'ultima lingua disponibile, cioè la quinta, allora il pulsante viene disattivato).
Altro tipo di dinamicità è data dalla comparsa di ulteriori campi da inserisce in base a determinate scelte degli utenti. Ad esempio se l'utente nella prima sezione dell'anagrafica, sceglie il valore *Richiesta di stage* nel campo *Tipologia di candidatura*, allora sarà obbligato ad inserire un campo di testo dove deve specificare per cosa si vuole proporre all'azienda. Il codice associato alla selezione di tale voce è il seguente:

```
var divdisplay=0;
function altroTextPop(){
  if (divdisplay==0){
```

```
    document.getElementById("altroCan").
      setAttribute("style","display: yes");
    divdisplay=1;
  }
}
function altroTextOut(){
  if (divdisplay==1){
    document.getElementById("altroCan").
      setAttribute("style","display: none");
    divdisplay=0;
  }
}
```

Il codice è composto da due funzioni: la prima che mostra il campo di testo e la seconda che lo rimuove e tale decisione viene effettuata in base al valore della variabile *divdisplay*, che inizialmente assume il valore 0 (cioè il campo non è visibile). In realtà si poteva realizzare un'unica funzione, ma per questioni di comodità e per lasciare possibili sviluppi nella gestione di tale dinamicità del sistema, si è scelto di separare tale funzionalità in due funzioni separate.

Nel momento in cui l'utente prova a proseguire alla sezione successiva (o a salvare le modifiche ai dati se si trova nella gestione utente), viene attivata la funzione di controllo dei dati inseriti: tale funzione genera un alert in javascript che avvisa quali campi sono mancanti o sbagliati. Nel caso in cui non siano presenti errori, la funzione di controllo porta direttamente alla funzione di salvataggio: tale funzione prende i valori inseriti (non deve effettuare nessun controllo perchè è già stato fatto) e li spedisce con una chiamata Ajax al back_end, una volta terminato il salvataggio, in base al valore della variabile page, indirizza l'utente o verso la prossima sezione dell'inserimento della candidatura o verso la stessa pagina se l'utente è nella Gestione Utente. A tilolo di esempio, il codice che si occupa di effettuare la chiamata Ajax per il salvataggio degli interessi e delle disponibilità di una candidatura è il

seguente:

```
function toFine(type){
  // Prendi i valori
  var parametriScript="";
  //INTERESSI
  var interessi1 = document.newRegistration.interessi1.value;
  var interessi2 = document.newRegistration.interessi2.value;
  var interessi3 = document.newRegistration.interessi3.value;
  var interessi4 = document.newRegistration.interessi4.value;
  // Disponibilità
  var disp1 = document.newRegistration.disp1.checked;
  var disp2 = document.newRegistration.disp2.checked;
  var disp3 = document.newRegistration.disp3.value;
  var altrodisp3 = document.newRegistration.altroDisText.value;
  parametriScript ="interessi1="+interessi1+"&interessi2="+interessi2+
  "&interessi3="+interessi3+"&interessi4="+interessi4+"&disp1="+disp1+
  "&disp2="+disp2+"&disp3="+disp3+"&altrodisp3="+altrodisp3;
  // Chiamata Ajax che salva i dati
  var xmlHttp = assegnaXMLHttpRequest();
  xmlHttp.open("POST",pathModel+'saveInteressi.php',true);
  xmlHttp.setRequestHeader('Content-Type',
    'application/x-www-form-urlencoded');
  xmlHttp.send(parametriScript);
  xmlHttp.onreadystatechange=function(){
    if(xmlHttp.readyState==4){
      if(xmlHttp.status == 200){
        if (type=='registration') Fine();
        else canInteressi('user');
      }
    }
  }
```

```
  var stringa;
  stringa = "<div style=\"text-align: center; margin-left: 40px;\">
    <img src=\"images/loading.gif\" />";
  document.getElementById('right').innerHTML=stringa;
}
```

La funzione quindi semplicemente prende i valori del form di inserimento e li spedisce attraverso la variabile *parametriScript* al file *saveInteressi.php*, quando tale file risponderà correttamente al file javascript allora in base alla variabile *type* (variabile globale che indica in quale pagina il sistema si trova) il javascript deciderà quale pagina visualizzare; nel mentre viene mostrata un'immagine di loading.
Il codice della funzione *saveInteressi.php* è il seguente:

```
<?
@session_start();
$id=$_SESSION['idcand'];
require('dati.php');
$interesse1=$_POST['interessi1'];
$interesse2=$_POST['interessi2'];
$interesse3=$_POST['interessi3'];
$interesse4=$_POST['interessi4'];
// Interessi
if ($interesse2 != ""){
  $result = explode("****", java_context()->getServlet()->
    saveInteresse($id,$interesse2,"due"));
  if ($result[0]=="error"){
    echo "ERRORE: ".$result[1];
  }else{
    echo "INSERT/UPDATE interesse 2 effettuato correttamente\n";
  }
}else{
  $result = explode("****", java_context()->getServlet()->
```

```
    removeInteresse($id,"due"));
if ($result[0]=="error"){
    echo "ERRORE: ".$result[1];
  }else{
    echo "REMOVE interesse 2 effettuato correttamente\n";
  }
}
...
// Disponibilità
$disp1=$_POST['disp1']; if ($disp1=="false") $disp1=0; else $disp1=1;
$result = explode("****", java_context()->getServlet()->
  updateFieldCandidatura($id,'raggiungimento',$disp1));
if ($result[0]=="error") echo "ERRORE: ".$result[1]; else
  echo "UPDATE disp1 effettuato correttamente\n";
$disp2=$_POST['disp2']; if ($disp2=="false") $disp2=0; else $disp2=1;
$result = explode("****", java_context()->getServlet()->
  updateFieldCandidatura($id,'trasferirsi',$disp2));
if ($result[0]=="error") echo "ERRORE: ".$result[1]; else
  echo "UPDATE disp2 effettuato correttamente\n";
...
// Il primo interesse è per ultimo perchè aggiorna la sessione
if ($interesse1 != ""){
$result = explode("****", java_context()->getServlet()->
  saveInteresse($id,$interesse1,"uno"));
  if ($result[0]=="error"){
    echo "ERRORE: ".$result[1];
  }else{
    echo "INSERT/UPDATE interesse 1 effettuato correttamente\n";
    $_SESSION['next']="fine";
  }
}
```

```
?>
```

Tale file quindi prende i valori che gli sono stati mandati via POST attraverso la chiamata Ajax e richiamando le funzioni del back_end effettua i salvataggi. Ad ogni fase della registrazione quindi viene aggiornata la variabile di sessione *next* (per gestire il refresh).

3.4.3 Gestione utente

La Gestione Utente è gestita da un unico file di nome libUser.js, tale file si occupa di mostrare il menù utente:

- cambio password
- visualizza CV
- modifica anagrafica 1° parte
- modifica anagrafica 2° parte
- modifica formazione
- modifica esperienze lavorative
- modifica conoscenze
- modifica interessi
- modifica file

La prima funzionalità è banalmente implementata con un form di richiesta della vecchia password e di quella nuova (in duplice copia per evitare errori di digitazione).
La seconda funzionalità invece è sviluppata attraverso diverse chiamate Ajax: l'intero CV è mostrato all'interno di un unico generale DIV suddivisiso al suo interno da sei div, uno per ogni parte della candidatura; le chiamate Ajax sono effettuate verso il file GetDati.php del model che permette con

un parametro di richiamare solo alcuni dati a partire da un id_candidatura. Quindi il CV è popolato dinamicamente e in modo asincrono: tale scelta è stata effettuata in vista di un futuro sviluppo in cui sarà possibile modificare il CV direttamente dalla sua visualizzazione, modificando solo una parte dei dati interessati e ricaricando solo tale parte, in modo che il restate CV non debba essere ricaricato nuovamente.
A titolo di esempio viene mostrata il caricamento degli interessi legati ad una candidatura:

```
function loadInte(){
  var xmlHttp = assegnaXMLHttpRequest();
  xmlHttp.open("POST",pathModel+'GetDati.php',true);
  xmlHttp.setRequestHeader('Content-Type',
    'application/x-www-form-urlencoded');
  parametriScript = "type=interessi&save=no";
  xmlHttp.send(parametriScript);
  xmlHttp.onreadystatechange=function(){
    if(xmlHttp.readyState==4){
      if(xmlHttp.status == 200){
        var stringa;
        stringa = "<br/><br/><br/><h2>Interessi</h2>";
        var splittedInteressi = xmlHttp.responseText.split("????");
        if (splittedInteressi[0].replace(/\s+$|^\s+/g,"")!= ""){
          var splittedInteresse = splittedInteressi[0].split("****");
          stringa=stringa+'<b>Interesse 1:</b>
            '+splittedInteresse[1]+'<br/><br/>';
        }
        if (splittedInteressi[1]!= null){
          var splittedInteresse = splittedInteressi[1].split("****");
          stringa=stringa+'<b>Interesse 2:</b>
             '+splittedInteresse[1]+'<br/><br/>';
        }
```

```
        if (splittedInteressi[2]!= null){
          var splittedInteresse = splittedInteressi[2].split("****");
          stringa=stringa+'<b>Interesse 3:</b>
            '+splittedInteresse[1]+'<br/><br/>';
        }
        if (splittedInteressi[3]!= null){
          var splittedInteresse = splittedInteressi[3].split("****");
          stringa=stringa+'<b>Interesse 4:</b>
            '+splittedInteresse[1]+'<br/><br/>';
        }
        document.getElementById('datiInte').innerHTML=stringa;
      }
    }
  }
  var stringa;
  stringa = "<div style=\"text-align: center; margin-left: 40px;\">
      <img src=\"images/loading.gif\" />";
  document.getElementById('datiInte').innerHTML=stringa;
}
```

Come descritto precedentemente, viene effettuata una chiamata Ajax al file GetDati.php, passando come parametro 'interessi', il quale serve a specificare quali dati si vogliono ottenere della candidatura; una volta pronto, la forma dei dati è la seguente:

id_interesse1****interesse????id_interesse2****interesse????
id_interesse3****interesse????id_interesse4****interesse

quindi una forma composta da una coppia (id_interesse,testo_interesse) ripetuta per il numero di interessi (che sono sempre fissi uguali a quattro).
La sezione del file GetDati.php che si occupa di restituire i valori degli itneressi è la seguente:

```
...
// Prende gli interessi
if ($type=='interessi'){
  $resultString="";
  $resultStringError="";
  $result = explode("????", java_context()->getServlet()->
    getInteressi($id));
  if ($result[0]=="error"){
    $resultStringError = "ERRORE: ".$result[1];
  }else{
    if ($result[0]=="ok"){
      for ($contatore=1; $contatore < count($result); $contatore++){
          $resultString=$resultString.$result[$contatore];
          if ($contatore<count($result)-1)
            $resultString=$resultString."????";
      }
    }
  }
  // Disponibilità
  $result = explode("****", java_context()->getServlet()->
    getFieldCandidatura($id,"raggiungimento"));
  if ($result[0]=="error") $resultStringError = "ERRORE: ".$result[1];
  else $resultString=$resultString."##raggiungimento=".$result[1];
  $result = explode("****", java_context()->getServlet()->
    getFieldCandidatura($id,"trasferirsi"));
  if ($result[0]=="error") $resultStringError = "ERRORE: ".$result[1];
  else $resultString=$resultString."##trasferirsi=".$result[1];
  $result = explode("****", java_context()->getServlet()->
    getFieldStringCandidatura($id,"trasferimento"));
  if ($result[0]=="error") $resultStringError = "ERRORE: ".$result[1];
  else $resultString=$resultString."##trasferimento=".$result[1];
```

```
  if ($save!="no") $_SESSION['next']="inte";
  if ($resultStringError!=""){
    echo $resultStringError;
  }else{
    echo $resultString;
  }
}
...
```

L'ultima parte della Gestione Utente consiste in una serie di link per la modifica delle singole parti della candidatura, tali link nascondono chiamate alle stesse funzioni di inserimento della candidatura, che essendo generali permettono di inserire il form con i dati da, in questo caso, modificare. In base poi alla pagina, in questo caso User, una volta che si modificheranno i dati si verrà riportati alla stessa pagina di modifica con i dati nuovi salvati.

3.4.4 Chiamata Ajax

In questa ultima sezione viene semplicemente riportato il codice per effettuare la chiamata Ajax ai file della cartella model. Tale funzione è compatibile con tutti i browser specificati nella progettazione.

```
function assegnaXMLHttpRequest() {
  var XHR = null;
  var browserUtente = navigator.userAgent.toUpperCase();
  if(typeof(XMLHttpRequest) === "function" ||
      typeof(XMLHttpRequest) === "object") {
    XHR = new XMLHttpRequest();
  } else if (window.ActiveXObject &&
        browserUtente.indexOf("MSIE 4") < 0) {
    if (browserUtente.indexOf("MSIE 5") < 0) {
      XHR = new ActiveXObject("Msxml2.XMLHTTP");
    } else {
```

```
        XHR = new ActiveXObject("Microsoft.XMLHTTP");
      }
    }
    return XHR;
}
```

Capitolo 4

Esempi di casi d'uso

In questo capitolo vengono presentati i due casi d'uso principali del sistema: l'inserimento della candidatura e la gestione della candidatura da parte dell'utente.
I casi d'uso sono diversi e molteplici, possono portare a diverse situazioni anche di difficile gestione: un utente che inserisce solo metà candidatura, un utente che inserisce una candidatura senza attivarla e poi ne inserisce una seconda, un utente che mentre inserisce una candidatura vuole tornare indietro a modificare dei dati, utenti che non attivano mai le candidatura e molti altri ancora.
Per semplicità però verranno mostrate alcune schermate di come il sistema si comporta nei due casi d'uso 'normali' che coprono il 90% dell'uso del sistema.

4.1 Inserimento della candidatura

4.1.1 Inizio registrazione e anagrafica

L'inserimento della candidatura comincia nell'homepage del sito (la pagina Index.php), nella parte destra della pagina sotto la dicitura *Inizia una nuova registrazione*, l'utente può cominciare ad inserire la sua candidatura, in questa fase vengono richiesti il nome, il cognome, l'email e la password, inoltre per evitare che il form sia utilizzato da un BOT viene richiesto di inserire il codice CAPTCHA mostrato a video.
Il sistema durante l'inserimento dei dati, effettua diversi controlli in tempo reale:

- controllo che la password inserita abbia un minimo di complessità (per evitare password banali)
- controllo che le due password siano uguali
- controllo che l'email sia della forma corretta (indirizzo@provider.estensione)
- controllo che l'email non sia già presente nel sistema
- controllo che le due email siano uguali
- controllo che il codice CAPTCHA inserito sia corretto

Tutti questi controlli sono effettuati in tempo reale mentre l'utente inserisce i dati: in questo modo il pulsante *Registrati* permetterà di proseguire solo ed esclusivamente quando tutti i controlli saranno sistemati. Nel caso in cui i controlli siano positivi, la registrazione comincia: inizialmente i dati inseriti vengono già salvati nel database in modo che se l'utente chiude il browser e riaccede può continuare l'inserimento della candidatura (deve ovviamente effettuare il login con l'indirizzo email e la password inseriti nella prima schermata).
La schermata successiva è la prima pagina di inserimento della candidatura, la prima parte dell'anagrafica.

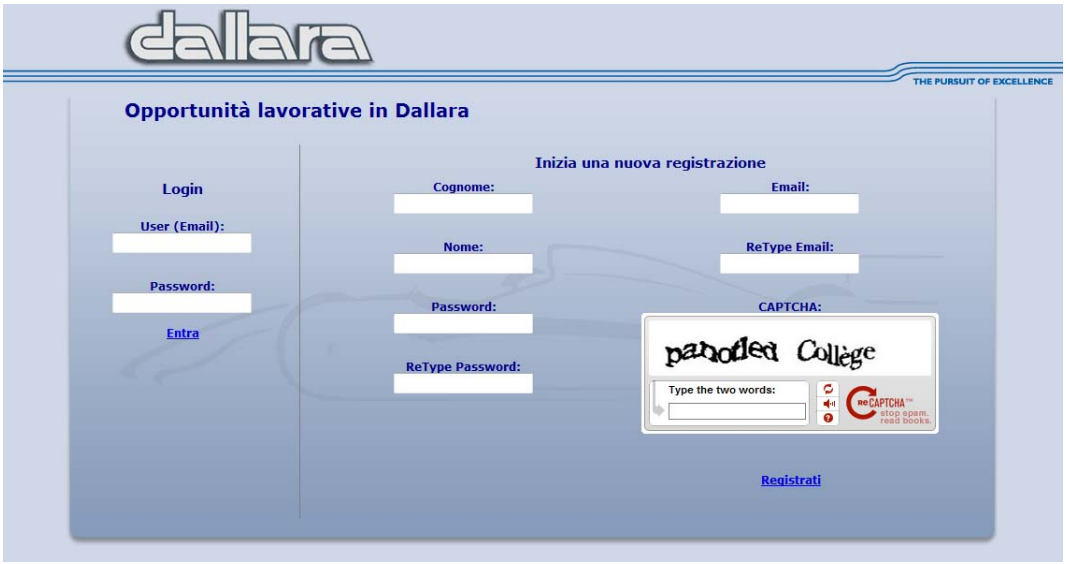

Figura 4.1: Pagina iniziale

Nel passaggio dalla pagina iniziale alla pagina di inserimento della candidatura, il layout dell'intera pagina subisce una leggera trasformazione, in quanto il login sulla sinistra viene ridimensionato in larghezza in modo da lasciare maggior spazio al form di inserimento.
Viene inoltre aggiunta una barra in alto che indica l'avanzamento della candidatura: tale barra è composta da un link a sinistra per l'annullamento dell'inserimento, ed una freccia che indica quale tipologia di dati si stanno inserendo; sono anche mostrate le prossime fasi che saranno richieste con una scritta più leggera e piccola rispetto alla sezione che si sta inserendo, inoltre procedendo nelle fasi, le scritte precedenti diventano dei link per tornare a modificare dati già inseriti, nel caso in cui l'utente si accorga di avere compiuto errori nell'inserimento. La prima sezione dell'anagrafica contiene campi che sono già stati inseriti nella prima fase (nome, cognome ed email) e quindi vengono riportati in automatico.
Il campo *Data di nascita* non è editabile, in quanto per evitare errori di forma è necessario usare l'icona al fianco del campo che permette di selezionare date attraverso un apposito calendario in javascript.

Figura 4.2: Messaggi di errore durante l'inserimento

Figura 4.3: Stato avanzamento dell'inserimento della candidatura

In questa prima sezione, la dinamicità della pagina è dati dai campi *Come è entrato in contatto con dallara*, *Tipologia candidatura* e *Categoria protetta*: se nel menù a tendina di *Come è entrato in contatto con dallara* si seleziona la voce *Altro* allora comparirà dinamicamente un campo di testo per specificare la voce; per quanto riguarda la *Tipologia candidatura* invece il campo di testo per la specifica comparirà se si selezionano le voci *Richiesta di Stage*, *Richiesta di Tirocinio Formativo*, *Richiesta di Tesi*, *Richiesta di Alternanza scuola/lavoro*. Infine se si seleziona il flag *L. 68/99* allora comparirà un ulteriore campo da specificare riguardo la percentuale di invalidità, mentre se si seleziona *Art. 18 Comma 2*, verrà in automatico selezionato anche *L. 68/99*

Figura 4.4: Campo di testo dinamico

ma il campo di testo della percentuale di invalidità non sarà più richiesto.

Nel caso in cui l'utente volesse proseguire nell'inserimento senza aver specificato i campi obbligatori, il sistema mostra un avviso su quali campi bisogna ancora inserire, come in figura (4.5).

Una volta che tutti i campi obbligatori sono stati inseriti, il sistema procederà con il salvataggio dei dati inseriti e con il mostrare la successiva sezione di dati da inserire (in tal caso la seconda parte di anagrafica).

Nella seconda parte dell'anagrafica vengono richiesti gli indirizzi di residenza e di domicilio e le eventuali patenti possedute. Per agevolare il processo di inserimento è presente un link per copiare i dati dalla residenza al domicilio.

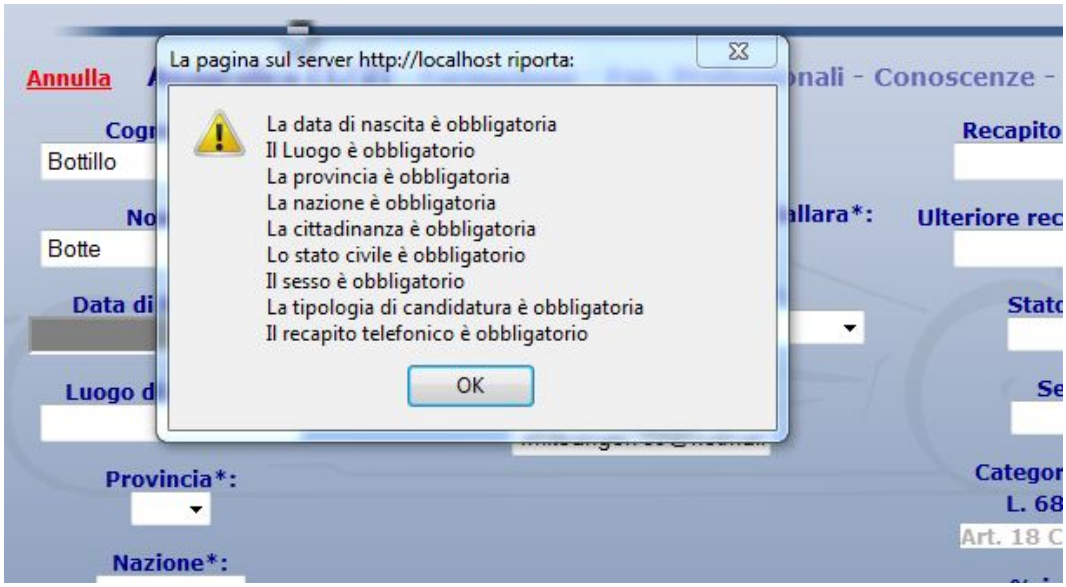

Figura 4.5: Notifica errori in inserimento

4.1.2 Formazione

La sezione della Formazione è la sezione più complicata e lunga da completare: sono presenti cinque aree opzionali più una obbligatoria, le aree opzionali corrispondono a:

- Phd/Master
- Attività formative/tirocinio formativi/tesi in azienda
- Laurea 5 anni
- Laurea 3+2 anni
- Laurea 3 anni

L'area obbligatoria è quella del diploma.
Per migliorare l'usabilità e la facilità dell'inserimento dei dati, evitando quindi di mostrare circa 60 campi di testo da inserire, ogni area opzionale inizialmente viene mostrata chiusa, mostrando esclusivamente i campi del diploma:

Figura 4.6: Seconda parte dell'anagrafica

se l'utente poi ha conseguito una delle aree opzionali selezionando tale area compariranno dinamicamente i campi relativi all'area; tutte le aree meno *Attività formative/tirocinio formativi/tesi in azienda* hanno a loro volta una comparsa dinamica di campi nel caso in cui l'area che si sta inserendo è stata conseguita o meno, in quanto vengono mostrati dell'*Anno di conseguimento*, *Votazione*, *Titolo e argomento della tesi*.

Nel caso in cui l'utente si accorga di aver sbagliato, deselezionando l'area non verrà aggiunta alla sua formazione, ma i dati all'interno dei campi di testo rimangono salvati in modo da averli già pronti se l'utente selezionerà nuovamente l'area.

4.1.3 Esperienze lavorative

In questa sezione le esperienze lavorative, come nella sezione precedente, vengono richieste solo ed esclusivamente se l'utente seleziona il relativo flag. Le date dell'esperienze lavorative vengono inserite, come nella prima fase dell'anagrafica, esclusivamente con i calendari javascript che aggiungono dinamicamente le date nel formato corretto.

Per quanto riguarda le voci *Settore*, *Area* e *Tipologia contrattuale*, se si

Figura 4.7: Formazione

seleziona la voce *Altro* compaiono dinamicamente dei campi di testo per la specifica da parte dell'utente di tale campo.
Come la sezione precedente, se si deseleziona un'esperienze lavorativa non verrà salvata insieme alla candidatura, ma i dati già inseriti saranno mantenuti in modo da non doverli reinserire se l'utente seleziona nuovamente tale esperienza.

4.1.4 Conoscenze

La sezione *Conoscenze* è suddivisa in tre parti:

- Conoscenze linguistiche
- Conoscenze informatiche
- Esperienze di pista/SAE

Per ogni lunga è richiesto l'inserimento della lingua e della conoscenza riguardo il parlato e lo scritto, è possibile aggiungere (e rimuovere) dinamicamente

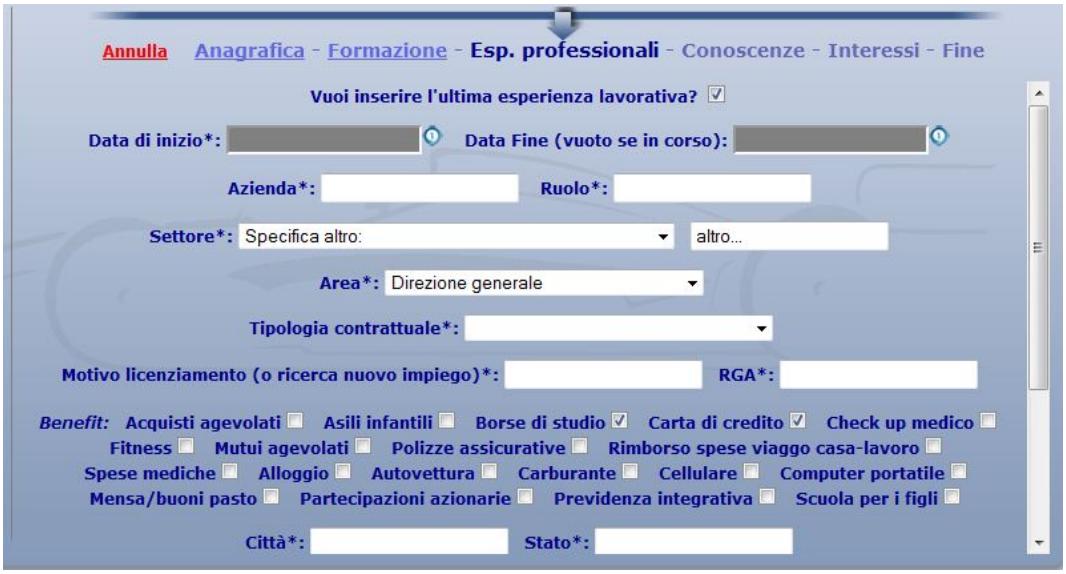

Figura 4.8: Esperienze lavorative

fino a 5 lingue.

Le conoscenze informatiche sono semplicemente la spunta di alcuni flag. Per quanto riguarda la esperienze di pista e SAE, il sistema è simile a qullo delle esperienze lavorative: se l'utente ha una di queste due esperienze, selezionandole vengono mostrati i campi da inserire.

4.1.5 Interessi e Fine

Le ultime due sezioni sono le più brevi e facili da completare: nella sezioni *Interessi* l'utente deve selezionare esclusivamente fino a quattro interesse da un menù a tendina e specificare eventuali disponibilità a trasferte e/o trasferimento, l'unica dinamicità del form è data dalla voce *Disponibilità di trasferirsi all'estero*, dove se si specifica *Altro* viene richiesto di specificare il paese.

Nella sezione *Fine*, l'utente deve semplicemente opzionalmente effettuare l'upload di un CV digitale ed accettare l'informativa sulla privacy. Una volta che l'utente accetta l'informatica e prosegue, la candidatura viene completata

Figura 4.9: Conoscenze

e messa in stato *da attivare*: il sistema quindi genera un codice di attivazione che viene spedito via email all'utente, quando l'utente inserisce il codice corretto nella schermata dell'attivazione si considera la candidatura salvata nel sistema.

Figura 4.10: Interessi

4.2 Gestione Utente

La *Gestione Utente* è raggiungibile esclusivamente dopo che l'utente effettua l'attivazione della candidatura, in tale sezione l'utente può modificare la propria password, visualizzare la propria candidatura e modificarla.
In questo capitolo vediamo due casi d'uso: l'utente che vuole visualizzare la propria candidatura e l'utente che vuole modificare la sezione *Esperienze lavorative* della sua candidatura.

4.2.1 Visualizzazione candidatura

Una volta che l'utente effettua il login correttamente, viene portato in una sezione dedicata alle sue funzionalità: viene mostrata una barra di funzioni sulla sinistra e un messaggio di benvenuto al centro della pagina.
Nella barra di funzioni posizionata sulla sinistra, l'utente può accedere alla visualizzazione della candidatura attraverso il link *Visualizza CV*, cliccando su tale link infatti il sistema posizionerà al centro della pagina l'intera can-

Figura 4.11: Attivazione

didatura: Tutte le informazioni sono quindi raccolte in un'unica pagina di

Figura 4.12: Visualizzazione CV

consultazione, per non modificare il layout della pagina, la candidatura è inserita all'interno di un riguadro che è possibile scorrere con l'apposita barra di scorrimento sulla destra.

4.2.2 Modifica Esperienze Lavorative

Nella barra di funzioni sulla sinistra sono inoltre presenti una serie di link per la modifica delle varie sezioni della candidatura; per semplicità viene riportata solo la modifica delle esperienze lavorative, in quanto le restanti sezioni sono semanticamente identiche.

Cliccando sul link *Modifica esp. lavorative*, il sistema porterà l'utente in

Figura 4.13: Modifica esperienze lavorative

un form per la modifica di tale esperienze lavorative: il form in questione è **identico** al form di inserimento, infatti essendo i dati e la forma dell'inserimento e della modifica identici, viene riutilizzato il codice dell'inserimento; a differenza dell'inserimento però non viene mostrata nessuna barra in alto dello stato, in quanto senza più valenza e inoltre cliccando sul tasto *Modifica* non si viene portati a nessun'altra sezione successiva ma semplicemente nuovamente alla modifica dei dati, in modo che l'utente possa controllare se effettivamente i dati sono stati modificati o meno.

Le restanti sezioni di modifica sono esattamente uguali alla modifica delle esperienze lavorative: una volta effettuate le modifiche si viene riportati nella

stessa sezione e non è presente la barra di avanzamento.
L'anagrafica è spezzata in due sezioni di modifica separate.

Capitolo 5

Conclusioni

Il periodo di collaborazione con la Dallara Automobili è durato in tutto quattro mesi, da aprile 2010 a luglio 2010: inizialmente si sono svolti alcuni incontri per la definizione del lavoro da eseguire, successivamente gli incontri sono stati effettuati per la definizione dei requisiti e delle specifiche tecniche; gli ultimi due mesi invece sono stati ad appannaggio dello sviluppo vero e proprio.
Lo sviluppo ha coinvolto sia, ovviamente, il mio sforzo che quello della Dallara Automobili in quanto la predisposizione della macchina di sviluppo e del back_end è stata eseguita dai loro tecnici. Infatti il back_end in java era un componente già presente all'interno dell'infrastruttura Dallara ed è stato semplicemente riutilizzato per questo lavoro di tesi. Quindi il back_end è stato installato e configurato dai loro tecnici come il relativo progetto in Eclipse per la scrittura dei metodi, scrittura che è stata effettuata completamente da me insieme al front_end.

5.1 Stato attuale del software

Lo sviluppo del software al momento della stesura di questa tesi di laurea, è arrivato più o meno al 60%: è stata completato l'inserimento della candidatura e la sezione della Gestione Utente, come da specifica manca quindi

tutto lo sviluppo della parte riguardo la Gestione Amministrazione.
Ovviamente nell'implementazione di tale ultima sezione, le prime due saranno a loro volta leggermente da modificare (eg. lo status delle candidature).

5.2 Sviluppi futuri

Come primo sviluppo futuro non si può considerare l'implementazione della *Gestione Amministratore*: tale sezione presupporà l'introduzione di una serie di account particolari che dopo la fase di login porteranno alla nuova sezione *Gestone Amministratore*, in tale sezione si dovrà permettere la modifica della password, la creazione di nuovi account amministrativi, la visualizzazione delle candidature, la ricerca e il salvataggio (una sorta di carrello delle candidature).
L'implementazione di tale sezione può ricorrere ad alcune parte di codice già implementate: la modifica della password e la visualizzazione delle candidature infatti sono elementi già presenti nella sezione *Gestione Utente* che possono essere facilmente riproposti in tale sezione.
Le funzionalità preponderante di tale area è sicuramente la ricerca: infatti, come da progettazione, l'amministratore deve essere in grado sia di effettuare ricerche libere tra tutti i campi sia di effettuare ricerche mirate per **ogni** singolo campo che l'utente ha inserito durante la fase di registrazione, come tale questa tipo di ricerca è sicuramente il fulcro più grosso e impegnativo della sezione di amministrazione.
Per quanto riguarda altri sviluppi futuri oltre la progettazione già avvenuta, sicuramente bisognerebbe includere un aumento dell'intelligenza del sistema sia durante l'inserimento della candidatura che durante la visualizzazione: durante l'inserimento della candidatura sarebbe sicuramente utile migliorare l'usabilità, predisponendo messaggi informativi ed inserendo messaggi di avviso sulla tipologia dei dati che bisogna inserire, infatti adesso si controlla solamente se un campo obbligatorio è stato inserito o meno, ma non si controlla se l'inserimento rispetti la semantica di quel campo; sarebbe in-

oltre naturale permettere una navigazione tra le aree dell'inserimento: infatti attualmente è possibile inserire i dati, tornare indietro ad una sezione precedente, ma non è possibile tornare all'ultima sezione che si stava inserendo, bisogna necessariamente passare da tutte le sezioni già inserite.
Nella visualizzazione invece, un possibile sviluppo sarebbe quello di permettere la modifica al volo dei dati che si stanno visualizzando: il sistema è già impostato in modo che il caricamento del CV è separato per ogni area, in questo modo durante la visualizzazione se si modifica per esempio un campo delle conoscenze, è possibile ricaricare solamente l'area delle conoscenze e non tutta l'intera pagina della candidatura.
Ovviamente come sviluppo del sistema bisogna includere anche un rifacimento grafico, essendo questa una prima versione dove si è puntato maggiormente nello sviluppo e nella logica e poco sull'interfaccia.

5.2.1 Ulteriori punti di accesso

Il sistema è predisposto per accettare ulteriori punti di accesso oltre l'interfaccia web classica: non è da escludere pensare ad una possibile mobile web application che permetta di accedere ai dati, visualizzarli e modificarli direttamente da un qualsiasi browser web moderno per smartphone; al tempo stesso si potrebbe prendere in considerazione lo sviluppo di applicazioni native per sistemi operativi come iOS, Rim e Android.
Un ulteriore punto di accesso sarebbe quello derivante l'integrazione di tali dati con il sistema informativo presente in Dallara Automobili, in modo da unificare il punto di accesso degli amministratori (che sono i dirigenti e il personale in Dallara).

In conclusione, tale lavoro di laurea ha portato alla creazione di un software per la ricezione di dati curriculari per candidati che vogliono lavore in Dallara Automobili. Il lavoro è stato suddiviso in due fase principali: una di progettazione e una di sviluppo; la fase di progettazione è stata completata mentre quella di sviluppo è in via di completamento.

Capitolo 6

Ringraziamenti

Un capitolo di ringraziamenti è necessario per esprimere la mia graditudine verso chi mi ha aiutato durante questi sei anni di studio, di difficoltà e di apprendimento. Penso che per affrontare qualsiasi corso di laurea, sia necessario un supporto costante da parte di amici e famiglia, senza tale supporto infatti, oltre alle difficoltà tecniche dello studio si presentano difficoltà organizzative ed emotive che pregiudicano la buona riuscita del proprio corso di laurea. Per questo ringrazio:

- la mia famiglia per avermi dato la possibilità di iscrivermi all'Università
- il professore Roccetti Marco per l'aiuto e l'occasione datami di confrontarmi con una realtà esterna come la Dallara Automobili
- tutte le persone che ho conosciuto in Dallara Automobili che sono state disponibili per completare il mio percorso di tesi
- Serena per avermi dato la forza di concludere il mio percorso di laurea
- i miei amici per il giusto svago da alternare allo studio
- tutte le persone che nel bene e nel male mi hanno aiutato

Infine un grazie verso tutti i professori che hanno contribuito a formarmi durante questo corso di studi e permesso di raggiungere questo ambito traguardo.

Grazie.

Bibliografia

[1] TIM O'REILLY, 30/09/2005. What is web 2.0 Design Patterns and Business Model for the Next Generation of Software. *O'Reilly Media*, 20

[2] Morales-Chaparro, Linaje, Preciado, Sanchez-Figueroa. MVC Web design patterns and Rich Internet Applications, Software *Engineering Group, Escuela Politécnica. Universidad de Extremadura*, 8

[3] Gustavo Rossi, Daniel Schwave, Robson Guimaraes, May 1-5 2001. Designing Personalized Web Applications, *ACM Computing*, 275 - 284

[4] Luciano Baresi, Franca Garzotto, Paolo Paolini, 2000. From Web Sites to Web Applications: New Issues for Conceptual Modelling, *Lecture Notes in Computer Science*, 89-100

[5] J. Sergio Zepeda, Sergio V. Chapa, 5-7 Settembre 2007. From Desktop Applications Towards Ajax Web Applications, *IEEE Computer Society*, 4

[6] Linda Dailey Paulson. Building Rich Web Applications with Ajax, *IEEE Computer Society*, 4

[7] Gino Roncaglia, 15-16 Ottobre 2009. Web 2.0 and the future of research: new tools for research networks, *Contemporary History in the Digital Age, Luxembourg*, 8

[8] Piero Fraternali, Settembre 1999. Tools and Apporaches for Developing Data-Intensive Web Applications: A Survey, *ACM Computing Surveys*, 227-263

[9] Inge van de Weerd, Sjaak Brinkkemper, Jurriaan Souer, Johan Versendaal, 2006. A Situational Implementation Method for Web-based Content Managament System-applications: Method Engineering and Validation in Practice, *Wiley InterScience*, 521-538

[10] Hans-W. Gellersen, Martin Gaedke, Febbraio 1999. Object-Oriented Web Application Development, *IEEE Computer Society*, 60-68

[11] Athula Ginige, San Murugesan, 2001. Web Engineering: An Introduction, *IEEE Computer Society*, 14-18

[12] Franca Garzotto, Paolo Paolini, Davide Bolchini, Sara Valenti, 2010. 'Modeling-by-Patterns' of Web Applications, *Lecture Notes in Computer Science*, 293-306

[13] Gustavo Rossi, Daniel Schwabe, Fernando Lyardet, 2010. Web Application Models Are More then Conceptual Models, *Lecture Notes in Computer Science*, 239-252

[14] Xiaojun Tan, Mu Zhou, Xiang Zuo, Yuyonh Cui, 2008. Integration WebGis with Ajax and XML Based on Google Maps. *IEEE Computer Society*, 4

[15] PHP Java/Bridge, *http://php-java-bridge.sourceforge.net/*

[16] Eclipse, *http://www.eclipse.org/*

[17] Notepad++, *http://notepad-plus-plus.org*