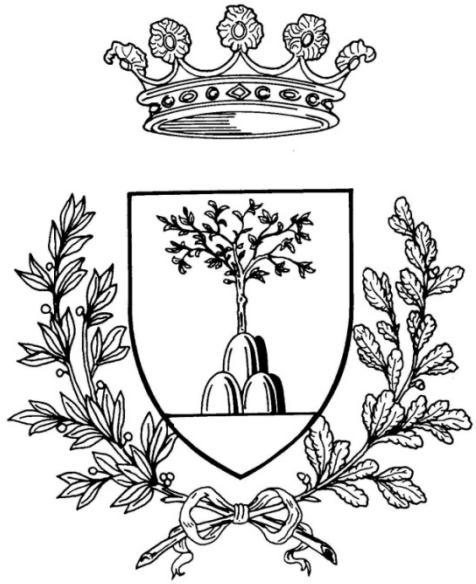

UNIVERSITÀ DEGLI STUDI DI FERRARA

FACOLTÀ DI INGEGNERIA

Corso di Laurea in Ingegneria dell'Informazione

Progetto e sviluppo di un'agenda web per dispositivi multitouch per il settore sanitario

Tesi di Laurea di:
Aguiari Fabiano

Relatore:
Chiar.mo Prof. Ing. Cesare Stefanelli

Correlatori:
Ing. Michele Tedeschi
Ing. Giovanni Foiani

Anno Accademico 2010-2011

Indice

Introduzione 3
Capitolo 1. Rich Internet Applications per il settore sanitario 5
1.1 Definizione di Rich Internet Applications 5
1.2 Usabilità 5
1.2.1 Definizione e requisiti 5
1.2.2 Navigabilità 6
1.2.3 Interfaccia grafica 7
1.3 Profilo tecnologico degli utenti finali 7
1.3.1 Compatibilità sui diversi browser 7
1.4 Progettazione per dispositivi multitouch 7
1.4.1 Usare interazioni “naturali” 8
1.5 Settore di riferimento 8
Capitolo 2. Analisi dei requisiti e progetto dell'applicazione 9
2.1 Requisiti 9
2.1.1 Ruoli e permessi 9
2.1.2 Funzionalità 10
2.2 Progetto 10
2.2.1 Schema del database 10
2.2.2 Struttura dell’applicazione 12
2.2.3 Layout e interazione 13
Capitolo 3. Tecnologie 15
3.1 Instant Developer 15
3.1.1 La programmazione relazionale 15
3.1.2 Il framework RD 18
3.1.3 Anatomia di un progetto In.de 20
3.1.4 Compilazione delle applicazioni 20
3.2 Macchina di sviluppo 20
3.2.1 Notebook touch-screen 21
Capitolo 4. Implementazione dell'applicazione 22
4.1 Connessione al database server 22
4.2 Login 23
4.3 Scelta ambulatorio e relativa attività specifica 24
4.4 Vista mese 26
4.5 Visualizzazione referti 27
4.6 Vista giorno 29
4.7 Vista mesi-giorni 29
Conclusioni 31
Riferimenti bibliografici 32

Introduzione

L'avvento di notebook multitouch, tablet e smartphone ha messo a disposizione degli sviluppatori software dispositivi hardware con caratteristiche che permettono la realizzazione di Rich Internet Applications ad alta interattività, aprendo nuove opportunità di crescita e sviluppo in un mondo che non ha ancora espresso tutte le sue potenzialità.
Questi nuovi dispositivi si affiancheranno ai tradizionali PC offrendo differenti modi di fruire i servizi informatici. I software attuali andranno rivisti e riprogettati per interpretare e soddisfare al meglio le nuove esigenze emergenti. Le aziende che sapranno intervenire fin da subito in questo senso ne trarranno i massimi benefici negli anni a venire. Non prestare attenzione a questo fenomeno e restare semplicemente in attesa favorirebbe solo i grandi produttori internazionali che hanno le risorse per riscrivere più e più volte le proprie applicazioni.
Questa tesi è il frutto di un tirocinio svolto presso Delta Informatica, azienda che sta cercando di affrontare queste difficoltà, tenendo comunque in considerazione il costo di mantenimento delle soluzioni attuali. La problematica affrontata è stata il progetto e lo sviluppo di una applicazione web per dispositivi multitouch. L'applicazione in questione è un'agenda per la consultazione di prenotazioni e di informazioni correlate destinata al settore sanitario, area di spicco dell'azienda che mi ha ospitato durante lo stage.
L'applicazione sviluppata offre spunti sui cambiamenti necessari da attuare nel progetto del funzionamento di applicazioni gestionali nell'ambito della sanità, in modo da sfruttare la semplicità e l'immediatezza di utilizzo dei dispositivi multitouch.
Infatti, un dispositivo mobile per essere tale deve avere le seguenti caratteristiche: essere portatile, quasi sempre a disposizione e utilizzabile all'istante e possedere un qualche tipo di connessione. Portatile significa che deve essere possibile portarlo in giro, ovunque e comunque senza particolari accorgimenti. Il fatto che sia portatile rende possibile avere il dispositivo mobile sempre a disposizione, in ogni momento e situazione della giornata. Un'altra caratteristica decisamente interessante è la possibilità di utilizzarlo all'istante e facilmente, anche stando in piedi e mentre si cammina, eliminando dunque la necessità di un piano di appoggio per poterci lavorare. Infine un dispositivo mobile deve potersi connettere a internet. In particolare esistono dispositivi a connessione totale, cioè che possono connettersi sempre in pochi secondi. Tali dispositivi si distinguono da quelli a connessione parziale, cioè che occasionalmente potrebbero non poter stabilire una connessione internet.
È facile intuire che l'introduzione dei dispositivi mobile nel settore sanitario, specialmente quelli dotati di input multitouch, porta a vantaggi enormi. In primo luogo, essendo infermieri e medici sempre in movimento all'interno delle varie aree e reparti che costituiscono la struttura in cui operano, il fatto di poter aver sempre con se, poter consultare e interagire immediatamente col dispositivo mobile permette una riduzione dei tempi e una prontezza di intervento. In secondo luogo, si possono evitare errori dovuti alla mancanza immediata di informazioni allineate e aggiornate, favorire la comunicazione fra il personale e limitare l'utilizzo di informazione cartacea solo dove necessario.
Nel primo capitolo vengono introdotti gli elementi necessari per la comprensione dell'argomento trattato. In particolare si fa riferimento alle Rich Internet Applications, alla loro usabilità e alle problematiche che vengono a crearsi nel loro sviluppo. Inoltre vengono delineati gli aspetti che caratterizzano e distinguono la progettazione per dispositivi multitouch da quella tradizionale.
Nel secondo capitolo verranno analizzati i requisiti di progetto dell'applicazione agenda. Nello specifico vengono individuati gli utenti che ne faranno uso e relativi ruoli e le funzionalità volute. Successivamente viene trattato il progetto di tale applicazione sottolineandone la struttura, il layout e l'interazione a partire dallo schema di database utilizzato in azienda.
Il terzo capitolo presenterà il sistema di sviluppo Instant Developer, che è stato lo strumento utilizzato per la realizzazione dell'agenda web. Tale sistema facilita lo sviluppo di Rich Internet

Application di classe enterprise superando il problema della frammentazione di dispositivi, linguaggi e tecnologie che sta avvenendo. Inoltre, viene descritta la macchina di sviluppo utilizzata, utile per capire la tipologia di dispositivi multitouch a cui è principalmente indirizzata l'applicazione.
Nel quarto capitolo verranno descritte in maniera dettagliata le diverse fasi di implementazione dell'applicazione. Qui vengono spiegati gli oggetti e le strategie messi a disposizione da Instant Developer per ottenere dall'applicazione i risultati desiderati, espressi nei requisiti. Sono inoltre presenti stralci di codice e screenshot che arricchiscono la comprensione.
Infine verranno riportati i risultati ottenuti, i possibili sviluppi futuri del lavoro svolto e, per concludere, alcune considerazioni personali e di carattere tecnologico.

Capitolo 1. Rich Internet Applications per il settore sanitario

Il progetto di Rich Internet Applications (RIA), soprattutto quelle rivolte a dispositivi multitouch, richiede la comprensione dei concetti generali quali usabilità, navigabilità e interfaccia grafica dei siti web, fondamentali per lo sviluppo di servizi efficaci. Molte delle idee e strategie utilizzate nello sviluppo di applicazioni desktop vanno adattate al mondo web e rese fruibili tramite il tocco del dito.
In questo capitolo verranno descritte le caratteristiche e problematiche legate alla creazione e utilizzo di applicazioni web e le principali linee guida per la progettazione di interfacce touch. Infine, verrà introdotto il servizio sanitario che l'applicazione offre.

1.1 Definizione di Rich Internet Applications

Le Rich Internet Applications **[Web2.0]** sono applicazioni web definibili come "ricche" in quanto dotate di funzionalità e caratteristiche che le fanno somigliare ad applicazioni desktop che girano in locale, senza però la necessità di installazione del software su disco fisso. Tali applicazioni possono trasferire sul browser già una buona parte della logica applicativa e alcuni dati, in modo che possano essere utilizzabili subito dall'utente non appena richiesti.
Le RIA **[Wikipedia]** si caratterizzano perciò per la velocità d'esecuzione, la multimedialità e per la rilevante interattività. Anche l'interazione con una RIA avviene in remoto, tramite un comune web browser. La possibilità però di richiedere al server l'aggiornamento solo di parte della pagina, e non di tutta, rende più efficiente e rapido per l'utente il dialogo client/server.
In un certo senso le RIA rappresentano una generazione di applicazioni che permette un'interazione totalmente rinnovata, fondata sugli aspetti migliori delle caratteristiche funzionali e progettuali prerogativa soprattutto delle applicazioni desktop.

1.2 Usabilità

Chi naviga un sito web è alla ricerca di informazioni oppure vuole avvalersi di un servizio. Per risultare usabile, un sito deve quindi soddisfare i bisogni dell'utente. Si deve perciò evitare che una informazione o una funzionalità attesa sia mancante, nascosta o difficile da reperire.
Un'altra situazione da evitare è quella di organizzare i dati o il servizio in modo difficilmente fruibile. Questo capita, ad esempio, quando il sistema di navigazione non rende possibile costruire aspettative chiare su cosa occorre fare per accedere all'informazione o alla funzione desiderata. Questa situazione purtroppo è spesso frequente con la conseguenza di portare alla frustrazione l'utente.

1.2.1 Definizione e requisiti

Un sito web è usabile **[UsabSiti]** quando soddisfa i bisogni informativi dell'utente finale che lo sta utilizzando e interrogando, fornendogli facilità di accesso e di navigabilità e consentendo un adeguato livello di comprensione dei contenuti.
La progettazione e il design di un sito web centrati sui bisogni informativi degli utenti finali devono poggiare su almeno sei requisiti che derivano da ricerche e test di usabilità condotti dalla Human Computer Interaction. Nella **Tabella 1** viene mostrata una breve descrizione dei sei requisiti citati.

Requisiti	Descrizione
Navigabilità	esistenza di un sistema di navigazione che aiuti a orientarsi nel sito e a cercare l'informazione.
Utilità attesa	disponibilità nel sito di informazioni e servizi che corrispondano alle aspettative degli utenti finali e che consentano il conseguimento di una gamma di scopi.
Completezza dei contenuti	presenza nel sito di contenuti informativi al livello di dettaglio desiderabile per gli utenti finali.
Comprensibilità delle informazioni	forma e qualità con cui l'informazione e i contenuti vengono presentati nel sito. L'uso di un linguaggio e di forme di rappresentazione dell'informazione vicine a quelle in uso fra gli utenti finali.
Efficacia comunicativa	come si riflette sull'interfaccia grafica la strategia comunicativa del sito. Fanno parte dell'efficacia comunicativa la rilevanza dei contenuti rispetto al soggetto su cui è sviluppato il sito.
Interfaccia grafica	qualità della grafica e la piacevolezza visiva del sito.

Tabella 1 Requisiti emergenti dell'usabilità dei siti web

1.2.2 Navigabilità

È certamente il requisito di usabilità più importante. Per far si che l'applicazione sia navigabile **[UsabSiti]**, possa essere cioè fruita, usata e finalizzata, bisogna innanzitutto organizzarla. Affinché quest'organizzazione sia funzionale, l'applicazione deve soddisfare i bisogni degli utenti finali garantendo un facile accesso alle informazioni contenute e un rapido e semplice utilizzo dei servizi offerti.

Un buon sistema di navigazione riduce le probabilità di perdita di orientamento da parte dell'utente durante l'utilizzo del sito web. È fondamentale quindi valorizzare tutti gli ausili alla navigazione presenti nel sito. Ad esempio, i link o i bottoni di collegamento fra le varie pagine tolgono immediatamente da qualunque imbarazzo l'utente durante una navigazione incerta.

Il sistema di navigazione è dunque, senza dubbio, un fattore critico per determinare il successo dell'applicazione. L'architettura delle informazioni e delle funzionalità ben pianificata e realizzata offre diversi vantaggi. In primo luogo è possibile, anche essendo un utente che utilizza per la prima volta l'applicazione, avere un contesto chiaro e utile per la navigazione. Risulta quindi immediato raggiungere un determinato dato o servizio senza che sia necessario costruirsi in anticipo un modello mentale di come convenga procedere. In secondo luogo, costituisce una struttura che è più semplice da gestire in modo tale da consentire una agevole modifica o estensione dell'applicazione.

È molto utile a questo scopo strutturare gerarchicamente l'applicazione, realizzarne cioè una mappa che, a seconda delle sue caratteristiche, favorirà percorsi obbligati oppure percorsi liberi.

1.2.3 Interfaccia grafica

La grafica delle applicazioni web **[UsabSiti]** gioca diverse funzioni emotive sull'utente finale, alcune chiare ed esplicite altre oscure e nascoste. La riuscita di una realizzazione grafica è determinata dal giusto equilibrio tra estetica e funzionalità.

L'uso della grafica a soli scopi estetici, senza cioè trasmissione di informazioni, ha i difetti di appesantire l'interazione e aumentare i tempi di caricamento delle pagine senza aver nessun beneficio concreto. L'effetto prodotto è anzi quello di ridurre la credibilità e la fiducia riposta nell'applicazione. Di contro, un'applicazione di solo testo e informazioni scritte non è affatto persuasiva e produce uno scarso coinvolgimento dell'utente finale.

L'obiettivo è dunque quello di presentare dei contenuti attraverso la grafica. Sarà infatti sicuramente gradito agli utenti se la grafica ha lo scopo di aiutare a prendere decisioni e indirizzare verso la meta desiderata. Come esempio si può pensare all'introduzione di icone, immagini, foto e altro come sostegno alla scelta dei possibili percorsi all'interno di un'applicazione. Tuttavia è necessario tenere in considerazione che la sola icona potrebbe non bastare e che quindi sia comunque necessaria, ad esempio, una breve frase esplicativa.

Una ulteriore funzione che può svolgere la grafica è quella di ausilio all'organizzazione delle informazioni. Spesso infatti gli utenti hanno un'esperienza di confusione nell'utilizzo di un sito web. Questo accade quando la pagina presenta una elevata densità di informazione e scarsi aiuti per discriminare tra contenuti di diversa priorità. Alle volte basterebbero anche semplici rimedi di separazione di contenuti per migliore la qualità, l'eleganza e la credibilità del sito.

1.3 Profilo tecnologico degli utenti finali

Considerare e raccogliere informazioni sulla dotazione tecnologica con la quale gli utenti finali accedono a un sito web è un processo utile per assicurarne la qualità**[UsabSiti]**. In primo luogo, si individuano gli ambienti per i quali progettare il sito. In secondo luogo, si individuano gli ambienti di cui tenere conto per i testing di usabilità; ambienti che riflettono quelli che gli utenti useranno per accedere al sito.

1.3.1 Compatibilità sui diversi browser

Consideriamo adesso l'informazione tecnologica riguardante il browser utilizzato. Ogni browser analizza e implementa l'HTML in base alle proprie logiche e regole, e l'esperienza visiva del sito può differire in funzione di esse. Inoltre, non tutti i browser sono in grado di attivare qualsiasi funzionalità presente nel sito. Una chiara consapevolezza su vantaggi e limitazioni presenti in ogni browser permette di adattare al meglio le soluzioni di progetto. Di solito è più facile e comodo scegliere di progettare solo per le ultime versioni dei browser maggiormente diffusi. Tuttavia questa scelta nasconde delle insidie poiché rischia di tagliare fuori un buon numero di utenti. Inoltre, spesso è fondata sul presupposto sbagliato che l'HTML viene gestito allo stesso modo da differenti versioni dello stesso browser. In realtà, l'aderenza alle specifiche dell'HTML può variare da versione a versione. È quindi buona norma tenere conto delle differenti versioni dei diversi browser, soprattutto in sede di testing di usabilità.

1.4 Progettazione per dispositivi multitouch

I dispositivi multitouch **[ProgWebMobile]** hanno funzionalità peculiari, in quanto ogni pixel dello schermo è un potenzialmente attivo. Questo comporta, oltre ai concetti e alle considerazioni generali viste sinora, ulteriori accorgimenti per la progettazione delle interfacce su questi dispositivi. Bisogna tenere in conto quali gesti sono supportati dai dispositivi multitouch in questione e che gli utenti utilizzano il dito come puntatore (nel caso non si utilizzi o non sia previsto

lo stilus). Per questa ultima considerazione è utile che l'input scritto richiesto all'utente sia ridotto al minimo indispensabile.
Di seguito si analizza come è possibile rendere un'interfaccia grafica il più intuitiva, usabile e semplice per un uso touch.

1.4.1 Usare interazioni "naturali"

Uno dei grandi meriti di questa nuova generazione di interfacce sta nella capacità di rendere l'interazione il più intuitiva possibile, eliminando la mediazione di dispositivi di input non necessari, come il mouse. Questo è possibile attraverso gesti più o meno codificati, che permettono di manipolare gli elementi in modo semplice e immediato.
Quando si progetta un'applicazione è utile quindi usare elementi interattivi che assecondino la naturalezza dell'interfaccia touch.
È opportuno, ad esempio, utilizzare il meno possibile scrollbar. Una intuizione geniale è stata quella di sostituire questo scomodo elemento con qualcosa di più naturale: lo scorrimento del dito sullo schermo che permette di scorrere il contenuto in modo del tutto naturale.
Un altro accorgimento è quello di sfruttare le gestures usate per attivare lo zoom. In quasi tutti i dispositivi touch, i controlli di zoom sono stati ormai sostituti da una gesture standard (avvicinamento/allontanamento delle dita), che rappresenta un'ottima rappresentazione che riduce il distacco concettuale tra effetto desiderato (ingrandimento/rimpicciolimento dello schermo) e l'azione necessaria. È inutile quindi aggiungere altri sistemi di controllo dello zoom.
Un ulteriore elemento consigliato sono i controlli di tipo drag-and-drop. Essi rappresentano infatti ottimi esempi di interazioni naturali, ovvero simili al mondo reale in cui viviamo.
Entrare nell'ottica di progetto di interfacce grafiche per dispositivi multitouch richiede un procedimento di distacco rispetto alla progettazione di interfacce grafiche abituale, quella adatta al cursore del mouse. Il mouse infatti possiede una elevata precisione (1-2 pixel) mentre il dito, o meglio il polpastrello, ha una precisione decisamente più bassa. Questo implica l'ingrandimento degli elementi interattivi e una spaziatura adeguata fra essi, nel caso siano presenti almeno due di questi oggetti ravvicinati.
Bisogna inoltre considerare che il tocco del dito sullo schermo comporta per quest'ultimo la funzione di dispositivo di input e output insieme. Per questo motivo durante l'input, la mano potrebbe coprire elementi di rilievo. È quindi opportuno non posizionare etichette sotto gli elementi di interazione.
Infine è importante notare che l'evento hover, che attiva comportamenti speciali al passaggio del mouse su determinati elementi, nei dispositivi multitouch non esiste. È indispensabile quindi che sia immediatamente evidente nell'interfaccia cosa è interattivo e cosa invece no.

1.5 Settore di riferimento

Il settore di riferimento dell'applicazione web da me realizzata è quello sanitario. Questo settore è molto vasto e comprende aziende pubbliche (aziende sanitarie locali, aziende ospedaliere) e aziende private (Cliniche e Case di Cura).
L'ambito di riferimento da me affrontato, in maniera ristretta e parziale, è stato quello dell'agenda, ossia uno dei servizi richiesti nell'ambito della gestione ambulatoriale relativo alle aziende pubbliche. L'agenda permette una gestione completa, da parte del personale autorizzato, delle prenotazioni relative ad ogni attività svolta negli ambulatori che compongono l'intera struttura pubblica. Inoltre, viene offerta la possibilità di consultare i referti sinora stilati dei pazienti che richiedono tali prenotazioni.
L'agenda, come le altre applicazioni web del settore sanitario, sono considerate RIA in quanto esse si distinguono per l'alta interattività e la rapidità di esecuzione.

Capitolo 2. Analisi dei requisiti e progetto dell'applicazione

Attualmente la maggior parte delle applicazioni software indirizzate al settore sanitario danno ancora scarso rilievo all'interfaccia utente con i conseguenti problemi di usabilità che questo comporta.
Tali interfacce si presentano, per lo più, solo come una lunga sequenza di campi da riempire o di record visualizzati. L'unico aspetto estetico che si può notare è la colorazione diversa del testo o dello sfondo di alcuni campi importanti e l'utilizzo di bottoni di conferma.
Sicuramente il passaggio a un'interfaccia grafica di forte attrattiva o l'utilizzo di queste applicazioni in dispositivi multitouch non è cosa semplice e immediata.
Nel caso dell'agenda ambulatoriale da me preso in esame però questo passaggio è più facile rispetto ad altri prodotti. In esso infatti ci sono molti spunti su dove e come utilizzare la grafica per esprimere contenuti informativi e su come sfruttare e inserire interazioni naturali per un interfacciamento coinvolgente con l'utente.
In questo capitolo verranno presentati i requisiti necessari per la progettazione di tale applicazione su dispositivi multitotuch, illustrando e motivando le scelte progettuali relative all'interfaccia prese in questo senso.

2.1 Requisiti

L'applicazione è rivolta a medici e personale sanitario e deve offrire la possibilità di visualizzare le prestazioni prenotate per una determinata attività di un determinato ambulatorio. Ogni ambulatorio ha quindi una o più attività specifiche al suo interno e il personale di una attività può essere diverso da quello di un'altra attività, anche se l'ambulatorio di riferimento è lo stesso.
In primo luogo, all'acceso, bisogna quindi individuare l'utente che richiede l'utilizzo dell'applicazione in modo da poter discriminare su ambulatori e relative attività da rendere visibili.
Una volta autenticato, l'utente deve poter visualizzare tutti gli ambulatori di sua competenza e, selezionatone uno, deve poter visualizzare le relative attività a cui ha accesso.
Una volta selezionata un'attività specifica, deve essere possibile scegliere un qualsiasi giorno per visualizzarne le relative prenotazioni. Tale possibilità di scelta è la qualità fondamentale di questa applicazione agenda.
Lo scopo del progetto è quello di rendere la scelta e la visualizzazione molto intuitiva e vicina al modello che conosciamo e che è presente nell'immaginario comune di agenda di appuntamenti.
Un ulteriore requisito è quello di poter mostrare il dettaglio di una singola prenotazione e, nel caso siano presenti, visualizzare i relativi referti del paziente che ha richiesto la prenotazione. Anche per quanto riguarda la visualizzazione dei referti l'obbiettivo è di realizzare una interazione piacevole e stimolante per l'utente finale.
Infine, per soddisfare i bisogni dell'ambiente sanitario, l'interfaccia dovrà presentare i seguenti requisiti: i record che rappresentano una prenotazione dovranno contenere i soli dati tecnici utili e dovranno essere ognuno ben visibile (il font size minimo per una lettura corretta è 12 pt); i relativi dettagli dovranno avere le stesse caratteristiche ma senza ripetere informazioni già visualizzate nella testata; i tasti dovranno essere pochi, ben visibili e posizionati nella parte superiore della pagina per facilitarne la comprensione e l'utilizzo.

2.1.1 Ruoli e permessi

Gli utenti applicativi sono suddivisi in gruppi in base al ruolo che ricoprono all'interno dell'azienda ospedaliera in cui operano. Il codice del gruppo è memorizzato in un campo apposito della tabella che contiene l'anagrafica degli utenti autorizzati, insieme ovviamente all'username, la password e altri dati.

Essendo l'applicazione di sola consultazione tutti gli utenti sono comunque abilitati alla sola lettura delle informazioni e la consultazione avviene nella stessa modalità per tutti. La circoscrizione degli ambulatori e delle attività specifiche, in termini di visibilità, dipende generalmente dal singolo soggetto in questione e non a priori dal ruolo. Tale restrizione avviene durante la fase di login, dove vengono controllate le credenziali dell'utente.

2.1.2 Funzionalità

Una volta autenticato l'utente deve poter visualizzare la lista di tutti gli ambulatori a cui è autorizzato ad accedere e deve poterne selezionare uno da questo elenco.
Successivamente l'utente deve poter scegliere l'attività specifica di cui visualizzare le prenotazioni dalla lista delle attività a cui ha accesso, relative all'ambulatorio precedentemente selezionato.
L'applicazione deve offrire tre funzionalità di visualizzazione dell'agenda: vista mese, vista giorno e vista mesi-giorni.
La vista mese di una attività specifica deve visualizzare i giorni del mese e offrire informazioni riguardanti quanti slot (periodi di tempo che variano a seconda dell'ambulatorio e della attività specifica, tipicamente di 15, 30, 45 o 60 minuti) sono liberi e quanti invece sono già occupati.
La vista giorno di una attività specifica deve visualizzare invece tutti gli slot presenti in una determinata giornata (ogni giornata può avere orari diversi: solo la mattina, solo il pomeriggio, entrambi o nessuno, solo alcune ore) con relativo orario. Nello slot corrispondente, se presente, deve essere visualizzata la testata della prenotazione che comprende: nome, cognome, codice fiscale, sesso, data di nascita e codxmpi (identificativo univoco) del paziente, nonché il codice e la data in cui è stata richiesta la prenotazione.
La vista mesi-giorni deve infine visualizzare i mesi e i giorni insieme, permettendo una scelta di uno specifico giorno più ampia e meno restrittiva di quella di un singolo mese.
Inoltre deve essere possibile visualizzare il dettaglio di una specifica prestazione, ossia vedere quale o quali visite sono previste e corrispondono a quel paziente, in quella determinata data e orario. È infatti possibile che un paziente debba fare due o più esami con la stessa prenotazione, ad esempio due prelievi del sangue diversi; in quest'ultimo caso devono essere visualizzati entrambi i prelievi. Il dettaglio prevede le informazioni riguardati il tipo, il codice e la descrizione dell'esame o visita da effettuare.
Un'ulteriore funzionalità è quella di visualizzazione dei referti di un paziente: deve essere possibile accedere all'intera lista dei referti di un paziente e, selezionatone uno, si deve poterne visualizzare il contenuto.
Deve essere sempre possibile tornare indietro alla scelta dell'ambulatorio e dell'attività specifica per consentire il cambio di prenotazioni che si vogliono visualizzare.
Infine, le tre diverse viste agenda dell'attività specifica devono essere fra loro collegate in modo da potersi muovere facilmente da una visualizzazione all'altra a seconda delle necessità.

2.2 Progetto

In questo paragrafo verranno analizzati lo schema del database utilizzato e le scelte progettuali fatte che hanno portato alla realizzazione dell'applicazione sulla base dei requisiti e delle funzionalità esposti.

2.2.1 Schema del database

Per lo svolgimento dell'applicazione mi è stato assegnato un utente Oracle. Oltre ad avere i diritti di lettura e scrittura su un database ormai inutilizzato in modo che potessi fare delle prove, tale utente

aveva accesso, attraverso un db-link, al database contenente le tabelle coinvolte nell'applicazione agenda in modalità di sola lettura. Le tabelle da me utilizzate sono:

1) UTE_ANAG: contiene l'elenco degli utenti applicativi autorizzati ad accedere;
2) RISORSE_UNITA: contiene ambulatori e attività specifiche (unità) alle quali l'utente è autorizzato ad accedere;
3) UNITA_GERARCHIE: definisce il legame tra unità master (ambulatorio) e le unita correlate (attività specifiche);
4) UNITA: contiene l'anagrafica unità (ambulatori e attività specifiche);
5) AGENDAESPLOSA, ANAGPRENOT, MOVIPRENOT, PRESCPRENOT : sono tabelle di gestione rispettivamente di attività specifiche e prenotazioni;
6) DOCUMENTI_PDF: contiene i referti dei pazienti.

Queste tabelle sono state il punto di partenza del progetto. È chiaro che l'intento della mia esperienza non era quello di modificare lo schema del database. Innanzitutto non era necessario visto che avevo già a disposizione un sistema funzionante e collaudato da molti anni. Inoltre la costruzione di questo database, che in realtà comprende molte più tabelle, richiede molta esperienza nel settore medico e molto tempo di progettazione.
Per questi motivi le uniche informazioni e documentazioni ricevute in merito allo schema riguardano le query per soddisfare parti delle funzionalità richieste. In **Figura 1** è riportato lo schema a blocchi delle entità coinvolte nell'applicazione (non corrispondono alle tabelle), in cui è possibile capire visivamente i dati forniti dalle tabelle.

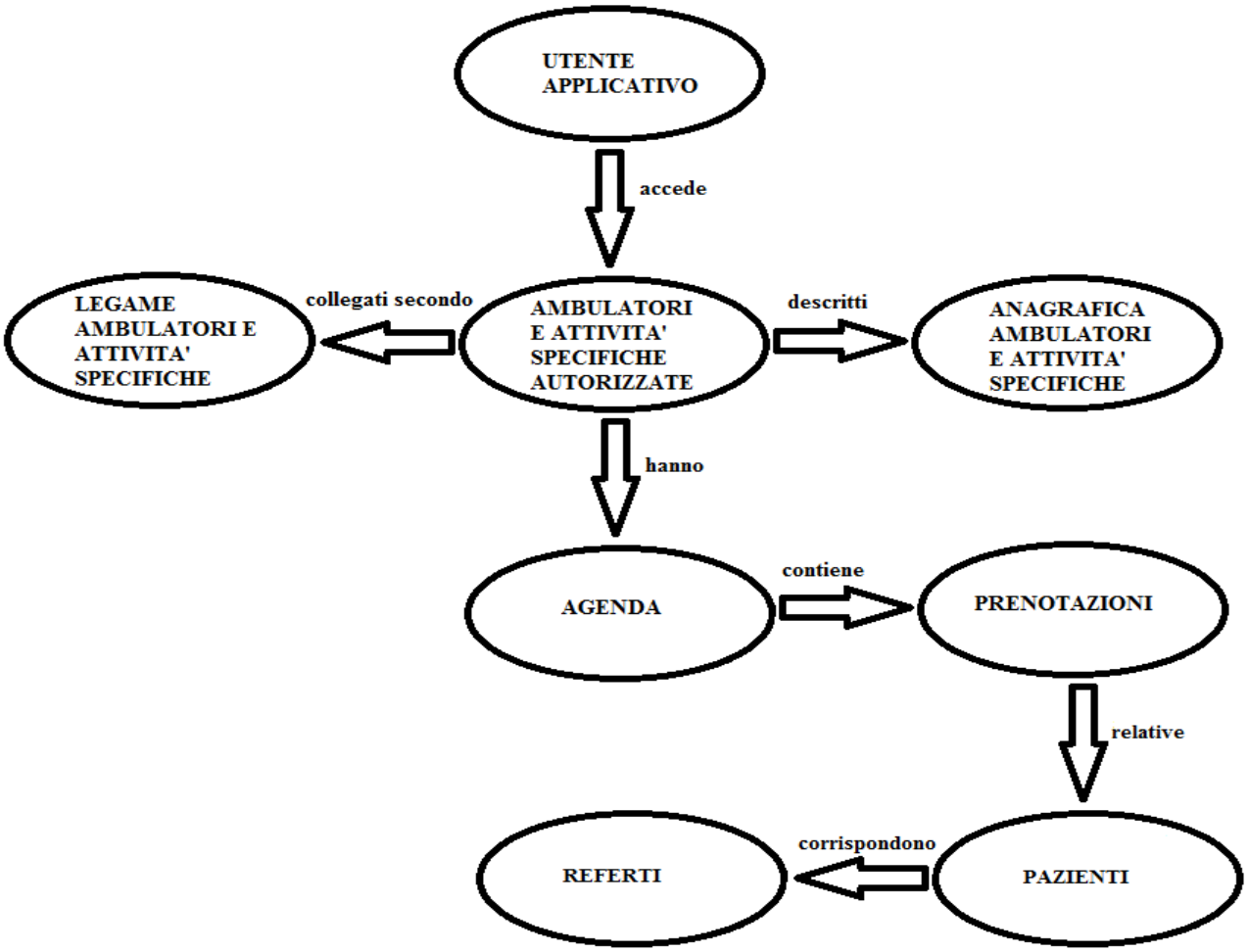

Figura 1 Schema a blocchi delle entità coinvolte nell'applicazione.

2.2.2 Struttura dell'applicazione

La struttura dell'applicazione è stata pensata per permettere sempre la possibilità di ritornare alla scelta dell'ambulatorio e della relativa attività specifica e per far muovere l'utente da una qualsiasi vista agenda a un'altra. Alcune delle funzionalità sono state raggruppate in unica pagina secondo opportune motivazioni di seguito riportate. L'applicazione presenta diversi flussi di comunicazione e casi d'uso rappresentati in **Figura 2**. Di seguito vengono descritte le varie sezioni dell'applicazione seguendo lo schema e simulando la navigazione di un utente tra le pagine.

Come si può notare dallo schema, la pagina iniziale è la form di login. Essa richiede username e password all'utente per permetterne l'identificazione e la verifica dell'autorizzazione o meno a procedere.

In caso affermativo, la seconda pagina che si incontra è quella di scelta dell'ambulatorio e relativa agenda. Si è deciso di raggruppare queste due funzionalità perché estremamente legate una con l'altra. Dividerle, inoltre, avrebbe comportato un ulteriore livello per l'utente finale da ripetere ogni qual volta si decida di cambiare ambulatorio.

Selezionata un'attività specifica di un ambulatorio, il passo successivo del percorso, sinora obbligato, prevede la vista mese di tale attività. Dopo di che è invece sempre possibile muoversi tra le tre diverse visualizzazioni dell'agenda, ognuna implementata in una diversa pagina.

La funzionalità di visualizzazione del dettaglio della prenotazione scelta è stata accorpata nella vista giorno dell'agenda perché non abbastanza consistente da formare una pagina a se stante.

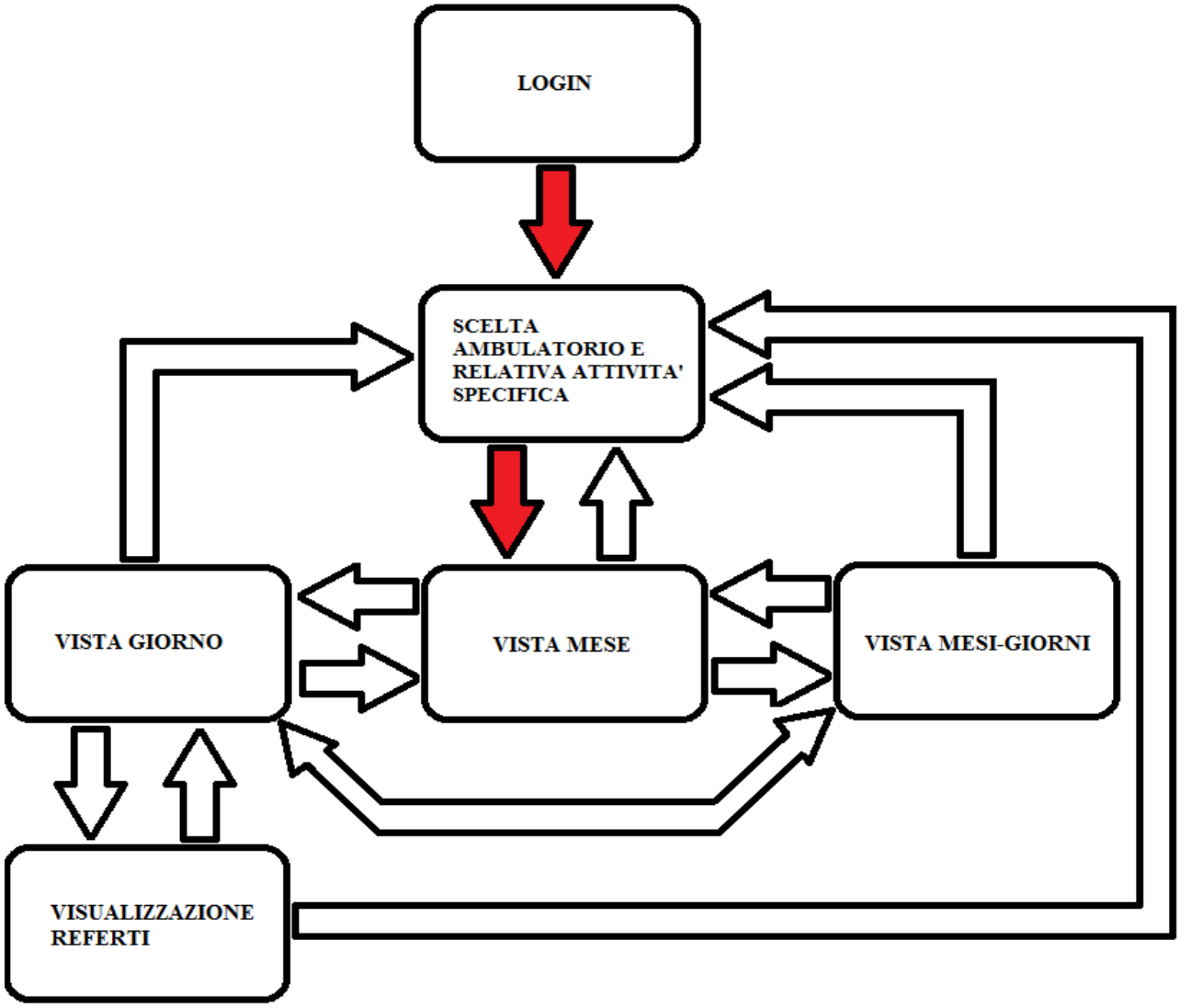

Figura 2 Diagramma struttura applicazione agenda. In rosso è riportato il percorso obbligato.

Inoltre, questa informazione deve variare a seconda della testata della prenotazione selezionata quindi il raggruppamento di queste due funzionalità, di per se correlate, permette una maggiore comprensione della informazione esposta.
Infine, la pagina che realizza la funzionalità di visualizzazione referti è stata collegata solo alla vista giorno dell'agenda. Tale funzionalità necessita infatti, come parametro in ingresso, dell'identificatore del paziente visibile solamente nella testata delle prenotazioni. Da qui la scelta della consequenzialità esclusiva tre le due videate.

2.2.3 Layout e interazione

L'obiettivo principale dell'applicazione era progettare una struttura grafica funzionale per i dispositivi e il settore di interesse e che invitasse l'interazione con l'utente. Questa interazione doveva essere il più naturale possibile affinchè l'utente in modo intuitivo potesse muoversi nell'applicazione. Analizziamo adesso le decisioni progettuali prese in questo senso.
La prima pagina che si incontra durante la navigazione dell'applicazione, dopo il login, è la modalità di scelta dell'ambulatorio e dell'attività specifica, che è stata adeguatamente studiata e modificata rispetto all'applicazione tradizionale per PC. Infatti, tale scelta non poteva più essere attuata attraverso dei menù a tendina dove selezionare i valori desiderati e poi confermare l'operazione attraverso un apposito bottone di conferma, in quanto questa soluzione non è adatta a una fruizione rapida e ottimale col dito. Si è optato quindi per la visualizzazione di una lista di ambulatori autorizzati da poter leggere rapidamente attraverso lo scorrimento del dito sullo schermo. Automaticamente al cambio della selezione dell'ambulatorio vengono visualizzate, in un'altra lista da poter scorrere col dito, le attività specifiche relative. Per confermare l'attività specifica basta selezionare la riga desiderata.
Proseguendo la navigazione, alla vista mese dell'agenda si è data una impaginazione stile calendario dove i giorni sono raffigurati attraverso dei riquadri. Attraverso dei tasti è possibile muoversi al mese seguente o precedente. Lo stesso vale per cambiare anno. Attivando uno specifico giorno si passa alla vista giorno dell'attività specifica. Attraverso un bottone è possibile passare alla vista mesi-giorni.
Per quanto riguarda invece la vista giorno dell'agenda sono presenti anche qui due liste da poter scorrere col dito. La prima visualizza le testate delle prenotazioni. Automaticamente al cambio della testata della prenotazione selezionata, nella seconda lista, ne viene visualizzato il dettaglio. Anche qui è possibile muoversi nel calendario attraverso tasti precedente e successivo oppure scegliendo direttamente una data specifica. Per accedere alla visualizzazione referti si è pensato di sfruttare l'immediatezza e l'intuitività del drag-and-drop: tirando le credenziali del paziente, dalla lista delle testate delle prenotazioni, e rilasciandole su un'apposita area, dotata di link e icona annessa, si passa alla visualizzazione referti di quel paziente. Attraverso due bottoni è possibile passare rispettivamente alla vista mese e alla vista mesi-giorni di quel giorno specifico.
Anche per quanto concerne la vista mesi-giorni si è pensata una impaginazione stile calendario. Questa vista prevede i mesi disposti verticalmente, in colonna uno sotto all'altro, sul lato sinistro della pagina. Nella parte restante di pagina, invece, disposti orizzontalmente, in riga uno di fianco all'altro, i giorni relativi al corrispondente mese. Ovviamente non possono essere visualizzati insieme tutti i dodici mesi e tutti i relativi giorni. È quindi possibile scorrere i mesi dell'anno e scorrere i relativi giorni col dito oppure con appositi tasti. Anche qui è possibile cambiare anno da visualizzare. Inoltre selezionando un mese dall'apposita colonna si passa alla vista mese, selezionando un giorno dall'apposita riga si passa alla vista giorno.
La visualizzazione dei referti ha una struttura grafica che si avvicina allo stile rassegna stampa dei quotidiani vista in tv. Lo scopo è quello cioè di visualizzare su una colonna laterale la lista di referti relativa al paziente da poter scorrere mediante scorrimento del dito o utilizzo di tasti. A questo punto deve essere possibile tirare un referto dalla lista e rilasciarlo nella restante parte della pagina in modo da visualizzarne il contenuto. Inoltre, è possibile visualizzare due referti

contemporaneamente, svuotare e cambiare lo sfondo della parte di pagina su cui si visualizzano i referti e tornare alla vista giorno, tutto attraverso appositi tasti.

Capitolo 3. Tecnologie

Oggi lo sviluppo di Rich Internet Application sia in ambito PC che dispositivi mobile è diventata un'attività complessa; basta pensare infatti alla notevole segmentazione dei dispositivi, dei linguaggi e delle tecnologie di sviluppo. Ecco quindi che ogni software house deve affrontare queste problematiche cercando di ridurre i costi e i tempi di produzione evitando di ricercare altro personale con competenze elevate e specifiche.
In questo capitolo verrà trattata la soluzione adottata nella realizzazione dell'applicazione agenda per risolvere le problematiche introdotte. Tale soluzione è il sistema di sviluppo Instant Developer di cui ne viene analizzato il funzionamento, il framework, la strutturazione di un progetto e la compilazione delle applicazioni.
Successivamente verrà presentata la macchina di sviluppo utilizzata, utile per capire la tipologia di dispositivo multitouch verso cui è principalmente indirizzata l'applicazione.

3.1 Instant Developer

Instant Developer **[In.deGuidaUso]**, spesso abbreviato con In.de, è prodotto da Pro Gamma e offre un ambiente esclusivo per generare rapidamente, con facilità e costi contenuti, applicazioni RIA anche su dispositivi mobile, estensioni web a sistemi esistenti, conversioni applicative, senza vincoli di architettura e linguaggio.
Grazie all'utilizzo della programmazione relazionale, In.de rappresenta una delle alternative per risolvere i problemi sopracitati: sia per quanto riguarda la gestione della complessità, che lo rende uno dei sistemi più efficaci per affrontare applicazioni di carattere enterprise, sia per quanto concerne la volatilità delle tecnologie offrendo la possibilità di ottenere Rich Internet Application sempre allo stato dell'arte.
Instant Developer è' un vero e proprio sistema di sviluppo che gestisce tutti gli aspetti del ciclo di vita del software, dall'analisi all'installazione e oltre.
Applicazioni interattive (come quelle client/server) e di interfaccia intuitiva (come quelle per dispositivi multitouch) sono ottenibili con In.de in tempi brevi e senza dover per forza conoscere tutti i dettagli tecnici.
Le applicazioni create vengono automaticamente tradotte e compilate sia in linguaggio Java che C#, in modo da funzionare su qualunque piattaforma. Si collegano in modo ottimizzato a ogni tipo di database supportato e possono essere usate con qualsiasi browser, compresi quelli di tablet e smartphone in cui vengono sfruttate le caratteristiche peculiari come il multitouch, le animazioni native e il caching HTML5.
Il codice sorgente generato è standard. Esso è leggibile, commentato ed eventualmente manutenibile anche senza utilizzare Instant Developer, in modo da non esserne dipendenti.

3.1.1 La programmazione relazionale

Il cuore del funzionamento di Instant Developer è la programmazione relazionale. In.de permette infatti di descrivere il comportamento del software tramite la composizione di un grafo di relazioni invece che con la scrittura di tanti file di testo, come avviene nei sistemi tradizionali.
Se ad esempio pensiamo a un'applicazione dell'ordine delle migliaia di righe di codice ci rendiamo conto che modificarla è difficile, perché molte righe di codice sono scritte in modo da dipendere da altre. Consideriamo a tale scopo un database contenente un campo di tipo integer e una procedura che ne legge il valore in memoria. Tale lettura avverrà tenendo conto che il campo è di tipo numerico; se poi tale numero deve essere incrementato, ancora una volta verranno utilizzate funzioni di tipo numerico. Immaginiamo di entrare nel file DDL e di modificarne la riga in cui è definito il campo, da integer a varchar(3). Molto probabilmente l'applicazione smette di funzionare

perché non è previsto che il campo sia di tipo carattere, ma bensì numerico. Per sistemare tutto dovranno essere modificate manualmente tutte le righe di codice che dipendono dal campo e, a cascata, tutte le righe di codice che dipendevano dalle prime righe di codice modificate, con un processo iterativo totalmente manuale che può essere molto lungo e richiedere numerose fasi di test.
All'interno di Instant Developer, invece, il codice non viene memorizzato in un file di testo ma direttamente in un grafo le cui relazioni vengono tracciate automaticamente dall'IDE. Nell'esempio di prima, quando viene letto il campo integer dal database l'istruzione di lettura contiene una relazione con il campo; quindi se esso viene modificato, anche le istruzioni che lo referenziano vengono modificate concordemente. Di conseguenza quasi tutto il lavoro di adattamento viene svolto in modo automatico.

3.1.1.1 Gestione della complessità

Il primo vantaggio specifico della programmazione relazionale è la gestione della complessità, che In.de rende possibile grazie agli strumenti di analisi e sezionamento del grafo delle relazioni e ai meccanismi di auto-adattamento descritti in precedenza.
Infatti, si può notare che all'aumentare della complessità del software aumenta il vantaggio portato dalla programmazione relazionale, che tuttavia è già alto anche in caso di progetti molto semplici.
Inoltre, le applicazioni create rimangono facilmente manutenibili anche dopo anni e anche se diventano molto corpose. Anzi, il vantaggio derivante dalla programmazione relazionale, che è già presente nella fase di creazione del sistema software, aumenta ancora nelle fasi di manutenzione successiva.
Infine, è possibile comprendere il funzionamento del software molto velocemente tramite gli strumenti di software intelligence inclusi. Questo permette di fare manutenzione all'applicazione senza per forza esserne lo sviluppatore e quindi senza comportare costi aggiuntivi.

3.1.1.2 Indipendenza dalla tecnologia

Un ulteriore vantaggio specifico della programmazione relazionale è l'indipendenza dalla tecnologia: gli oggetti contenuti nel grafo delle relazioni sono infatti tali da non dipendere da una specifica implementazione tecnologica. I compilatori interni a Instant Developer sono poi in grado di generare il codice sorgente relativo alla configurazione di tecnologie desiderate.
Ad esempio, è possibile scrivere una query, anche molto complessa (con join fra tabelle, union, subquery e funzioni di calcolo del database) e generarne il codice sorgente specifico per Oracle, SQL Server, DB2, Postgres, MySQL e altri ancora. Addirittura fra una versione e l'altra dello stesso database la query potrebbe essere generata in modo diverso per sfruttarne le caratteristiche migliorative.
Questo significa che c'è una valorizzazione delle specificità di ogni piattaforma tecnologica invece che, come tradizionalmente accade, un'omogeneizzazione al ribasso.
Quanto avviene a livello di database per tutti gli aspetti che li riguardano comprese le query, le stored procedure e i trigger, si estende poi alle varie architetture applicative che compongono l'intero sistema software come ad esempio le web application, i web services, i servizi batch e così via. È possibile ottenere la generazione automatica dell'intero sistema software sia in linguaggio Java che in Microsoft C#, in modo da poter far funzionare l'applicazione su qualunque tipo di server.
Il framework RIA per la generazione delle web application consente di utilizzare l'applicazione su qualunque browser, compresi quelli dei dispositivi mobile sfruttandone al meglio le caratteristiche peculiari, come il multitouch, la geolocalizzazione, le animazioni native e così via.
Il risultato ottenuto è un'applicazione cross-browser non solo in maniera approssimativa ma sostanziale sia nella grafica che nei comportamenti.

Infine, non sarà più necessario aggiornare con frequenza le proprie applicazioni perché l'infrastruttura tecnologica è diventata obsoleta, il medesimo progetto, infatti, potrà essere ricompilato con le versioni successive di In.de e sfruttarne gli avanzamenti a livello tecnologico rimanendo senza sforzo allo stato dell'arte.

3.1.1.3 Possibilità di concentrarsi sull'ambito applicativo

L'implicazione conseguente all'indipendenza dalla tecnologia, per quanto riguarda il programmatore, è quella di potersi concentrare sulla realizzazione della miglior soluzione applicativa possibile in termini di funzionalità, interfaccia utente e user experience. Vi è quindi una separazione tra ambito applicativo e tecnologico che comporta una serie di vantaggi.
Come esempio si pensi alla necessità di rendere cross-browser una Rich Internet Application basata su HTML e Javascript. A oggi, più o meno ogni due settimane, viene pubblicata una nuova versione di un qualche browser, sia esso Internet Explorer che viene aggiornato tramite Windows Update, oppure Firefox, Safari, Chrome o Opera. A ogni aggiornamento vengono corretti dei problemi, ma contemporaneamente ne vengono aggiunti dei nuovi che, in un qualche modo, possono modificare il comportamento delle applicazioni web. Solo per gestire questo problema occorrerebbe un pool di tecnici per testare e verificare in continuazione il comportamento delle applicazioni su tutte le versioni attive di tutti i browser, correggere e di conseguenza indicare come i vari tipi di problemi debbano essere affrontati sui vari browser.
Con In.de invece non è più necessario un aggiornamento continuo dell'intero gruppo di lavoro sulle modifiche che avvengono a livello di piattaforma tecnologica ed è proprio per questo che ci si può concentrare sull'ambito applicativo.

3.1.1.4 Gestione del ciclo di vita del software

La programmazione relazionale implica la conoscenza delle relazioni fra i vari componenti del sistema informativo in fase di costruzione, a partire dallo schema dei database fino all'ultimo dei report che ne fanno parte.
Instant Developer dunque contiene tutti gli strumenti necessari alla gestione dell'intero ciclo di vita del software e non solo di una parte dello stesso.
Le fasi gestite da In.de sono: Analisi, Business Logic, Presentazione, Debug e Test Automatico, Configurazione a Run Time, Documentazione, Assistenza e Versioning.
La gestione unitaria del software consente un controllo completo del sistema informativo in fase di costruzione o modifica; ad esempio, modificando lo schema del database, automaticamente questa informazione viene propagata fino all'ultimo report presente nel progetto.
Un altro aspetto interessante è che i vari moduli hanno accesso a informazioni complessive e quindi permettono ulteriori vantaggi in termini di rilavorazione.
Il sistema di profilazione, che configura le funzioni ammesse ai vari profili applicativi, consente anche di modificare il layout dei report in funzioni delle informazioni che i vari utenti possono vedere.
Il sistema di traduzione multilingua centralizzato può tradurre anche le stampe.
Il modulo di team working, che permette il lavoro di gruppo e il versioning del codice, riesce a operare su tutto il progetto a cominciare dallo schema del database fino ai report.

3.1.1.5 Sfruttare le caratteristiche dei linguaggi di programmazione tradizionali

È importante sottolineare che la programmazione relazione non preclude l'espressività e le potenzialità dei linguaggi di programmazione tradizionali. Le funzionalità previste sono molte:

1. E' possibile usare i costrutti della programmazione object oriented: classi, estensione, interfacce, metodi virtuali…

2. E’ possibile importare le classi Java o C# esistenti per estendere le librerie di In.de e servirsene all’interno dell’editor di codice come quelle predefinite. Il lavoro già fatto può essere così riutilizzato all’interno di un nuovo progetto Instant Developer.
3. E’ presente un framework ORM che permette di costruire la business logic con le stesse logiche di Hybernate + Spring, ADO Entity Framework o J2EE.
4. Sono presenti anche comportamenti di tipo aspect oriented programming (AOP) per sfruttare al meglio le iniezioni di codice e una reflection avanzata.
5. E’ possibile usare codice SQL direttamente all’interno del linguaggio in modo da ottenere un controllo sintattico e semantico a compile time e non avere sorprese durante l’esecuzione dell’applicazione.

Se qualcosa può essere scritto in Java o C#, allora può essere scritto anche con Instant Developer in modo similare e con la stessa logica; anche il codice relazionale viene espresso con un meta-linguaggio simile a Java e C# all’interno del visual code editor.

3.1.2 Il framework RD

Le Rich Internet Application sviluppate con Instant Developer sono basate su un framework dedicato in grado di ottenere applicazioni sicure e performance elevate anche su dispositivi mobile.
Lo schema di funzionamento è riassunto nel diagramma di **Figura 3**. Le principali aree funzionali sono le seguenti:

1. Area RD3: costituita dal browser in cui è in funzione una libreria javascript dedicata e dal blocco RD3 Manager; ha il compito di renderizzare nel browser lo stato dell’interfaccia utente dell’applicazione.

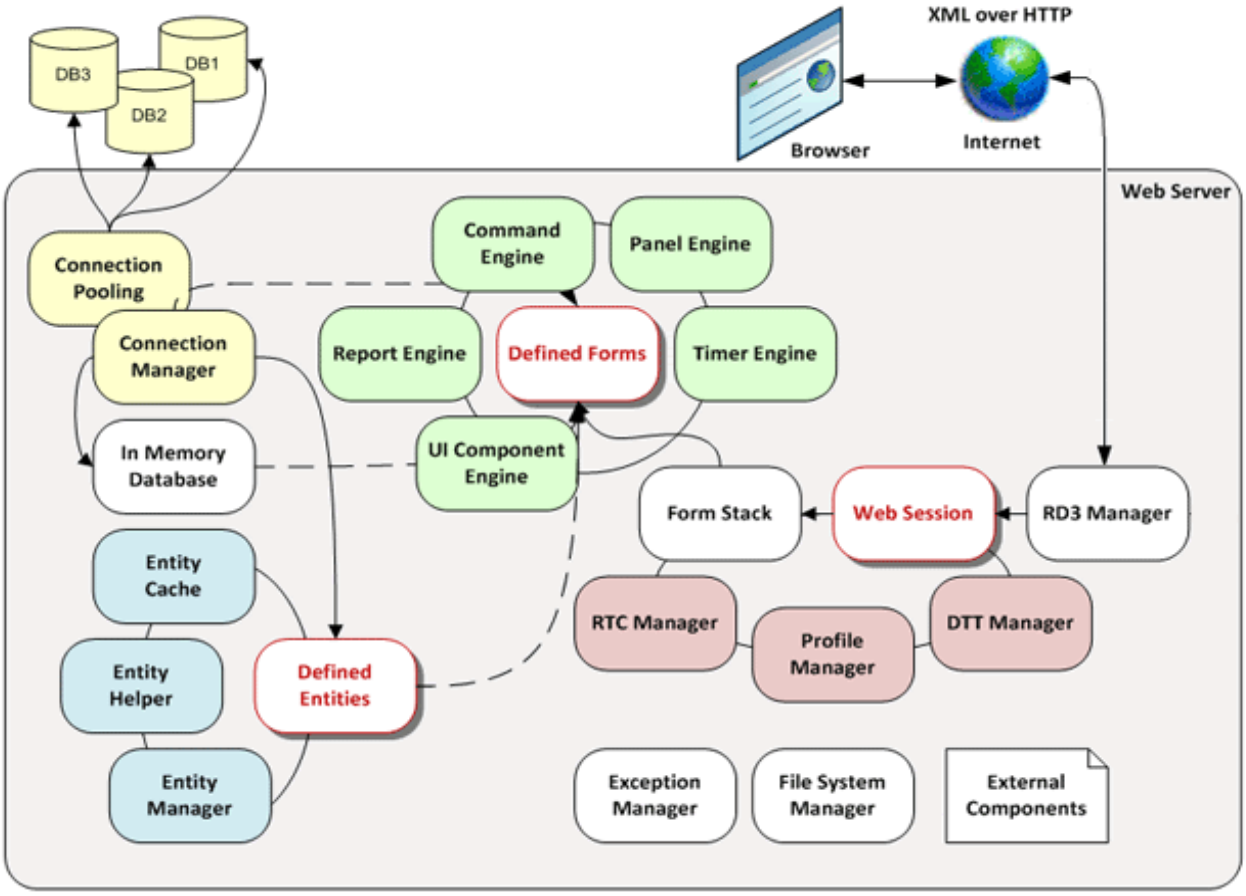

Figura 3 Schema di funzionamento del framework RD.

2. Area Database (gialla): consiste in una serie di servizi di gestione della connessione ai vari database. In questo modo non è mai necessario gestire manualmente le connessioni che saranno sempre sicure e ottimizzate.
3. Area ORM (azzurra): questi moduli costituiscono il sistema Object Relational Mapper di In.de. La realizzazione del business layer non è mai stata cosi semplice.
4. Area UI (verde): contiene i moduli per la rappresentazione logica dell'interfaccia utente lato server, che viene poi trasferita al browser dal modulo RD3 Manager.
5. Area Controllo Sessione (rosa): costituita dai moduli per l'applicazione dei profili applicativi, per la personalizzazione dell'applicazione e per il controllo dell'andamento della sessione (DTT = Debug, Test & Tracing).
6. In memory database: è un oggetto notevole non incluso nelle aree precedenti, visualizzato nel lato sinistro dello schema. Assume particolare importanza perché è parte del controller del framework, cioè del sistema di coordinamento fra la business logic e il presentation manager.

3.1.2.1 Sviluppo di Rich Internet Application

Le applicazioni web create con In.de soddisfano la definizione di RIA indicata nel **Capitolo 1**, infatti il presentation manager dell'interfaccia funziona all'interno del browser tramite una libreria javascript ad alte prestazioni chiamata RD3. Essa comunica con il server web tramite un protocollo basato su XML, in maniera ottimizzata in funzione delle caratteristiche della rete come, ad esempio, banda e tempo di attraversamento. I dati vengono scaricati a finestre e non vengono ulteriormente richiesti al server, permettendo così una navigazione di tipo live scrolling delle liste di dati.
L'ultima parte della definizione da controllare è quella relativa alla fruibilità dell'applicazione da parte di un comune web browser. E' semplice verificare come le applicazioni create con In.de siano cross-browser sia nella grafica che nei comportamenti. Il livello di compatibilità grafica è quasi totale, infatti è possibile sovrapporre gli screenshot dei vari browser e verificarne l'uguaglianza quasi pixel per pixel.
Infine RD3 non richiede plug-in di nessun genere e le applicazioni che lo usano hanno il massimo livello di compatibilità con le leggi sull'accessibilità oggi vigenti.

3.1.2.2 Grafica dell'interfaccia utente

Un aspetto fondamentale nel fare applicazioni oggi è la cura della grafica dell'interfaccia utente che spesso viene pretesa e data per scontato dagli utenti finali. In.de per facilitare ciò mette a disposizione diverse tecniche, tra cui:

1. Stili grafici: all'interno dell'IDE è possibile definire una serie gerarchica di stili grafici che controllano come le informazioni vengono presentate all'utente. Uno stile consiste nell'insieme di quasi 100 proprietà grafiche che permettono di decidere come le informazioni devono essere rappresentate in funzione dei possibili stati applicativi. Il vantaggio è che sono gerarchici, quindi è sufficiente modificare lo stile "padre" per aggiornare lo stile dell'intera applicazione.
2. Temi grafici: permettono di configurare le caratteristiche generali degli oggetti dell'interfaccia e consistono di una serie di icone e di un file cascade style sheet. In.de contiene già alcuni temi grafici che permettono di avere subito un'interfaccia utente allo stato dell'arte, ma è possibile adattarli o crearne dei nuovi per uniformare l'aspetto grafico secondo le proprie esigenze.
3. Libreria Javascript: è la parte del framework RD3 che gestisce l'interfaccia utente dell'applicazione nel browser. E' una libreria in codice aperto, pensata per essere estesa o modificata secondo le proprie esigenze.

3.1.3 Anatomia di un progetto In.de

Un progetto Instant Developer contiene la descrizione di un sistema informativo o di una parte di esso a livello di database, applicazioni e librerie. Gli oggetti coinvolti sono i seguenti:
Progetto: rappresenta il progetto nella sua interezza, l'intera struttura relazionale che lo compone. E' l'oggetto radice dell'albero degli oggetti; non ha una valenza applicativa ma serve per identificare il progetto all'interno del sistema di Team Working.
Database: rappresenta la connessione a un database contenuto in un server. Contiene la definizione di una serie di tabelle e rappresenta il normale contesto transazionale delle operazioni che coinvolgono i dati in esse contenuti. Nota: In.de permette di sviluppare sistemi informativi anche senza connessioni ai database. I dati infatti possono essere recuperati da diversi tipi di fonti come, ad esempio, i web service.
Applicazioni: sono le applicazioni che permettono di gestire i dati dei database.
Ogni oggetto applicazione può rappresentare una web application, un web service o un batch service.
Librerie: descrivono i servizi dell'ambiente operativo messo a disposizione dal framework o dagli ambienti di runtime. In.de permette di utilizzare vari tipi di librerie, fra cui referenze a web service, oppure classi precompilate in Java o Microsoft .NET.

3.1.4 Compilazione delle applicazioni

Per capire come personalizzare o estendere le applicazioni realizzate con In.de, occorre entrare nel merito del processo di compilazione delle stesse.
Quando si realizza la costruzione del progetto, per ogni applicazione che deve essere compilata, In.de esegue diversi passi.
Il primo passo consiste nel creare il codice sorgente dell'applicazione nella cartella di output, basandosi sui file del template.
Successivamente l'applicazione viene compilata tramite il compilatore nativo del linguaggio scelto per poi essere inserita all'interno del web server di sviluppo.
Infine, viene aperta una finestra browser che si collega al web server di sviluppo per provare l'applicazione.
Nella generazione del codice sorgente sono coinvolte le directory *Template*, *Custom* e *Output*.
La directory *Template* è il punto di partenza, contiene il modello di tutti i file che la versione generata dell'applicazione dovrà possedere.
La directory *Output* è dove verrà compilata l'applicazione; essa contiene quindi i file del template e il codice sorgente necessario.
Infine, la directory *Custom* indica una directory che contiene i file del template personalizzati per questo progetto. Se quindi si desidera modificare il template, non è consigliabile farlo sui file originali, oppure copiarlo interamente in un'altra directory, ma è preferibile inserire i propri file all'interno della directory indicata come *Custom*.
Si noti che alcune operazioni, come l'importazione di una libreria esterna, creano o modificano automaticamente la directory *Custom* perché la nuova libreria deve entrare a far parte del compilato, altrimenti si otterrebbero errori a runtime.

3.2 Macchina di sviluppo

Durante il tirocinio sono stato equipaggiato con un PC tablet con schermo multitouch della Dell, denominato "Latitude XT2". Esso è stato un riferimento sul target di dispositivo utilizzabile, in maggioranza, dall'utente finale dell'applicazione da me realizzata.

3.2.1 Notebook touch-screen

Latitude XT2 è un sistema ad alte prestazioni, progettato con funzionalità di input intuitive, tra cui penna e schermo multitouch capacitivo, che permette di lavorare nella modalità preferita, fornendo una straordinaria fruizione del tablet. Basta un leggero tocco con la punta delle dita per: scorrere grandi pagine web; attivare lo zoom su immagini grafiche, fotografie o mappe online; ingrandire messaggi di posta elettronica per facilitarne la lettura; chiudere la schermata, collaborare in tempo reale e molto altro ancora. La comodità è assicurata sia in modalità PC notebook che in modalità tablet.

In modalità tablet è possibile utilizzare lo schermo multitouch per una straordinaria efficienza e comodità con le dita, oppure utilizzando la penna senza batteria per scrivere o disegnare a piacimento, come su un normale foglio di carta. In **Figura 4** possiamo vedere Dell Latitude XT2 in questa modalità.

Figura 4 PC tablet Dell Latitude XT2, modalità tablet con penna.

Tutto questo è disponibile anche in modalità notebook, ma con la comodità di una tastiera estesa con doppio dispositivo di puntamento: trackstick e pulsanti. In **Figura 5** vediamo Dell Latitude XT2 in questa seconda modalità.

Figura 5 PC tablet Dell Latitude XT2, modalità notebook.

Capitolo 4. Implementazione dell'applicazione

In questo capitolo verranno descritte alcune fasi di sviluppo dell'applicazione, utili per capire come è stato possibile arrivare ai risultati richiesti in fase di progetto grazie a Instant Developer. Il primo passo affrontato è stata la connessione al database; successivamente la realizzazione delle funzionalità.

4.1 Connessione al database server

Per collegare una qualunque applicazione software ai vari tipi di database server occorrono driver specifici. L'IDE di Instant Developer è un'applicazione scritta in Microsoft Visual C++, quindi richiede driver OLEDB installati sulla macchina di sviluppo.
Dopo l'installazione del client Oracle sulla macchina di sviluppo, per effettuare la connessione al database Oracle è sufficiente quindi impostare e testare i parametri di connessione attraverso una schermata delle proprietà. In **figura 6** è mostrata tale schermata dove si può notare che è anche possibile spuntare i database con i quali si desidera mantenere la compatibilità.
I parametri di connessione al database sono: Tipo, Nome Server, Database, Username e Password.
Una volta creata la connessione al database è sufficiente importarne la struttura attraverso il comando contenuto nel menù contestuale relativo a esso; a questo punto è possibile utilizzare le tabelle importate all'interno dell'applicazione.

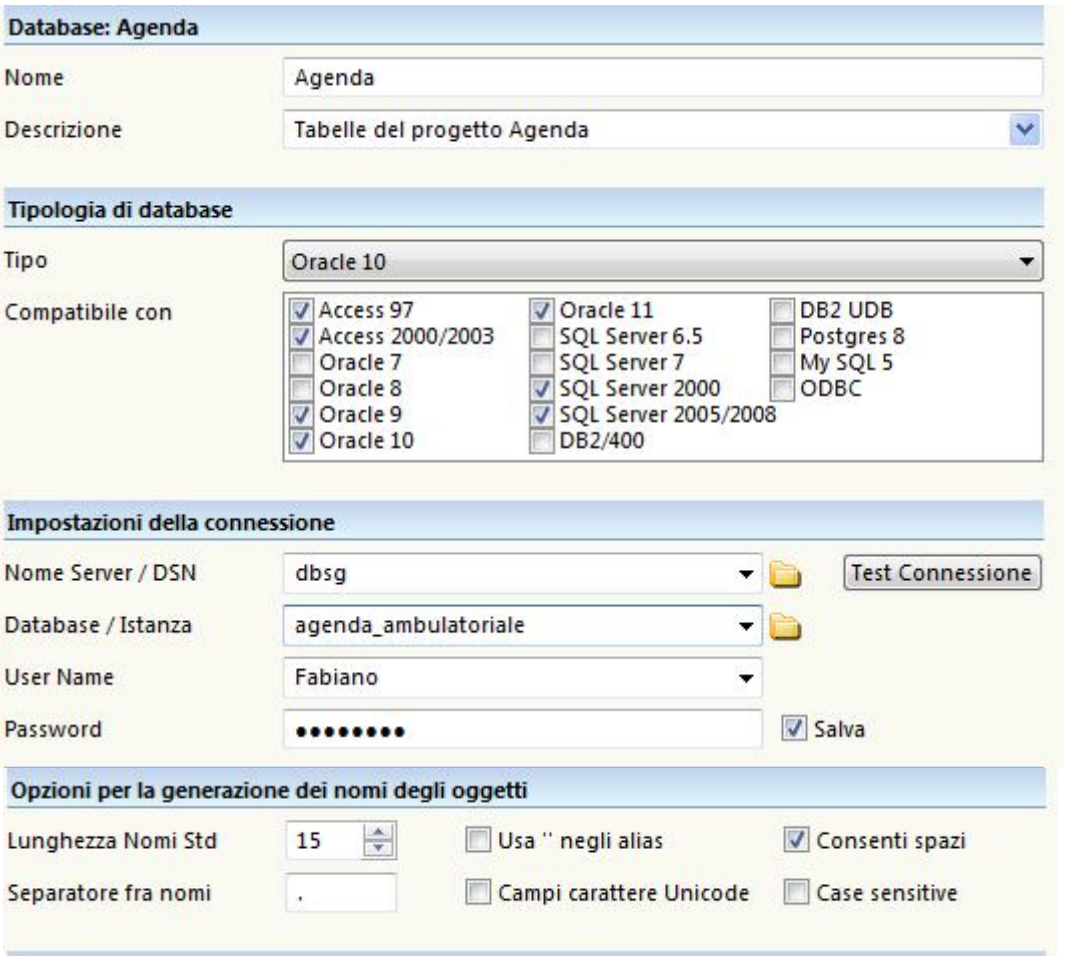

Figura 6 Schermata delle proprietà per la configurazione di un database di In.de.

4.2 Login

Le sessioni in un'applicazione In.De sono gestite come quelle di una qualunque applicazione web: la sessione ha inizio quando il browser si collega per la prima volta al web server e rimane attiva fino a che esso continua a chiamare il server prima della scadenza del timeout di sessione.
Il web server comunica il cookie (stringa di caratteri che identifica in modo univoco il browser client) al browser client e crea un oggetto sessione per ogni browser client che si collega a esso.
Per la gestione delle diverse fasi del ciclo di vita della sessione In.de mette a disposizione diversi eventi notificati all'applicazione web dal framework. Le fasi principali del ciclo di vita della sessione sono la fase di avvio e quella di conclusione.
La fase di avvio della sessione può essere gestita tramite gli eventi *Initialize* (notificato solo la prima volta che il browser si collega al web server), *OnLogin* e *AfterLogin* dell'oggetto applicazione web. Lo scopo primario di questi eventi è quello di capire se il browser che si collega è autorizzato a utilizzare l'applicazione e con quale profilo può farlo. Questo avviene impostando la proprietà dell'applicazione *UserRole* a uno dei ruoli utente definiti nel progetto. Se questa viene lasciata vuota, l'applicazione non parte e viene visualizzata la pagina di login che richiede username e password all'utente. Se quindi nell'evento *Initialize* non viene impostata la proprietà *UserRole*, viene segnalato all'applicazione l'evento *OnLogin* e viene visualizzata la pagina di controllo accesso in cui l'utente può inserire il proprio username e password. Se all'interno dell'evento *OnLogin* viene impostata la proprietà *UserRole* l'applicazione può partire (viene segnalato l'evento *AfterLogin* all'applicazione) altrimenti viene nuovamente visualizzata la pagina di controllo accesso. L'evento *OnLogin* dell'applicazione Agenda è il seguente:

```
event Agenda.OnLogin(
  inout boolean DataValid // Impostare a True per indicare che le credenziali sono valide
  inout string UserName   // User Name inserito dall'utente nella pagina di login
  inout string Password   // Password inserita dall'utente nella pagina di login
)
{
  ...
  select into variables (found variable)
    set username = UTENTE
    set codiceRisorsa = CODRISORSA
    set nome = NOME
    set ruolo = CODGRUPPO
  from
    UTEANAG // master table
  where
    PASSWORD = Password
    UTENTE = UserName
  ...
  // Operazioni se lo username è stato trovato
  if (username <> "")
  {
    Agenda.userName = username
    Agenda.userRole = ruolo
    ...
```

Figura 7 Evento *OnLogin*: verifica delle credenziali dell'utente e in caso di esito positivo impostazione della proprietà *UserRole* dell'applicazione.

È inoltre possibile personalizzare la form di default creta da In.de andando a modificare il file html corrispondente, magari aprendolo con un editor di testo. In **Figura 8** è mostrato come è stata realizzata tale form.

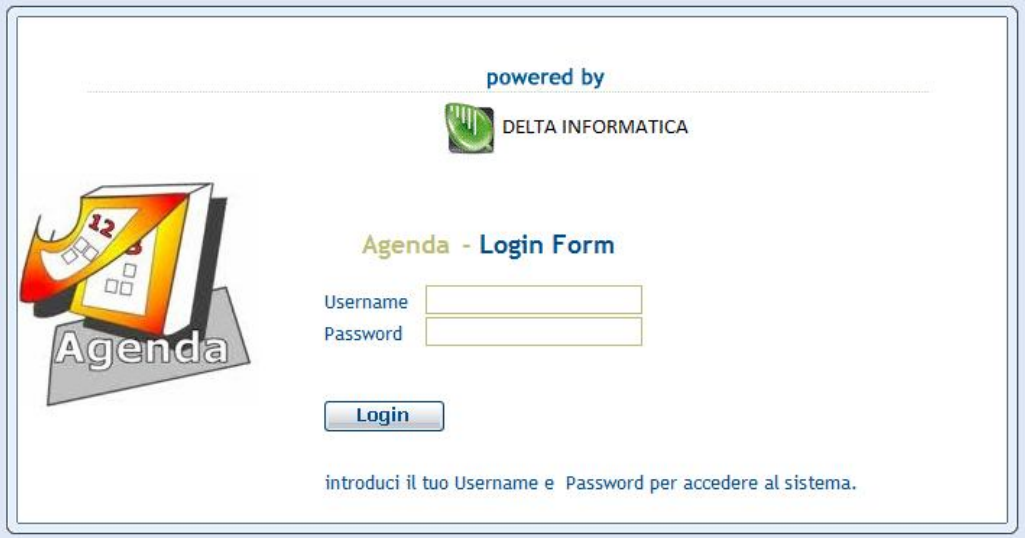

Figura 8 Form di controllo accesso in cui l'utente può inserire il proprio username e password. Rispetto alla form di default sono state modificate le dimensioni, aggiunte le immagini e la frase visibile nel fondo.

4.3 Scelta ambulatorio e relativa attività specifica

Per quanto riguarda la pagina di scelta ambulatorio e relativa agenda In.de mette a disposizione l'oggetto Pannello, ossia un oggetto di interfaccia utente in grado di visualizzare o modificare il risultato di una query di database o in memoria.
A questo proposito, viene gestita una cache locale dei record visualizzati in modo da poter gestire il live scrolling, come avviene nelle migliori applicazioni client server. Anche set di dati dell'ordine delle migliaia di righe scorrono come all'interno di un foglio di calcolo, ma qui i dati sono dall'altra parte della rete.
Un aspetto molto interessante ai fini del progetto è che col pannello, in caso di dispositivi multitouch, molte funzioni possono essere attivate anche con le dita, ad esempio lo scorrimento verticale per scorrere la lista e l'attivazione di una riga specifica.
Il pannello inoltre è un oggetto estremamente flessibile, è infatti possibile modificarne una quantità elevata di proprietà come ad esempio: lo stile e le dimensioni, la scelta dei pulsanti di modifica dati da mostrare, decidere se aprire il pannello in ricerca o visualizzazione dati e altro ancora. Per quanto riguarda lo stile sia il pannello che tutti gli oggetti in esso contenuti supportano l'impostazione di uno stile visuale uniforme per tutta l'applicazione comprese sfumature, trasparenze e bordi personalizzati.
Infine quando si utilizza più di un pannello nella stessa pagina è possibile sfruttare il meccanismo master-detail. Master-detail è un meccanismo di sincronizzazione tra diversi pannelli: la query del pannello detail può referenziare nelle condizioni di filtro il valore della riga attiva dei campi del pannello master. Quando cambia il valore dei campi referenziati, allora viene automaticamente rieseguita la query del pannello detail.
È chiaro che l'oggetto pannello era la scelta migliore per realizzare questa pagina. In particolare utilizzando: un primo pannello per la scelta dell'ambulatorio, dandogli il ruolo di pannello master, e un secondo pannello per la scelta della relativa attività specifica, dandogli il ruolo di pannello detail. Entrambi questi pannelli andranno aperti in modalità di visualizzazione dati.
La query del pannello detail è così implementata:

```
// Query pannello detail "attività specifiche relative all'ambulatorio selezionato"
select
   UNITA.CODUNITA
   UNITA.DESCR
from
   UNITA           // master table
   UNITAGERARCHIA // manually joined, see where clauses
   RISORSEUNITA    // manually joined, see where clauses
where
   RISORSEUNITA.CODUNITA = UNITA.CODUNITA
   RISORSEUNITA.CODUNITA = UNITAGERARCHIA.CODUNITA
   RISORSEUNITA.CODRISORSA = Agenda.DatiSessione.CodiceRisorsa
   UNITAGERARCHIA.CODUNITAMASTER = Ambulatoriautorizzati.CodiceAmbulatorio
```

Figura 9 Query pannello detail: l'ultima condizione di filtro è in base alla riga attiva del pannello master che recupera gli ambulatori autorizzati.

Per fare in modo invece che alla selezione di una attività specifica si apra la vista mese di quest'ultima è necessario modificare il codice dell'evento *OnMouseClick* del pannello relativo. Tale evento è lo stesso che viene notificato in caso si selezioni con il dito. Il codice realizzato dell'evento *OnMouseClick* del pannello detail è il seguente:

```
// ********************************************************************************
// Evento notificato dal pannello quando l'utente esegue un click singolo nel pannello
// ********************************************************************************
event SceltaAmbulatorioEAgenda.Agenderelativeall'ambulatorioselezionato.OnMouseClick(
   int Button // E' un numero intero che rappresenta il pulsante premuto: 0-left, 1-middle, 2...
   int X      // Posizione X in pixel nel pannello alla quale è stato premuto il mouse
   int Y      // Posizione Y in pixel nel pannello alla quale è stato premuto il mouse
   int XB     // Posizione X in pixel nel browser alla quale è stato premuto il mouse
   int YB     // Posizione Y in pixel nel browser alla quale è stato premuto il mouse
   int Column // Indice del campo di pannello che è stato cliccato (-1 se il click è avvenuto...
   int Row    // Indice del della riga del pannello che è stata cliccata, da 0 a VisibleRows ...
   inout boolean Cancel // Può essere impostato a True per non eseguire l'azione di default c...
)
{
   // Se è stata attivato il campo descrizione di una riga del pannello ambulatorio selezionato
   if (Column = Agenderelativeall'ambulatorioselezionato.Descrizione.me())
   {
      // se questo campo non è vuoto
      if (Agenderelativeall'ambulatorioselezionato.Descrizione <> "")
      {
         VistaMese.ApriPerAgenda(
            f(x) data = today()
            f(x) codiceAgenda = Agenderelativeall'ambulatorioselezionato.CodiceAgenda
            f(x) descAgenda = Agenderelativeall'ambulatorioselezionato.Descrizione
         )
      }
   }
}
```

Figura 10 Evento *OnMouseClick* del pannello detail: in caso si selezioni una descrizione non vuota di una attività specifica viene aperta la vista mese chiamando la procedura *ApriPerAgenda* (che come parametri in ingresso richiede il codice e la descrizione dell'attività specifica e una data) e li vengono passati i valori della riga selezionata dall'utente più la data odierna.

In **figura 11** è possibile vedere come è stata realizzata questa pagina.

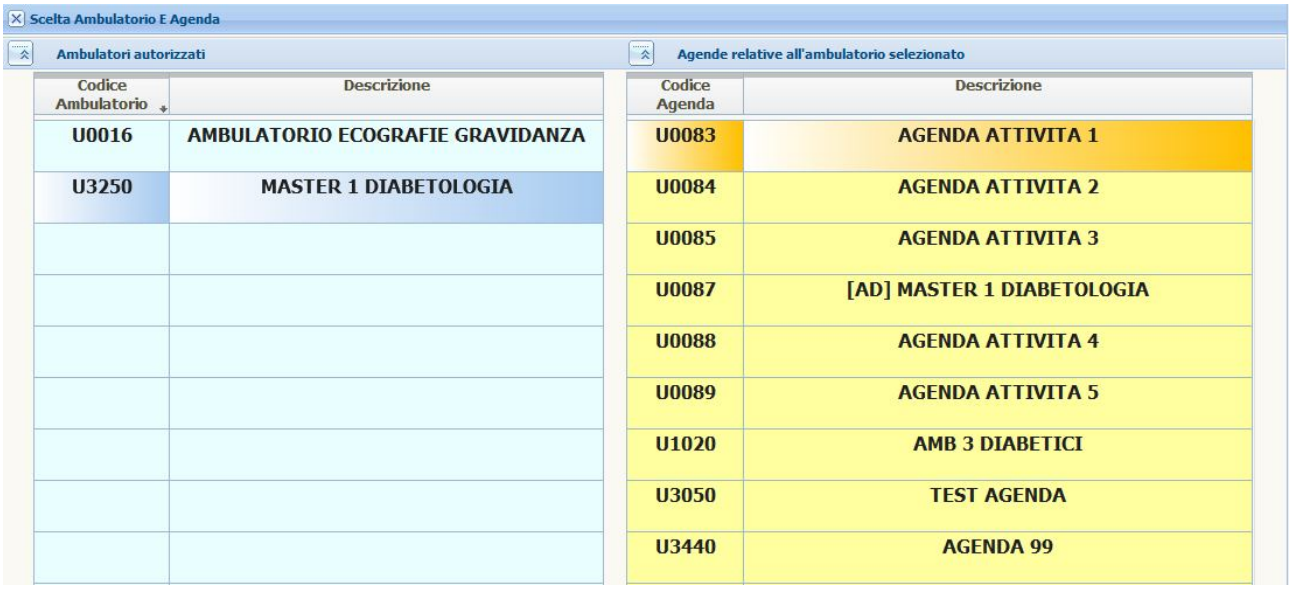

Figura 11 Esempio della pagina di scelta ambulatorio e relative attività specifiche (o agende).

4.4 Vista mese

Per quanto riguarda la vista mese la scelta è caduta su un altro oggetto che In.de mette a disposizione, il libro.
Il libro permette di creare visualizzazioni dati anche molto complesse che poi potranno essere "stampate" in file PDF o mostrate in anteprima nel browser.
È chiaro che per questa specifica la funzione desiderata è quella di usare il libro come oggetto grafico parte dell'interfaccia utente dell'applicazione.
Quando i libri sono utilizzati in questo modo offrono delle funzionalità touch-enabled: se il libro è mostrato in anteprima in un dispositivo multitouch, è possibile strisciare verso destra o verso sinistra per scorrere le pagine. Le dita possono essere usate per effettuare il drag-and-drop o per navigare nella pagina se è più grande dello schermo.
Una volta creato il libro tramite l'apposito comando dell'editor di videate, bisogna definire come strutturare le pagine che formano il libro attraverso l'editor di libri.
Ogni pagina può essere suddivisa in diverse parti, ognuna delle quali è legata a una o più sezioni di un report. Il report è costituito da una query ed è formato appunto da sezioni. La sezione può contenere al suo interno box di diverso genere (etichette, bottoni, link, immagini) e a ogni box si può associare un campo della query del report. Quando si chiama il metodo *Print* sull'oggetto libro, la sezione viene stampata, all'interno della parte di pagina del libro a cui è collegata, tante volte quanti sono i record restituiti dalla query del report; inoltre nelle box contenute nella singola sezione vengono stampati i valori dei campi del record relativo. Se in una parte di pagina non vi è più posto per stampare ulteriori sezioni e le sezioni non sono state tutte stampate, si prosegue la stampa su una nuova pagina che sarà formattata in egual modo.
Nel caso della vista mese il report principale recupera i giorni del mese ed è legato all'intera pagina. I giorni sono formati da una sezione contenente tre box con rispettivamente i valori di: data, numero di slot disponibili e numero di slot occupati. Alla box data è associato il campo data ritornato dalla query principale. I giorni vengono stampati in righe, una riga per ogni settimana, una sotto l'altra, fino all'esaurimento dei record. Le altre due box citate invece contengono ognuna un altro report che recupera il numero di slot, disponibili o occupati, per ogni giorno e sono legati appunto alla sezione che costituisce un giorno. Il risultato ottenuto è mostrato in **Figura 12**.

Figura 12 Esempio di vista mese dell'agenda "Test Agenda". Si può notare che lo sfondo dei giorni cambia a seconda che l'attività specifica non preveda prenotazioni (sfondo grigio), gli slot disponibili coincidano con quelli occupati (sfondo rosso), oppure sia ancora possibile prenotare uno slot (sfondo bianco).

4.5 Visualizzazione referti

Questa pagina è stata realizzata sfruttando l'oggetto libro e la possibilità di gestire gli eventi di drag-and-drop di elementi su altri che offre In.de.
La pagina è formata da due libri: il primo recupera i referti relativi al paziente e li mostra in colonna sulla parte destra dello schermo, il secondo serve da tavola per aprire il referto trascinato.
Attivando le funzionalità *Attiva Drag* sul primo e *Attiva Drop* sul secondo, l'utente a run time, può trascinare le box attivate per il drag, ossia ogni singolo referto della lista, su quelle attivate per il drop, ossia su una delle due metà della tavola. Quando questo avviene, i libri notificano alla schermata l'evento *OnGenericDrag* e *OnGenericDrop* che hanno come parametri gli indici delle due box coinvolte nell'operazione. Ogni indice deve essere confrontato con la proprietà *Me* della box per sapere qual è la box in questione. Al rilascio del referto su una metà della tavola, da codice viene chiamata la procedura che permette di aprire la form che mostra il contenuto dei referti, le cui dimensioni e posizione rispecchieranno il riquadro che delinea mezza tavola. Tale form è costituita da un singolo pannello di cui ne viene sfruttata la funzionalità prevista per i campi BLOB (Binary Large Object). Infatti, il framework di In.de automatizza il trattamento di questo tipo di file, rendendo molto semplice anche la creazione di un sistema di archiviazione documentale: se si crea un pannello a partire da una tabella che contiene un campo BLOB, il campo di pannello corrispondente presenta una toolbar che consente di gestire il file in esso contenuto. Attuando questo procedimento con la tabella di database contenente i file in formato pdf dei referti, e togliendo tutti i campi superflui, è stato possibile mostrare in anteprima nella finestra il contenuto del file.
In **Figura 13** è possibile vedere il drag di un singolo referto della lista. In **Figura 14** è invece possibile vedere il risultato del drop di tale referto sulla parte di tavola.

Figura 13 Esempio di drag di un referto dalla lista, nella pagina di visualizzazione referti.

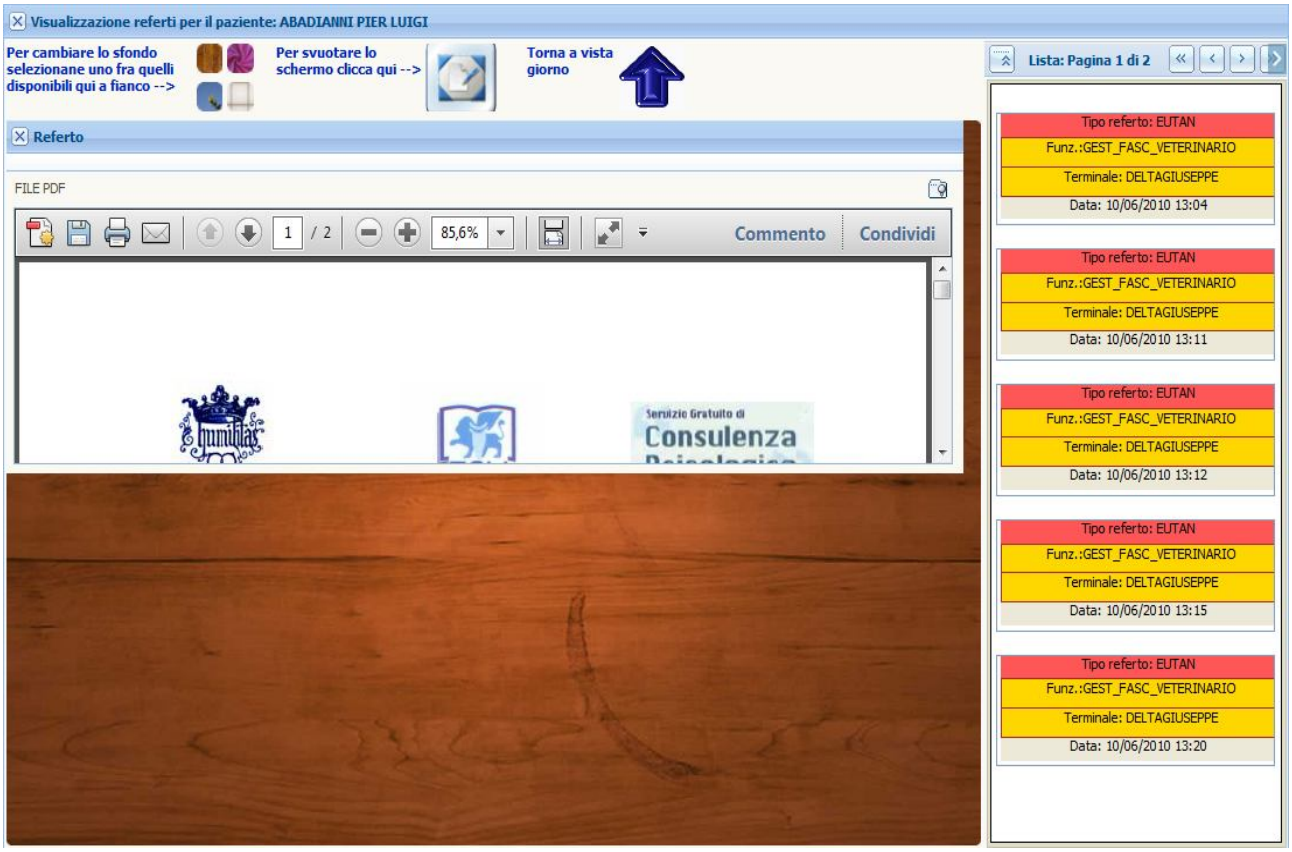

Figura 14 Esempio di drop del referto sulla tavola, ottenuto in seguito al drag di Figura 13.

4.6 Vista giorno

La vista giorno è stata realizzata allo stesso modo della pagina di scelta ambulatorio e relativa agenda ossia tramite due pannelli e l''uso del meccanismo master-detail. La query del pannello master questa volta estrae gli slot disponibili in base all'agenda e alla data selezionata; mentre la query del pannello detail estrae le informazioni del dettaglio della prenotazione in base al codice prenotazione della riga attiva del pannello master.
Inoltre è stata attivata la proprietà *AttivaDrag* sulle righe del pannello master e la proprietà *AttivaDrop* sul bottone che indirizza alla visualizzazione referti.
In **Figura 15** è possibile vedere il risultato ottenuto.

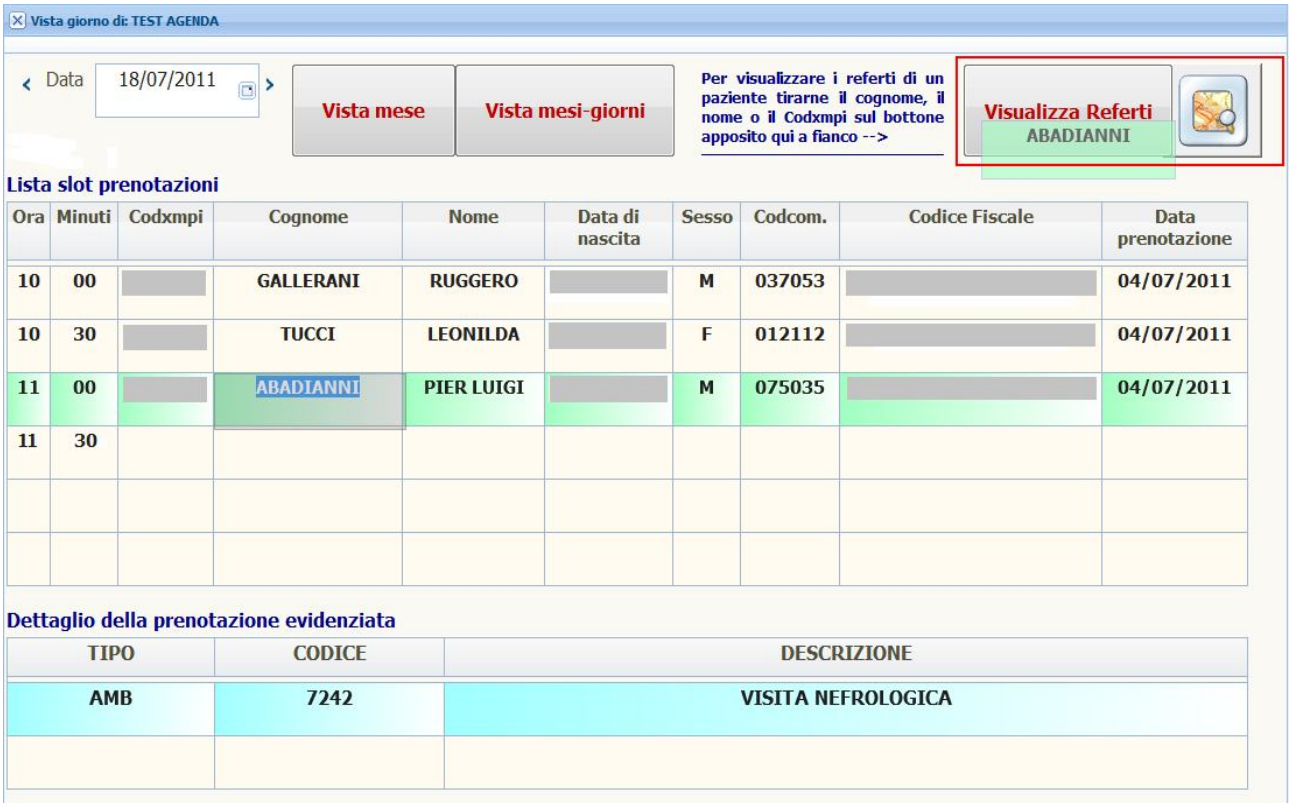

Figura 15 Esempio di vista giorno con drag di un paziente sul tasto visualizza referti. I dati sensibili dei pazienti sono stati oscurati.

4.7 Vista mesi-giorni

La pagina è composta da sette libri: uno per visualizzare i mesi e sei per visualizzare i giorni relativi. Infine, in alto, c'è un ultimo libro che funge da toolbox con bottoni per vedere i restanti mesi dell'anno, oppure per muoversi nei giorni all'interno di uno specifico mese, nel caso si voglia utilizzare il mouse o comunque non si vogliano sfruttare le funzionalità multitouch degli oggetti libri precedentemente descritte. Nella toolbox è inoltre possibile muoversi tra un anno e un altro.
Per rendere attivabile ogni mese e ogni giorno è sufficiente collegare alla sezione del report relativa l'oggetto che si vuole visualizzare in conseguenza alla selezione di quest'ultima (lo stesso procedimento è stato effettuato anche nella vista mese per rendere attivabile ogni riquadro che rappresenta un giorno). Tale oggetto sarà: nel primo caso la vista mese, nel secondo caso la vista giorno, con i relativi parametri passati in ingresso che stabiliscono la data in cui aprire le due viste.
Il risultato ottenuto è mostrato in **Figura 16**.

Figura 16 Esempio di vista mesi-giorni. Si può notare che la data del giorno corrente (16/06/2011) è colorata di giallo, la data dei giorni precedenti di verde e quella dei giorni futuri di blu.

Conclusioni

Questa tesi ha presentato solo alcune delle molteplici funzionalità richieste dal servizio di agenda ambulatoriale nell'ambito del settore sanitario.
Il lavoro che ha portato alla realizzazione di questa tesi ha evidenziato come progettare e analizzare un'applicazione affinché sia fruibile mediante dispositivi multitouch portatili, in modo da sfruttarne le peculiarità che ne stanno decretando il successo.
L'applicazione realizzata migliorerà e renderà più intuibile e usabile la consultazione delle prenotazioni ambulatoriali delle strutture che adotteranno questo software, permettendo di avere sotto mano con pochi tocchi la situazione clinica dei pazienti.
Questo lavoro può essere utilmente esteso a tutte le funzionalità richieste per l'agenda ambulatoriale. Nello specifico si potrebbe aggiungere la possibilità di inserimento, cancellazione e modifica delle prenotazioni affinché il personale medico possa gestire in maniera completa l'agenda. Inoltre, per quanto riguarda la visualizzazione dei referti dei pazienti bisogna effettuare una restrizione di visibilità, in modo da consentire all'utente dell'applicazione la consultazione dei soli documenti attinenti alla propria specializzazione e ai propri campi di accesso.
Il percorso iniziato con la realizzazione dell'agenda ambulatoriale potrebbe in futuro estendersi ad altri servizi offerti nel settore sanitario. Per alcuni di questi servizi il cambiamento sarebbe radicale; basti pensare a tutte le pagine composte da centinaia di campi e decine di comandi per comprendere quanto queste applicazioni debbano essere rivoluzionate.
Durante questo tirocinio ho potuto utilizzare strumenti evoluti per la progettazione e realizzazione di applicativi ad alta interattività, potendo constatare che l'utilizzo di Instant Developer offre diversi vantaggi al programmatore. Infatti, In.de semplifica la programmazione senza mascherarla e l'utilizzo dell'IDE è sostanzialmente simile a quello di altri ambienti di sviluppo presenti sul mercato, alcuni dei quali avevo già avuto modo di conoscere. Questo ha fatto sì che le conoscenze in mio possesso fossero sufficienti a permettermi un facile apprendimento dell'utilizzo di In.de. I vantaggi introdotti da questo ambiente di sviluppo hanno ridotto notevolmente la complessità del lavoro, che è risultata proporzionale alla dimensione del progetto. Questo mi ha permesso di dedicare molta attenzione anche alla rifinitura e alla semplificazione dell'interfaccia grafica, per renderla più gradevole e usabile per gli utenti. Le uniche problematiche riscontrate sono state relative all'utilizzo di alcuni oggetti complessi messi a disposizione dall'IDE, come ad esempio l'oggetto Libro; una volta capito il meccanismo di funzionamento, però, se ne possono apprezzare e comprendere i benefici e le funzionalità. Siccome le tecnologie informatiche hanno un ciclo di vita breve, più corto rispetto alla carriera di un professionista, evitare di perdere molto tempo di lavoro per esplorare tutto ciò che di nuovo si affaccia sul mercato rappresenta un punto di forza di questo prodotto.
La tesi è stata incentrata sul tirocinio svolto presso l'azienda Delta Informatica che mi ha permesso di misurarmi con il loro metodo di lavoro. L'esperienza vissuta in questo periodo è stata decisamente importante e utile specialmente in prospettiva futura. Nonostante alcune difficoltà iniziali di inserimento e adattamento al nuovo lavoro, ambiente e personale, dello stage effettuato mi rimangono impressioni, opinioni e ricordi decisamente positivi. A tal proposito, vorrei ringraziare il signor Massimo Rossi per avermi ospitato e aiutato durante lo svolgimento del tirocinio nella propria azienda.

Riferimenti bibliografici

[Web2.0] Luca Grivet Foiaia: “Web 2.0”, Hoepli Informatica, 2007.
[Wikipedia] Wikipedia®: ”http://it.wikipedia.org/wiki/Rich_Internet_application”, marzo 2011.
[UsabSiti] Michele Visciola: “Usabilità dei siti Web”, Apogeo, luglio 2000.
[ProgWebMobile] Maximiliano Firtman: “Programmare per il Web Mobile”, Hops-tecniche nuove, gennaio 2011.
[In.deGuidaUso] Andrea Maioli: “Instant Developer: guida all’uso”, 2011.