



**Università degli Studi dell'Aquila**  
**Facoltà di Ingegneria**

---

Tesi di Laurea in Ingegneria Informatica e Automatica

# **Studio ed uso di GWT per la realizzazione di un'applicazione web di supporto ad agenzie immobiliari**

**Relatore:**

Prof. Serafino Cicerone

**Laureando:**

Luca Finocchio

---

**Anno Accademico 2010-2011**

# Indice

<b>Abstract</b>	<b>5</b>
<b>1 Introduzione</b>	<b>7</b>
<b>2 Evoluzione del web</b>	<b>9</b>
2.1 Dal web 1.0 al web 2.0 . . . . .	9
2.1.1 Analisi storica . . . . .	9
2.1.2 Caratteristiche del web 2.0 . . . . .	12
2.1.3 Differenze e confronti con il web 1.0 . . . . .	16
2.2 Ajax . . . . .	17
2.2.1 Che cos'è ? . . . . .	17
2.2.2 Confronto con applicazioni esistenti . . . . .	19
2.2.3 Vantaggi e svantaggi . . . . .	20
<b>3 Google Web Toolkit</b>	<b>23</b>
3.1 Introduzione . . . . .	23
3.2 Contenuto del toolkit . . . . .	30
3.2.1 Plugin per Eclipse . . . . .	30
3.2.2 SDK . . . . .	32
3.2.3 Speed Tracer . . . . .	32
3.2.4 GWT Designer . . . . .	34
3.3 Creare un'applicazione GWT con Eclipse . . . . .	35
3.3.1 Organizzazione del progetto . . . . .	37

3.4	Costruire l'interfaccia . . . . .	39
3.5	Gestire gli eventi sul client . . . . .	46
3.6	Comunicazione col server . . . . .	46
3.7	Compilare un'applicazione GWT . . . . .	56
3.8	Librerie esterne . . . . .	58
3.8.1	Maps . . . . .	58
<b>4</b>	<b>Applicazioni di supporto ad agenzie immobiliari</b>	<b>60</b>
4.1	Generalità . . . . .	60
4.2	Confronto tra diverse applicazioni esistenti . . . . .	62
<b>5</b>	<b>Applicazione : Analisi</b>	<b>65</b>
5.1	Specifiche e requisiti . . . . .	65
5.2	Casi d'uso . . . . .	67
5.2.1	Caso d'uso Installazione . . . . .	68
5.2.2	System Sequence Diagram per Installazione . . . . .	69
5.2.3	Caso d'uso Consulta Annuncio . . . . .	70
5.2.4	System Sequence Diagram per Consulta Annuncio . . . . .	71
5.2.5	Caso d'uso Gestisci Annuncio . . . . .	71
5.2.6	System Sequence Diagram per Gestisci Annuncio . . . . .	73
<b>6</b>	<b>Applicazione : Progettazione</b>	<b>76</b>
6.1	Modello di dominio . . . . .	76
6.2	Diagramma delle classi . . . . .	78
6.2.1	Architettura logica . . . . .	78
6.2.2	Entity package . . . . .	79
6.2.3	Controller Package . . . . .	82
6.2.4	View Package . . . . .	84
6.2.5	Foundation package: Client . . . . .	85
6.2.6	Foundation Package: Server . . . . .	86
6.2.7	Database design . . . . .	88
6.3	Design Pattern . . . . .	90
6.3.1	Singleton . . . . .	90
6.3.2	Adapter . . . . .	91

6.3.3	Simple Factory . . . . .	92
<b>7</b>	<b>Applicazione : Implementazione</b>	<b>94</b>
7.1	Comunicazione con il server . . . . .	94
7.1.1	Hibernate . . . . .	95
7.1.2	Esempi . . . . .	97
7.2	Mappa . . . . .	103
<b>8</b>	<b>Conclusioni</b>	<b>105</b>
	<b>Bibliografia</b>	<b>108</b>

# Abstract

Internet, the network of networks, has always been described as a way to separate, in perpetual change, a place where you can search and find everyday items more or less innovative and useful, but always interesting. The web, just for its nature, have never ceased to grow and develop, to attract people to himself and to propose new. Since it was created is in a perpetual stage of improvement and innovation, as well as expansion thanks to the growing increase in people using this tool. The Web 2.0 is part of this growth can be seen, and the final product of years of continuous advancement and development of the web, whose history begins more than 15 years ago. One of the companies promoting these developments, today's leaders in this market and Google. Applications from all known and used by anyone on the network as the search engine <http://www.google.com> maps maps.google.com and much more are the example of next-generation applications based interactivity. The fast progress of this company and its products in the world of Web 2.0 is essentially tied to their software development toolkit GWT stands for Google Web Toolkit. This development framework enables software developers to create web applications by writing code in Java language only, without having to know HTML, CSS, JavaScript and a side language like PHP or Java is the same. In this text we realize a web application support for small real estate agencies. To do this we need to manage locations graphically on maps. The proposed solution will use GWT precisely integrated with some external libraries to interact with maps. In order to create a robust and reusable software we need to develop it according to the Model-View-Controller and manage the entire development through an iterative

incremental. In the early chapters give a general smattering of what has been the evolution of the Web from its very foundation, then move on to define what are the fundamental principles of the toolkit up to requirements analysis, design classes and the most important parts of the application implementation.

Capitolo **1**

## Introduzione

Internet, la rete delle reti, è sempre stato descritto come un modo a sé stante, in perpetuo mutamento, un luogo dov'è possibile cercare e trovare quotidianamente elementi più o meno innovativi e utili, ma sempre interessanti. E' un grande mare dove c'è spazio per tutti: per chi desidera lavorare, per chi svolge approfondimenti e ricerche, per chi punta a realizzare un business, per chi aspira ad instaurare relazioni sociali o divertirsi e anche per chi studia il comportamento degli utenti e visitatori.

Il web, proprio per la sua natura, non ha mai smesso di crescere e di svilupparsi, di attirare a sé gente e di proporre novità. Da quando è stato ideato si trova in un perpetua fase di miglioramento ed innovazione, oltre che di espansione anche grazie al sempre maggiore aumento di persone che utilizzano questo strumento. Il web 2.0 fa parte di questa crescita e può essere visto il prodotto non finale di anni di continuo avanzamento e sviluppo del web, la cui storia inizia oltre quindici anni fa.

Una delle aziende promotrici di questa evoluzione, leader oggi in questo mercato è Google. Applicazioni conosciute da tutti e usate da tutti in rete come il motore di ricerca “<http://www.google.com>”, le mappe “[maps.google.com](http://maps.google.com)” e molto altro ancora sono l'esempio di applicazioni di nuova generazione basate sull'interattività. Il progresso molto veloce di questa azienda e dei suoi prodotti nel mondo del web 2.0 è legato sostanzialmente al loro Toolkit di sviluppo software GWT acronimo di Google Web Toolkit. Questo framework di sviluppo permette agli sviluppatori software di creare applicazioni web scrivendo il codice nel solo linguaggio Java, evitando di dover conoscere HTML, CSS, Javascript e un linguaggio lato serve come PHP o

Java stesso.

In questo testo realizzeremo un'applicazione Web di supporto ad agenzie immobiliari di piccole dimensioni. Per fare ciò avremo bisogno di gestire luoghi graficamente su delle mappe. La soluzione proposta utilizzerà GWT integrato con alcune librerie esterne appunto per interagire con le mappe. Al fine di creare un software robusto e riusabile avremo bisogno di svilupparlo secondo il pattern Model-View-Controller e gestire tutto lo sviluppo mediante un modello iterativo incrementale.

Nei primi capitoli daremo un infarinatura generale su quella che è stata l'evoluzione del web dalla sua nascita ad oggi per poi passare a definire quelli che sono i principi fondamentali del toolkit fino ad arrivare all'analisi dei requisiti, al design delle classi e alle parti più importanti dell'implementazione dell'applicazione.

# Capitolo 2

## Evoluzione del web

In questo capitolo andremo ad analizzare cambiamento del web dalla sua nascita fino alla sua moderna accezione.

### 2.1 Dal web 1.0 al web 2.0

#### 2.1.1 Analisi storica

Il World Wide Web ( “WWW” o semplicemente “web” ) è un mezzo di informazione globale che gli utenti possono leggere e scrivere tramite computer collegati a Internet. Il termine viene spesso erroneamente utilizzato come sinonimo di Internet stessa, ma il Web è un servizio che opera su Internet, come ad esempio il servizio mail.

La storia di Internet infatti, ha radici significativamente precedenti rispetto a quello del World Wide Web. Nel 1980, Tim Berners-Lee, un imprenditore indipendente, presso l’Organizzazione Europea per la Ricerca Nucleare (CERN) in Svizzera, costruì *Enquire*, come un database personale di persone e di modelli di software, ma anche come un modo per giocare con gli ipertesti; ogni nuova pagina di informazioni in *Enquire* doveva essere collegata ad una pagina esistente.

Nel Natale del 1990, Berners-Lee aveva costruito tutti gli strumenti necessari perchè Web lavorasse: l’HyperText Transfer Protocol (HTTP) 0.9, l’HyperText Markup Language (HTML), il primo browser (chiamato WorldWideWeb, che era anche un editor web) , il primo software del server HTTP (più tardi conosciuta

come httpd del CERN), il primo web server (“<http://info.cern.ch>”), e le pagine web che descrivevano il progetto stesso. Il browser poteva accedere ai newsgroup Usenet e ai file FTP. Tuttavia, poteva essere eseguiti solo in NeXT. Fu Nicola Pellow poi, a creare un semplice browser di testo che potesse funzionare su praticamente qualsiasi computer. In linea con la sua nascita al CERN, i primi ad adottare il World Wide Web sono stati principalmente i dipartimenti scientifici universitari o laboratori di fisica, come Fermilab e SLAC. All’epoca però, non c’era ancora nessun browser grafico disponibile per i computer oltre alla NeXT.

Questa lacuna è stata colmata nel mese di aprile 1992 con il rilascio di Erwise, un’applicazione sviluppata ad Helsinki University of Technology, e nel maggio di ViolaWWW, creato da Pei-Yuan Wei, che comprendeva funzioni avanzate come la grafica incorporata, scripting e animazione. Il punto di svolta per il World Wide Web però, è stato l’introduzione del browser web Mosaic nel 1993, un browser grafico sviluppato da un team del National Center for Supercomputing Applications (NCSA) della University of Illinois at Urbana-Champaign (UIUC), guidata da Marc Andreessen. Verso la fine del 1994, anche se il numero totale di siti web era ancora molto più piccolo rispetto agli standard attuali, un buon numero di siti web sono stati creati, molti dei quali sono i precursori o esempi ispiratori dei servizi più popolari di oggi.

Possiamo affermare quindi, che per tutti gli anni 90 il web era composto prevalentemente da siti web statici, senza alcuna possibilità di interazione con l’utente fatta eccezione per la normale navigazione tra le pagine, l’uso delle e-mail e dei motori di ricerca: possiamo definire questo approccio come web 1.0. Le sue caratteristiche principali sono:

- costruzione dei primi portali e siti web;
- organizzazione gerarchica dell’informazione e navigazione attraverso menù;
- interazione tra sito e singolo utente;
- motori di ricerca;
- e-commerce;
- banda limitata;

In seguito, grazie all'integrazione con database e all'utilizzo di sistemi di gestione dei contenuti (CMS), Internet si è evoluta con siti dinamici (come ad esempio i forum o i blog); questo web dinamico è stato da alcuni definito Web 1.5. A partire dal 2002, nuove idee per la condivisione e lo scambio di contenuti ad hoc, come i blog e RSS, hanno guadagnato rapidamente visibilità sul web e attraverso l'utilizzo di linguaggi di scripting come Javascript, degli elementi dinamici e dei fogli di stile (CSS) per gli aspetti grafici, si è arrivati a creare delle vere e proprie applicazioni web che si discostano dal vecchio concetto di semplice ipertesto e che puntano a somigliare ad applicazioni tradizionali per computer : nasce così il web 2.0.

Il termine è stato lanciato dalla prima O'Reilly Media Web 2.0 Conference nell'Ottobre del 2004 e sta ad indicare uno stato di evoluzione del World Wide Web, rispetto alla condizione precedente. Si tende a indicare come Web 2.0 l'insieme di tutte quelle applicazioni online che permettono uno spiccato livello di interazione tra il sito e l'utente. Esempi evidenti sono :blog, forum, chat, wiki, flickr, youtube, facebook, myspace, twitter, google+, linkedin, wordpress, foursquare, ecc. . Successivamente agli inizi del 2006 in un articolo di Jeffrey Zeldman critico verso il Web 2.0 e le sue tecnologie associate come AJAX appare il termine web 3.0: termine a cui corrispondono significati diversi volti a descrivere l'evoluzione dell'utilizzo del Web e l'interazione fra gli innumerevoli percorsi evolutivi possibili.

Questi includono:

- trasformare il Web in un database, cosa che faciliterebbe l'accesso ai contenuti da parte di molteplici applicazioni che non siano dei browser;
- sfruttare al meglio le tecnologie basate sull'intelligenza artificiale;
- il Web Semantico;
- il Geospatial Web;
- e-commerce;
- il Web 3D;
- il Web Potenziato;

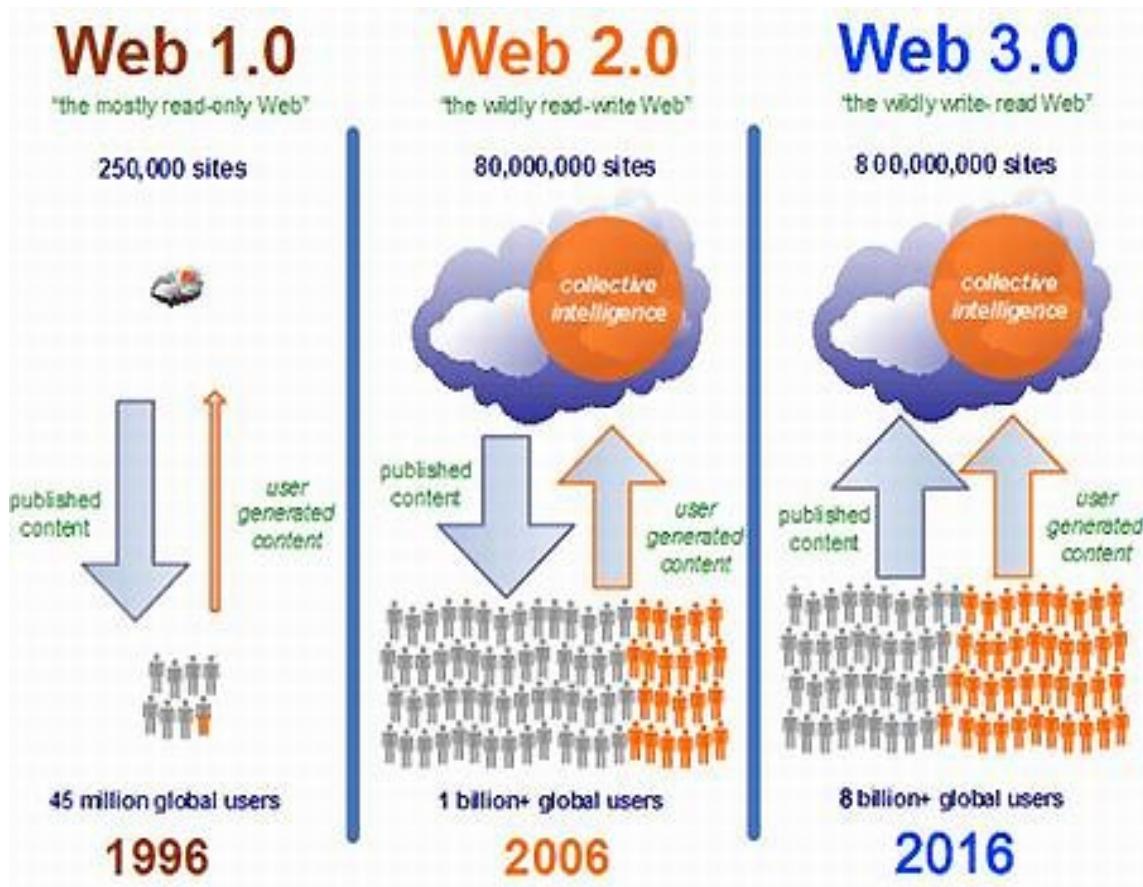


Figura 2.1: Evoluzione del web

### 2.1.2 Caratteristiche del web 2.0

Da un punto di vista strettamente tecnologico, il Web 2.0 è del tutto equivalente al Web 1.0, in quanto l'infrastruttura di rete continua ad essere costituita da TCP/IP e HTTP e l'ipertesto è ancora il concetto base delle relazioni tra i contenuti. La differenza, più che altro, sta nell'approccio con il quale gli utenti si rivolgono al Web, che passa fondamentalmente dalla semplice consultazione (seppure supportata da efficienti strumenti di ricerca, selezione e aggregazione) alla possibilità di contribuire popolando e alimentando il web con propri contenuti. Dal punto di vista funzionale, ciò che caratterizza il web 2.0 è sostanzialmente la centralità ed il protagonismo dell'utente che da fruitore diviene sempre più un controllore dei pro-

pri dati e dei contenuti che naviga, facendosi stesso produttore di informazioni e, contemporaneamente, principale giudice di quanto prodotti da altri.

Dalla comunicazione “da uno a molti” si passa a quella “da molti a molti”. Il Web 2.0 è una nuova visione di Internet che ha appena cominciato ad influenzare il vostro modo di lavorare ed interagire con le informazioni in rete. Il Web 2.0 non è un software specifico, né un marchio registrato dai Microsoft o Google, ma un insieme di approcci per usare la rete in modo nuovo e innovativo. Web 2.0 si riferisce alle tecnologie che permettono ai dati di diventare indipendenti dalla persona che li produce o dal sito in cui vengono creati. L’informazione può essere suddivisa in unità che viaggiano liberamente da un sito all’altro, spesso in modi che il produttore non aveva previsto o inteso. Questo paradigma del Web 2.0 permette agli utenti di prendere informazioni da diversi siti simultaneamente e di distribuirle sui propri siti per nuovi scopi. Non si tratta di derubare gli altri del loro lavoro per il proprio profitto. Anzi, il Web 2.0 è un prodotto open-source, che permette di condividere le informazioni sulle quali è stato creato Internet e rende i dati più diffusi. Questo permette nuove opportunità di lavoro e di informazioni che possono essere costruite sopra le informazioni precedenti.

Web 2.0 lascia ai dati una loro identità propria, che può essere cambiata, modificata o remixata da chiunque per uno scopo preciso. Una volta che i dati hanno un’identità, la rete si sposta da un insieme di siti web ad una vera rete di siti in grado di interagire ed elaborare le informazioni collettivamente. Il Web 2.0 è costruito con tecnologie come Ajax, un approccio di sviluppo web basato su JavaScript ed il linguaggio di programmazione XML. Questa miscela di tecnologie, come detto in precedenza, permette alle pagine di funzionare più come applicazioni per il desktop che come pagine di contenuto statico antiquate che troviamo di solito sul web.

Il web 2.0 prospera su di un’altra bellissima tecnologia di nome RSS. Come molti di voi sanno, gli RSS permettono agli utenti di ottenere aggiornamenti automatici non appena un sito cambia, anzichè controllarlo ogni volta per avere le ultime informazioni. Basta semplicemente iscriversi al feed RSS del sito e non appena il contenuto di tale sito cambia, viene automaticamente inviato al vostro lettore o aggregatore di RSS. Tramite gli RSS, il Web 2.0 inserisce il turbo e viene ampiamente usato per ricercare, filtrare e remixare le notizie, gli articoli ed altri tipi di contenuto in nuovi oggetti di informazione. E’ proprio nel remixare, nella selezione compe-



Figura 2.2: Copertina del Times dell'anno 2006

tente e nella giustapposizione del contenuto e delle informazioni esistenti che risiede il grande potenziale di web 2.0.

<b>Web 1.0</b>	<b>Web 2.0</b>
DoubleClick	--> Google AdSense
Ofoto	--> Flickr
Akamai	--> BitTorrent
mp3.com	--> Napster
Britannica Online	--> Wikipedia
personal websites	--> blogging
evite	--> upcoming.org and EVI
domain name speculation	--> search engine optimization
page views	--> cost per click
screen scraping	--> web services
publishing	--> participation
content management systems	--> wikis
directories (taxonomy)	--> tagging ("folksonomy")
stickiness	--> syndication

Figura 2.3: Differenze tra applicazioni web 1.0 e applicazioni web 2.0

Riassumendo alcuni considerano il web 2.0 come uno slogan, altri come un nome che identifica un grande cambio di paradigma nel web e dalla sua nascita ne sono state date molte definizioni ma Tim O'Reilly e John Battelle ne hanno riassunto le caratteristiche in nove principi chiave :

- il web inteso come piattaforma;
- i dati sono la forza trainante (“i dati sono il prossimo Intel Inside”);
- servizi, non pacchetti software;
- effetti “a rete” generati dalla partecipazione e grazie alla collaborazione degli utilizzatori (architetture partecipative);
- diritto di creare nuove funzionalità attraverso il “remix” di fonti dati distribuite e indipendenti (adozione collettiva e non restrizione privata, “Some rights reserved”);
- la fine del ciclo di approvazione del software (“la beta perpetua”);

- software sopra il livello del singolo dispositivo (il pc non è più l'unico device di accesso alle applicazioni web);
- sfruttamento dell'intelligenza collettiva;
- “the long tail” (i piccoli siti rappresentano il grosso del contenuto Internet, puntare sul customer self-service e sui dati per raggiungere “la lunga coda” e non solo “la grande testa”);

### 2.1.3 Differenze e confronti con il web 1.0

Il Web 2.0 costituisce anzitutto un approccio filosofico alla rete che ne connota la dimensione sociale, della condivisione, dell'autorialità rispetto alla mera fruizione: sebbene dal punto di vista tecnologico molti strumenti della rete possano apparire invariati (come forum, chat e blog, che “preesistevano” già nel web 1.0) è proprio la modalità di utilizzo della rete ad aprire nuovi scenari fondati sulla compresenza nell’utente della possibilità di fruire e di creare/modificare i contenuti multimediali. Sebbene potenzialmente in luce nello stesso paradigma di rete, che si nutre del concetto di condivisione delle risorse, rappresenta la concretizzazione delle aspettative dei creatori del Web, che solo grazie all’evoluzione tecnologica oggi costituiscono una realtà accessibile. La possibilità di accedere a servizi a basso costo in grado di consentire l’editing anche per l’utente poco evoluto, rappresenta un importante passo verso un’autentica interazione e condivisione in cui il ruolo dell’utente è centrale.

Nel descrivere le caratteristiche del Web 2.0 si procede spesso per confronto con il Web 1.0, indicando come nel passaggio di versione gli elementi fondamentali si siano evoluti o siano stati sostituiti da nuovi. Si tratta di un modo di rappresentare il Web 2.0 divulgativo e non prettamente tecnico, ma piuttosto efficace per riconoscere l’evoluzione dei sistemi su Internet. Se prima la costruzione di un sito web personale richiedeva la padronanza di elementi di HTML e di programmazione, oggi con i blog chiunque è in grado di pubblicare i propri contenuti, dotandoli anche di veste grafica accattivante, senza possedere alcuna particolare preparazione tecnica. Se prima le comunità web erano in stragrande maggioranza costituite da esperti informatici, oggi la situazione è completamente ribaltata. I principali produttori di blog sono scrittori, giornalisti, artisti le cui attività non presuppongono una conoscenza informatica

approfondita. La tecnologia Wiki (Wikipedia ne è la più celebre applicazione) è il punto di arrivo del content management, in quanto ne implementa tutti i paradigmi. Se prima erano necessarie più applicazioni informatiche per la gestione del ciclo di vita dell'informazione (dall'intuizione alla fruizione), oggi una stessa tecnologia supporta al meglio tutto il processo.

Si fruisce dell'informazione nell'ambiente stesso in cui essa è nata. Si nota, inoltre, che le tecniche utilizzate fino a ieri per tenere per più tempo i visitatori su un sito web (stickiness, letteralmente l'“appiccicosità” di un sito, cioè la capacità di tenere “incollati” gli utenti ad esso) stanno lasciando il posto ad altre concezioni di contatto con il fruitore. Attraverso le tecnologie di syndication (RSS, Atom, tagging) chi realizza contenuti fa in modo che questi possano essere fruiti non solo sul sito, ma anche attraverso canali diversi. Un esempio di questi nuovi canali sono i feed, cioè delle liste di elementi con un titolo (es. notizie di un giornale, thread di un newsgroup), che permettono il successivo collegamento ai contenuti informativi. Questi ultimi possono essere aggiornati e consultati di frequente con programmi appropriati o anche attraverso i browser e quindi consentono di essere sempre a conoscenza dei nuovi contenuti inseriti su più siti senza doverli visitare direttamente.

## 2.2 Ajax

### 2.2.1 Che cos’è ?

L’acronimo AJAX, che significa esattamente Asynchronous JavaScript And XML (JavaScript asincrono ed XML), è stato enunciato per la prima volta da Jesse Garrett, nel 18 Febbraio 2005, come titolo di un post all’interno del suo blog. Non si tratta di una nuova tecnologia né di un’invenzione bensì di un concetto utilizzato per sviluppare applicativi avanzati e particolari quali Gmail, Google Maps o Google Suggest quindi, come DHTML o LAMP, Ajax non è una tecnologia individuale, piuttosto è un gruppo di tecnologie utilizzate insieme. Le applicazioni web che usano Ajax richiedono browser che supportano le tecnologie necessarie. Questi browser includono: Mozilla, Firefox, Opera, Konqueror, Safari, Internet Explorer e Chrome. Tuttavia, per specifica, “Opera non supporta la formattazione degli oggetti XSL”.

Il concetto di fondo è in parte espresso nell'acronimo scelto, un utilizzo asincrono di Javascript che attraverso l'interfacciamento con XML, può permettere ad un client di richiamare informazioni lato server in modo veloce e trasparente, allargando gli orizzonti delle Rich Internet Applications. Queste applicazioni fino a poco tempo fa erano legate principalmente alle tecnologie Adobe-Macromedia Flash o Java (con le applet). Entrambe purtroppo non sempre interpretabili dai client degli utenti e troppo spesso usate a sproposito con il solo scopo di stupire, discorso che spesso e purtroppo vale anche oggi. In alternativa a queste tecniche di interazione client/server, quando nel 1996 venne introdotto l'iframe in Internet Explorer 3, molti sviluppatori sfruttarono quest'ultimo modificando l'attributo sorgente (src) della pagina racchiusa e simulando così un refresh trasparente di una parte di contenuti il che emulava, in modo abbastanza sporco, un'interazione asincrona. Nel 1998 Microsoft cominciò a sviluppare una tecnologia, chiamata Remote Scripting, con lo scopo di creare una tecnica più elegante per richiamare contenuti differenti ed è in questo periodo, seppur con nome differente, che AJAX venne utilizzato per la prima volta, per poi evolversi in versioni più mature fino a diventare un oggetto vero e proprio, noto ora come XMLHttpRequest.

A seconda del browser usato prende nomi differenti o viene richiamato in maniera differente. Nel caso di Internet Explorer, ad esempio, questo oggetto è restituito da un ActiveXObject mentre nei browsers alternativi più diffusi (Mozilla, Safari, Firefox, Netscape, Opera ed altri) XMLHttpRequest è supportato nativamente, cosa che dovrebbe accadere anche per IE dalla versione 7. Questo oggetto permette di effettuare la richiesta di una risorsa (con HTTP) ad un server web in modo indipendente dal browser. Nella richiesta è possibile inviare informazioni, ove opportuno, sotto forma di variabili di tipo GET o di tipo POST in maniera simile all'invio dati di un form.

Riassumendo, la tecnica Ajax utilizza una combinazione di:

- HTML (o XHTML) e CSS per il markup e lo stile;
- DOM (Document Object Model) manipolato attraverso un linguaggio ECMAScript come JavaScript o JScript per mostrare le informazioni ed interagirvi;

- l'oggetto XMLHttpRequest per l'interscambio asincrono dei dati tra il browser dell'utente e il web server;
- in genere viene usato XML come formato di scambio dei dati, anche se di fatto qualunque formato può essere utilizzato, inclusi testo semplice e HTML preformatto.

### 2.2.2 Confronto con applicazioni esistenti

Le applicazioni web tradizionali consentono agli utenti di compilare moduli e, quando questi moduli vengono inviati, viene inviata una richiesta al web-server. Il web server agisce in base a ciò che è stato trasmesso dal modulo e risponde bloccando o mostrando una nuova pagina. Dato che molto codice HTML della prima pagina è identico a quello della seconda, viene sprecata moltissima banda. Dato che una richiesta fatta al web server deve essere trasmessa su ogni interazione con l'applicazione, il tempo di reazione dell'applicazione dipende dal tempo di reazione del web server. Questo comporta che l'interfaccia utente diventi molto più lenta di quanto dovrebbe essere.

Le applicazioni Ajax, d'altra parte, possono inviare richieste al web server per ottenere solo i dati che sono necessari (generalmente usando SOAP e JavaScript per mostrare la risposta del server nel browser). Come risultato si ottengono applicazioni più veloci (dato che la quantità di dati interscambiati fra il browser ed il server si riduce). Anche il tempo di elaborazione da parte del web server si riduce poiché la maggior parte dei dati della richiesta sono già stati elaborati. Un esempio concreto: molti siti usano le tabelle per visualizzare i dati. Per cambiare l'ordine di visualizzazione dei dati, con un'applicazione tradizionale l'utente dovrebbe cliccare un link nell'intestazione della tabella che invierebbe una richiesta al server per ricaricare la pagina con il nuovo ordine. Il web server allora invierebbe una nuova query SQL al database ordinando i dati come richiesto, la eseguirebbe, prenderebbe i dati e ricostruirebbe da zero la pagina web reinviandola integralmente all'utente. Usando le tecnologie Ajax, questo evento potrebbe preferibilmente essere eseguito con uno JavaScript lato client che genera dinamicamente una vista dei dati con DHTML.

Un altro esempio potrebbe essere la scelta di un nuovo nickname in fase di creazione di un account su un sito web, nel caso classico, se il nome che abbiamo

scelto fosse già esistente, dovremmo compilare prima tutto il modulo ed accorgerci solo dopo aver atteso il caricamento della pagina di conferma che il nome è già esistente e dobbiamo cambiarlo, invece con AJAX può essere introdotto un controllo sull'evento onChange o addirittura OnKeyUp della casella di testo che ci può informare tempestivamente che il nome inserito non è valido, magari evidenziando il testo in rosso (CSS + Javascript). Tutto questo è dovuto al fatto che la richiesta è asincrona, il che significa che non bisogna necessariamente attendere che sia stata ultimata per effettuare altre operazioni, stravolgendo sotto diversi punti di vista il flusso dati tipico di una pagina web. Generalmente infatti il flusso è racchiuso in due passaggi alla volta, richiesta dell'utente (link, form o refresh) e risposta da parte del server per poi passare, eventualmente, alla nuova richiesta da parte dell'utente.

### 2.2.3 Vantaggi e svantaggi

#### Vantaggi

L'impressione di avere a che fare con pagine apparentemente più interattive, dinamiche e professionali è praticamente immediata. L'effetto novità si aggiunge prepotentemente alla lista, stuzzicando la curiosità del navigatore. Questi aspetti apparentemente futili hanno permesso a tecnologie come Flash di affermarsi nel contesto web. Questo perchè l'interesse collettivo è la ricerca di informazioni ma si preferisce navigare dove le informazioni sono presentate nel modo più semplice e veloce possibile.

Ci sono librerie dedicate, facilmente integrabili in applicativi AJAX e capaci di impressionare gli utenti in modo analogo a quello proposto dal plug-in grafico di Adobe, quali moo.fx, gratuita e veramente leggera, oppure lightbox, in questo caso la versione 2.0, pesante ma sicuramente di forte impatto visivo. La compatibilità praticamente totale con i browser più diffusi ed il supporto nativo, o integrabile, dell'oggetto XMLHttpRequest contribuiscono ulteriormente allo sviluppo di Rich Internet Applications, allargando i campi di impiego per gli sviluppatori e coinvolgendo sempre di più i navigatori. Ottenere modifiche dinamiche e leggere sulle pagine, come i suggerimenti sui campi di ricerca, in grado di suggerire parole presenti e velocizzare l'inserimento, mappe, votazioni, statistiche, amministrazione, lezioni, chat, giochi e chi più ne ha più ne metta, ecco il motivo di tanto successo ed ecco

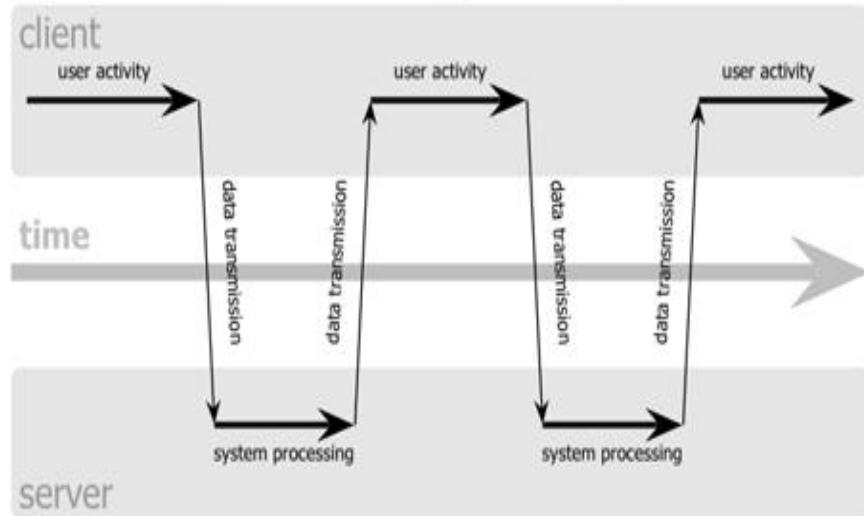
perchè gli utenti possono trarre numerosi vantaggi da AJAX.

### Svantaggi

Si potrebbe non disporre di un browser aggiornato, oppure in situazioni di JavaScript disabilitato, di assenza dell'oggetto XMLHttpRequest. In questi casi si scaricano centinaia di bytes che non verranno mai usati a causa del controllo a posteriori utilizzato per conoscere la reale compatibilità con AJAX. Questi aspetti puramente tecnici sono spesso raggiungibili, grazie ad una stesura degradabile dell'applicativo da parte dello sviluppatore o grazie al caricamento dinamico del Javascript proposto dopo aver verificato la compatibilità del browser. Ma lo svantaggio, probabilmente il più noto nonché fastidioso è sicuramente l'uso dei tasti 'avanti' e 'indietro' presenti nei browser. Qualsivoglia interazione asincrona, da comunque la sensazione di aver cambiato stato alla pagina e l'abitudine a tornare indietro, qualora il risultato non dovesse essere quello sperato, è intrinseca nel navigatore nonchè lecita.

Aprendo direttamente una pagina ricca di interazioni asincrone non sarà possibile, nemmeno volendo, cliccare sul tasto 'indietro' del browser per tornare allo stato precedente, si verrà reindirizzati invece alla pagina precedente, il che può causare disorientamento. In questi casi, la consapevolezza di aver usato AJAX, qualora il navigatore sia preparato, non può certo aiutare, poichè anche l'aggiornamento della pagina non serve a tornare allo stato precedente ma solo allo stato iniziale. Oltre ad essere un vincolo di navigazione, questo problema è anche un vincolo per l'indicizzazione o la possibilità di segnalare ad altri la pagina visualizzata. Questo accade perchè il comando gestito da JavaScript non è portatile come un link e scrivere ad un amico l'indirizzo attualmente visitato ed eventualmente modificato tramite AJAX non è possibile se non dando delle indicazioni precise sulle operazioni svolte una volta arrivati nella pagina in oggetto.

### classic web application model (synchronous)



### Ajax web application model (asynchronous)

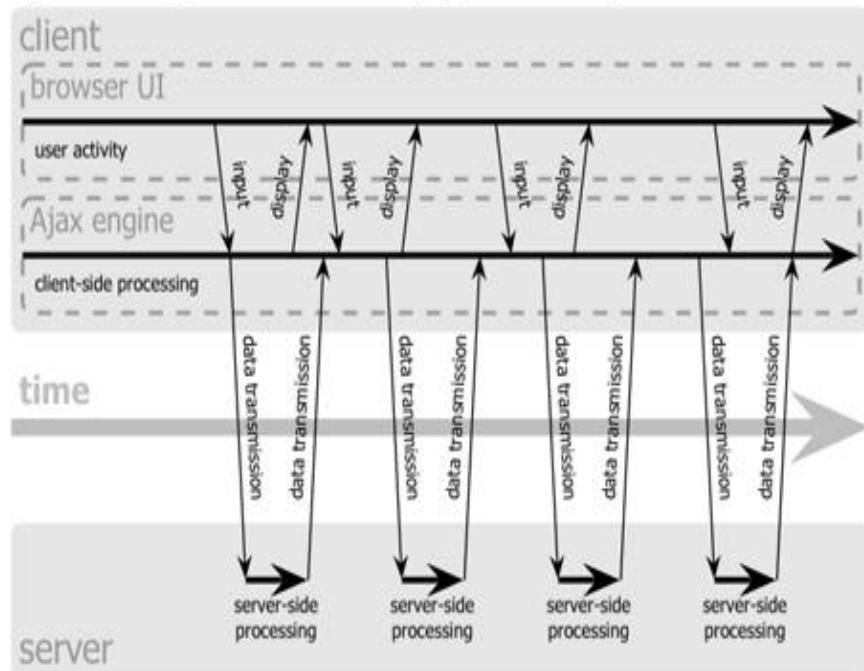


Figura 2.4: Differenza tra il modello di comunicazione delle applicazioni web 1.0 e 2.0

Capitolo **3**

# Google Web Toolkit

Andremo ora ad analizzare uno dei framework che permettono lo sviluppo di applicazioni web 2.0: GWT. Vedremo da cosa è composto il toolkit, ed un piccolo esempio esplicativo della tecnologia. Infine ci sarà una breve panoramica su una delle librerie esterne integrabili con la libreria base del toolkit: le API di Google Maps.

## 3.1 Introduzione

Google Web Toolkit (GWT) è un toolkit di sviluppo per la costruzione e l'ottimizzazione complesse applicazioni basate su browser. Il suo obiettivo è quello di consentire lo sviluppo produttivo delle applicazioni ad alte prestazioni lo sviluppatore web senza dover essere un esperto in cavilli browser, XMLHttpRequest, e JavaScript. GWT è utilizzata in molti prodotti di Google, compresi Google Wave e la nuova versione di AdWords. È open source, completamente gratuito, e utilizzato da migliaia di sviluppatori in tutto il mondo. Inizialmente era molto difficile sviluppare RIA davvero evolute, sia per le solite discrepanze tra i vari browser, sia perché l'HTML non è pensato per questo scopo, e quindi la sua trasformazione in un ambiente dinamico richiede un certo sforzo. Nel corso del tempo, però, si sono affinati vari strumenti che risolvono in gran parte sia la complessità che l'incompatibilità. Questi strumenti si dividono in due gruppi:

- le librerie JavaScript per la manipolazione del DOM (Document Object Model, in pratica l'HTML), e per le chiamate AJAX, la più diffusa delle quali è sicura-

mente jQuery, ma ce ne sono tante altre, come Prototype e l'interessantissima YUI 3;

- i toolkit di componenti di alto livello già pronti, il più delle volte legati a una tecnologia server, come Vaadin (Java), ComponentArt (.Net) ed Ext JS (non legato a tecnologie server).

A seconda del tipo di libreria scelto, lo sviluppo dell'applicazione cambia radicalmente: nel primo caso si creerà un prototipo grafico e funzionale altamente specializzato, pensato su misura per la specifica esigenza del cliente, con possibilità di controllo totale fino all'ultimo pixel, poi si userà, per esempio, jQuery per rendere dinamico l'HTML e dialogare con la parte server dell'applicazione, il cui lavoro sarà solo quello di fornire gli HTML iniziali e i dati successivi (generalmente in formato JSON), oppure i frammenti di HTML aggiornati.

Nel secondo caso, lo sviluppo somiglia sempre più a quello di un'applicazione desktop, dove si utilizzano i controlli disponibili nel toolkit scelto, e la personalizzazione grafica è molto più limitata, infatti spesso si finisce per scegliere uno dei temi forniti dal toolkit stesso, a meno che il progetto non sia abbastanza grande da permettere lo sviluppo di un tema grafico dedicato. La scelta dipende più che altro dal tipo di applicazione che si deve sviluppare: per applicazioni destinate all'utente finale (es. prenotazioni di voli), si sceglierà la prima strada, mentre per applicazioni tipo backoffice (es. mantenimento del catalogo di un sito), si andrà per la seconda. Questo perchè la prima soluzione, nonostante permetta di ottenere il “controllo totale”, ha una crescita esponenziale della complessità, specialmente pensando alla manutenzione e ai successivi sviluppi.

In realtà, però, esiste una terza via: il Google Web Toolkit. GWT è uno strumento creato da Google che va molto oltre ciò che si vede al primo sguardo. Molti pensano che sia solo un modo per creare applicazioni JavaScript senza usare JavaScript, ma i veri motivi per cui esiste non includono l’“antipatia” per il linguaggio del web. In realtà, l’obiettivo era quello di accedere a una delle cose che Java ha più di qualsiasi altro linguaggio al mondo: gli strumenti di sviluppo. Grazie a GWT, è possibile sviluppare applicazioni web con tutta la comodità di Eclipse o NetBeans, compreso il debugging completo, sia lato client che lato server.

In questo modo, sebbene i risultati ottenuti con GWT siano più simili a quelli del primo tipo menzionato in precedenza, la produttività aumenta, e anche applicazioni di dimensioni considerevoli risultano gestibili. In più GWT offre altri vantaggi, come l'ottimizzazione spinta delle dimensioni del codice da inviare al client (e del numero di connessioni HTTP necessarie, visto che gestisce anche CSS e immagini), la possibilità di gestire in maniera agevole le applicazioni multilingua e tante altre. Sviluppando in Java la parte server, poi, è possibile condividere codice tra lato client e lato server: non solo gli oggetti come prodotti, articoli, utenti ecc. possono essere usati tali e quali, ma anche il codice di utilità, di validazione e di business logic più in generale, non è necessario scriverli due volte.

E con GWT RPC anche tutti i valori vengono trasmessi automaticamente. Ne consegue che lo sviluppo di un'applicazione web con Java e GWT porta alla massima produttività, ed è anche possibile aggiungere un ulteriore strato di astrazione utilizzando un toolkit del secondo tipo, come il già citato Vaadin, che è basato proprio su GWT, e che offre degli splendidi componenti già pronti, nonchè varie utilità da usare sul server. Il successo attuale del framework Google Web toolkit è probabilmente dovuto al fatto che più di altri strumenti sembra rispondere alle domande della comunità degli sviluppatori Java EE (e non solo): quale è il modo migliore per realizzare una applicazione basata su Ajax senza dover obbligare lo sviluppatore a lavorare con tonnellate di script JavaScript? Esiste un modo per semplificare la produzione di applicazioni RIA che riduca il set di conoscenze necessarie per gestire questo genere di applicazioni? Infine, nella moltitudine di strumenti oggi necessari (dai vari framework simil JSF, come IceFaces o RichFaces, alle librerie JS) è possibile individuarne uno che si avvicini di più al modo di lavorare solo-Java che non obblighi a dover imparare nuovi linguaggi di script o dover domare applicazioni web affinchè assumano l'aspetto di una moderna applicazione web dalla GUI ricca e asincrona?

Ebbene lo scopo di GWT è proprio quello di rispondere a tutto ciò: offrire al programmatore la possibilità di lavorare direttamente in Java come se stesse realizzando una applicazione stile Swing (ovvero non come una applicazione web ma una standalone) e di trasformare trasparentemente e automaticamente il codice prodotto in una RIA Ajax con tanto di client che lavora in modalità asincrona direttamente dentro il browser.

Realizzare una applicazione GWT risulta particolarmente semplice, se si conosce bene il linguaggio Java (basta Java SE), l'uso della programmazione asincrona multithread (non c'è da preoccuparsi, basta una infarinatura) e se si hanno conoscenze di base sulla realizzazione di applicazioni a interfaccia grafica in Java stile Swing; l'applicativo è scritto tutto in Java e solo dopo la compilazione standard (eseguita dal compilatore javac a riprova che in partenza il codice è tutto e solo codice Java), il compilatore GWT esegue in maniera automatica le necessarie validazioni, convertendo il bytecode Java in script JavaScript quelle parti che dovranno essere eseguite all'interno del browser (come avremo modo di notare non tutta l'applicazione GWT viene eseguita sul client sotto forma di applicazioni Ajax, per cui non tutto il codice viene tradotto in JS).

Fin dalla prima volta che si vede in funzione GWT per la prima si rimane affascinati per il modo con cui i progettisti di Google hanno messo in pratica una idea tanto geniale quanto semplice. Per lungo tempo infatti molti programmati hanno considerato il JavaScript come uno scomodo vicino di casa con cui aver a che fare ma con cui spesso non era possibile evitare di interagire. Le motivazioni di questa "avversione" nei confronti di JavaScript sono molte; di seguito alcune delle più popolari:

- è un altro linguaggio da imparare: dopo aver studiato per lungo tempo Java (o altri concorrenti) ci dicono che per fare delle applicazioni web interattive, accattivanti e moderne è necessario passare a quello che in molti considerano il fratello minore e che bisogna usare complicati framework e librerie;
- anche se esistono librerie e framework che risolvono questo problema, scrivere una applicazione JS di fatto va incontro alle possibili problematiche di cross-compatibilità fra browser (il più delle volte risolte solo grazie alla scrittura di più versioni dello stesso codice);
- alto numero di framework disponibili: scrivere codice partendo da zero è un compito ormai anacronistico. E' preferibile usare uno dei framework disponibili (ce ne sono molti e questo è un bene), la cui scelta spesso non è immediata (e questo sebbene non sia un male, è un fattore da tenere in considerazione perchè "costa" in termini di tempo e di apprendimento);

- no type check: in JS non viene fatto nessun controllo sui tipi e sulle conversioni. Sebbene questo approccio venga visto da alcuni come una gran comodità, ogni vero programmatore sa che è la fonte di ogni male;
- gestione delle librerie poco formale: quando il progetto diventa complesso (pensiamo allo sviluppo di un grosso portale) sorgono spesso conflitti fra librerie di script; in tale contesto la mancanza di un meccanismo forte di type check spesso dà luogo a problemi non facilmente identificabili. Manca a tal proposito un sistema potente di classloading presente invece in Java;

Il lavoro dei progettisti dei vari strumenti web attualmente in voga (vedi ZK, IceFaces, RichFaces e lo stesso GWT) è quello di mascherare o di eliminare del tutto la scrittura del codice JavaScript che debba essere eseguito nel browser. Le strade scelte sono più o meno differenti fra loro. GWT ha intrapreso la soluzione del full-Java: nemmeno una riga di codice JS viene vista e non vi è la percezione (se non per come deve essere organizzato un progetto e per come viene svolta la comunicazione asincrona) che si sta realizzando una applicazione metà Java metà Ajax. A parte gli slogan più o meno trionfalistici che tendono ad amplificare le magiche funzionalità del nuovo strumento di Google, ecco di seguito una serie di punti che possono riassumere le principali caratteristiche del framework:

- tutto Java SE: le applicazioni GWT sono scritte totalmente in linguaggio Java, seguono le convenzioni di base del linguaggio e non richiedono né conoscenze né il supporto di librerie JavaEE. L'applicazione viene scritta e compilata con un comune IDE (esistono plugin per Eclipse, NetBeans e IntelliJ) o tramite l'SDK fornito direttamente da Google (compilatori, emulatori, server embedded e la possibilità di usare alcuni script di Ant). La programmazione di una applicazione GWT non necessita di conoscenze particolari, anche se è utile conoscere le basi della programmazione asincrona multithread e su come realizzare una GUI tramite panels, layout e container. In tal senso chi ha alle spalle un pò di esperienza con Swing si troverà avvantaggiato;
- chiara separazione delle parti: il codice e le risorse sono raggruppate secondo una precisa organizzazione; una applicazione GWT è in definitiva di una

application (il risultato finale infatti è un .war deployabile in un comune web container) che contiene una cartella WEB-INF dove sono collocati descrittori e tutte le risorse JS per poter eseguire la parte client (sono tutte parti che sono aggiunte in automatico dall'SDK o dal plugin scelto alla creazione del progetto). La parte interessante è la suddivisione già prevista dalla specifica, fra le classi che andranno sul server (pubblicate al client tramite una servlet) e quelle che invece verranno tradotte in codice JavaScript per essere eseguito all'interno del client. Un progetto GWT è quindi suddiviso in sottoprogetto server e sottoprogetto client. Le due parti seguono due processi di sviluppo e di deploy differenti;

- comunicazione asincrona GWT-RPC: è forse una delle caratteristiche più interessanti di GWT (insieme alla traduzione automatica da Java a JavaScript) e permette alla parte client Ajax di comunicare con la controparte server side che rimane in esecuzione all'interno del server. Dato che l'applicazione client interagisce con l'utente in maniera asincrona rispetto a tutte le eventuali chiamate verso lo strato server, il framework mette a disposizione un meccanismo di comunicazione client-server che possiamo considerare come la versione object oriented della comunicazione asincrona alla base di una applicazione Ajax. All'interno di una applicazione GWT infatti è possibile utilizzare un meccanismo di invocazione remota simil RMI, ma in cui oltre al modello di invocazione remota a oggetti è presente una efficace e ben progettata semantica asincrona. Il sistema prevede la gestione di notifiche basate su eventi per gestire la risposta asincrona ritornata dal server in modo da aggiornare automaticamente la GUI;
- presenza di librerie di componenti: il framework viene rilasciato con un set di componenti grafici (widget) tramite i quali è possibile realizzare interfacce grafiche piuttosto accattivanti (anche grazie alla alta interattività tipica di una applicazione Ajax). A dire il vero i widget di base di Google GWT non sono molto sofisticati e versatili, tanto che questa carenza ha dato modo a diversi super-framework di estenderne le funzionalità grafiche;
- possibilità di descrivere la struttura della GUI tramite XML: per chi da qualche

anno segue il mondo dell'IT si ricorderà che circa 12 anni fa Netscape presentò lo XUL, un dialetto XML con il quale era possibile descrivere la struttura di una GUI senza codificare nemmeno una riga di codice di sviluppo. I tempi non erano maturi per una soluzione di questo tipo tanto che sono dovuti passare altri 12 anni affinchè qualcuno la riproponesse in maniera concreta e fattiva. Adesso con GWT 2.0 è possibile infatti descrivere la struttura della interfaccia grafica tramite XML senza dover codificare nemmeno una riga di codice Java (e per quanto riguarda la definizione della GUI è codice sempre piuttosto fastidioso);

- automatic splitting: si tratta di un'altra novità della versione 2.0, grazie alla quale il programmatore può marcire il codice al fine di spezzare il programma complessivo in sottoparti in modo da consentire il download delle parti client side JS da eseguire all'interno del browser limitatamente a quel che serve al momento opportuno. Questa cosa ricorda molto quello che fu proposto a suo tempo come “download intelligente delle applet”, anche se le limitate risorse di banda di allora facevano optare quasi sempre per un download completo in un solo bundle;
- ulteriori framework che estendono le funzionalità base: come si è avuto modo di accennare, la creazione di GWT ha messo sul mercato un prodotto veramente potente e innovativo. L'obiettivo dei progettisti è stato fin da subito quello di lavorare sugli aspetti “under the hood” in modo da realizzare una piattaforma di base senza troppi “buchi” o carenze. Questo ha inevitabilmente lasciato spazio a prodotti terze parti (pubblicizzati direttamente sul sito del progetto Google) che pur basandosi sulle specifiche estendono il framework base con set di componenti sofisticati e molto potenti. A tal proposito sono degni di nota il progetto GWT-Ext (ormai dimesso) e il quasi omonimo Ext-GWT (detto anche GXT) entrambi basati sulla famosa libreria JS EXT e il più recente SmartGWT;

## 3.2 Contenuto del toolkit

Google Web Toolkit è un toolkit di sviluppo per la costruzione e l'ottimizzazione complesse applicazioni basate su browser. Il suo obiettivo è quello di consentire lo sviluppo produttivo delle applicazioni ad alte prestazioni senza che sviluppatore web sia un esperto in cavilli del browser, XMLHttpRequest, e JavaScript. GWT è utilizzata in molti prodotti di Google, compresi Google Wave e la nuova versione di AdWords. E' open source, completamente gratuito, e utilizzato da migliaia di sviluppatori in tutto il mondo. Ma cosa contiene esattamente il toolkit ?

- **Plugin per Eclipse:** il plugin per Eclipse fornisce supporto IDE per Google Web Toolkit;
- **SDK:** l'SDK GWT contiene le librerie Java, il compilatore e server di sviluppo. Permette di scrivere applicazioni lato client in Java e distribuirli come JavaScript;
- **Speed Tracer:** Speed Tracer è un'estensione Chrome che consente di individuare problemi di prestazioni nelle applicazioni web;
- **GWT Designer:** GWT Designer consente di creare interfacce utente in pochi minuti con strumenti per il layout intelligente , drag-and-drop e la generazione automatica del codice.

### 3.2.1 Plugin per Eclipse

Il plugin di Google per Eclipse è il modo più veloce per iniziare a sviluppare applicazioni con Google Web Toolkit e App Engine.

Il plugin è compatibile con Eclipse 3.7. Di seguito sono solo elencate alcune delle caratteristiche che il plugin fornisce per accelerare il processo di sviluppo.

#### Web Application Wizard

Creare velocemente applicazioni Web che utilizzano GWT e/o App Engine completamente configurate e pronte per l'utilizzo.

## Configurazioni per l'esecuzione

Eseguire il debug dell'applicazione web in locale utilizzando configurazioni di run completamente personalizzabili.

## Il supporto per Google Web Toolkit

- visualizzazione in modalità sviluppatore: controllare il vostro log di debug e gestire il proprio codice GWT lato server da Eclipse;
- supporto UiBinder: template editor con completamento automatico, la convalida as-you-type di template e proprie classi, wizard per la creazioni assistita;
- riconoscimento delle inline JavaScript (JSNI): completamento automatico per riferimenti Java, e highlighting della sintassi, auto-indetazione, ricerca nel package Java e integrazione del Refactoring;
- validazioni, soluzioni veloci e supporto per il refactoring per mantenere le interfacce RPC in sincronia;
- scorciatoie per il compilatore GWT e possibilità di configurazione dell'interfaccia utente;
- wizard per creare moduli, client bundle, entry-point e le pagine HTML;
- compatibilità con Eclipse for Java EE e progetti realizzati con Maven;
- supporto per testare GWT con JUnit;

## Il supporto per Google App Engine

- facilità di implementare App Engine;
- convalida as-you-type che assicura che il codice sia compatibile con App Engine;
- creare progetti e sviluppare classi JDO automaticamente senza bisogno di compilatori Ant;

### 3.2.2 SDK

Scrivere applicazioni web per più browser può essere un processo noioso e soggetto a errori. Si può spendere il 90% del tempo di lavoro intorno capricci del browser. Inoltre, la costruzione, il riutilizzo, e il mantenimento di grandi componenti di codice JavaScript e AJAX può essere difficile e fragile. Google Web Toolkit (GWT) facilita questo compito, permettendo agli sviluppatori di creare rapidamente e gestire complesse ma altamente performante applicazioni JavaScript nel linguaggio di programmazione Java. Con l'SDK GWT, si scrive il codice nel linguaggio di programmazione Java che GWT poi compilerà in JavaScript altamente ottimizzato che funzionerà automaticamente in tutti i principali browser. Durante lo sviluppo, è possibile ricorrere al ciclo edit - refresh - view come si è abituati con JavaScript, con l'ulteriore vantaggio di essere in grado di eseguire il debug ed eseguire il codice Java riga per riga. Quando si è pronti per il deploy, il compilatore GWT compilerà il codice sorgente Java in file Javascript ottimizzati. A differenza dei minimizzatori JavaScript che funzionano solo a livello testuale, il compilatore GWT esegue anche una completa analisi statica e diverse ottimizzazioni in tutto l'intero codice GWT, e spesso produce codice JavaScript che carica ed gira più veloce dell'equivalente JavaScript scritti a mano. Ad esempio, il compilatore GWT elimina in modo sicuro il cosiddetto “dead code”, pote aggressivamente classi non utilizzate, metodi, campi e anche i parametri dei metodi, per garantire che lo script compilato sia il più piccolo possibile.

### 3.2.3 Speed Tracer

GWT Designer è un Java GUI designer bidirezionale potente e facile da usare che rende molto semplice creare applicazioni GUI GWT senza spendere un sacco di tempo a scrivere codice per visualizzare le forme semplici. Con GWT Designer è possibile creare finestre complicate in pochi minuti. Si utilizza il visual designer e il codice Java verrà generato automaticamente. Si possono facilmente aggiungere i controlli tramite drag-and-drop, aggiungere gestori di eventi per i controlli, modificare le varie proprietà di controlli utilizzando un editor di proprietà, internazionalizzare l'applicazione e molto altro ancora. GWT Designer è costruito come un plug-in per Eclipse e i vari IDE basati su Eclipse (RAD, RSA, MyEclipse, JBuilder, ecc.). Il

plug-in costruisce un albero di sintassi astratta (AST) per navigare il codice sorgente e usa GEF per visualizzare e gestire la presentazione visiva. Utilizzando strumenti di layout WYSIWYG, non è necessario scrivere alcuna riga di codice Java: il codice verrà generato da GWT Designer. Inoltre il codice generato non richiede librerie aggiuntive personalizzate per compilare ed eseguire: tutto il codice generato può essere utilizzato senza aver installato GWT Designer. Questo è in grado di leggere e scrivere quasi tutti i formati e decodificare la maggior parte del codice Java GUI scritto a mano.

L'editor è composto dalle seguenti principali componenti di interfaccia utente:

- **Design View**: la principale area di layout visivo;
- **Source View** : scrivere codice e rivedere il codice generato;
- **Structure View** : composta dal Component Tree e il Property Pane;
  - **Component Tree**: mostra la relazione gerarchica tra tutti i componenti.
  - **Property Pane**: mostra le proprietà e gli eventi dei componenti selezionati.
- **Palette**: fornisce un accesso rapido a strumenti specifici componenti;
- **Toolbar**: fornisce l'accesso ai comandi di uso comune;
- **Context Menù**: fornisce l'accesso ai comandi di uso comune;

L'editor offre anche le seguenti caratteristiche principali:

- **Internazionalizzazione (i18n) / Localizzazione** : creare e gestire i resource bundle;
- **Pannelli personalizzati** : creare componenti personalizzate e riutilizzabili;
- **Factory** : creare classi personalizzate ;
- **Ereditarietà visiva** : creare gerarchie di componente visiva;
- **Gestione degli eventi** : aggiungere gestori di eventi per i componenti;

- **Menù Modifica** : creare visivamente e modificare barre di menù, voci di menu e menu a comparsa;
- **Morphing** : convertire un tipo di componente in un altro;

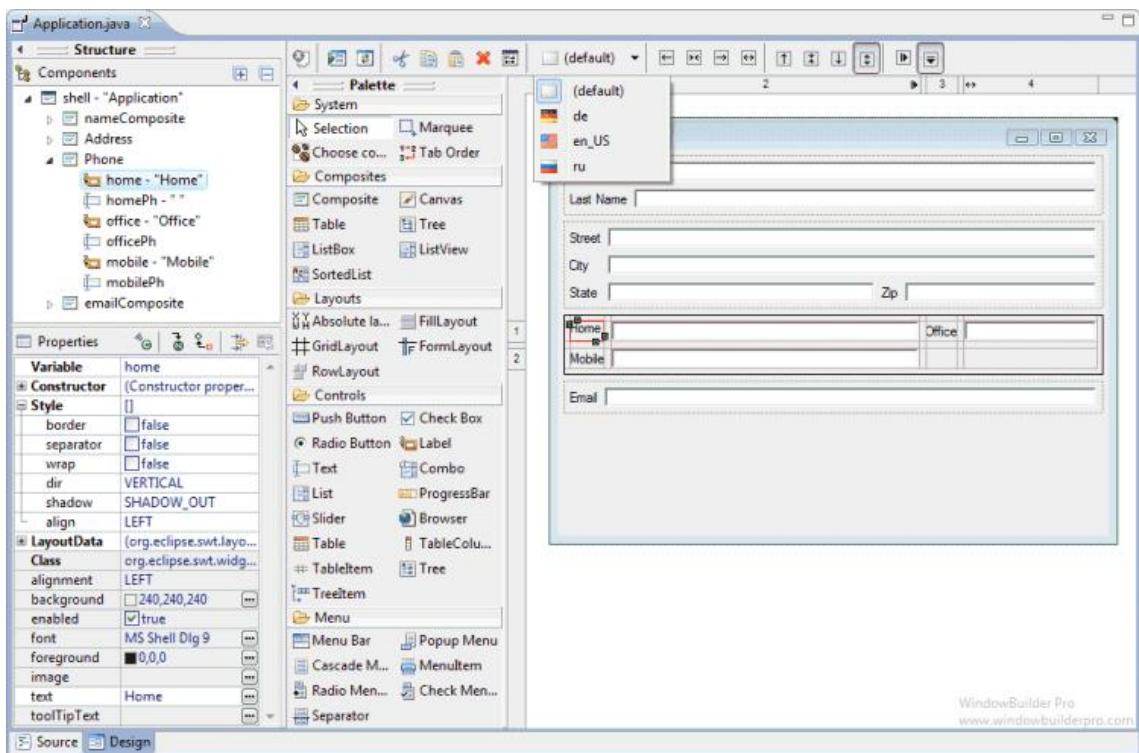


Figura 3.1: Esempio di interazione con il pannello principale di GWT Designer

### 3.2.4 GWT Designer

Speed Tracer è uno strumento per aiutare a identificare e risolvere i problemi di prestazioni nelle applicazioni web. Speed Tracer è disponibile come estensione Chrome e funziona su tutte le piattaforme dove le estensioni sono attualmente supportati (Windows e Linux). Utilizzando Speed Tracer si è in grado di ottenere un quadro più preciso di dove il tempo viene speso nella vostra applicazione. Sono inclusi i problemi causati dal JavaScript parsing e l'esecuzione, il layout, il ricalcolo di

stile CSS e la congruenza di selezione, la gestione degli eventi DOM, risorsa di carico di rete, incendi timer, chiamate XMLHttpRequest, render grafico e altro ancora.

### 3.3 Creare un'applicazione GWT con Eclipse

Uno dei vantaggi derivanti dall'utilizzo di GWT è che è possibile sfruttare gli strumenti, come il refactoring, il completamento del codice e debugging, disponibili in un IDE Java. Il plugin di Google per Eclipse contiene una procedura guidata per la creazione di applicazioni GWT. Ecco i passaggi per la creazione di un'applicazione di avviamento:

1. nella barra degli strumenti, fare clic sul pulsante New Web Application Project;
2. Compilare i dettagli del progetto:
  - (a) inserisci il nome del progetto, ad esempio “helloGWT”;
  - (b) inserire il pacchetto, ad esempio “com.google.gwt.sample”;
  - (c) assicurarsi che “Use Google Web Toolkit” sia spuntato e che “Use default SDK (GWT)” sia selezionato;
  - (d) (Facoltativo) se si utilizza Google App Engine, assicurarsi “Use Google App Engine” sia spuntato e che “Use default SDK (App Engine)” sia selezionato;
  - (e) se non avete installato SDK quando si è installato il plugin di Google per Eclipse, devi fare clic su “Configura SDK ...” per specificare la directory in cui è stato scompattato GWT (e App Engine SDK se necessario);
3. fare clic sul pulsante “Finish”;

Per controllare che tutti i componenti del progetto siano stati creati, è necessario avviare l'applicazione in modalità di sviluppo. In questa modalità, è possibile interagire con l'applicazione in un browser proprio come si farebbe quando è finalmente distribuito. Per eseguire il codice in modalità server di sviluppo bisogna seguire i seguenti passi:



Figura 3.2: Finestra di dialogo per la creazione di un'applicazione GWT

1. nella vista Package Explorer, selezionare il progetto “helloGWT”;
2. nella barra degli strumenti, fare clic sul pulsante “Run” (Esegui come Web Application);
3. quando la scheda modalità di sviluppo si apre, fare clic sull’URL per copiarla;
4. incollare l’URL in un browser a tua scelta (E’ necessario aver installato un plugin nel browser utile per il run in locale delle applicazioni GWT);

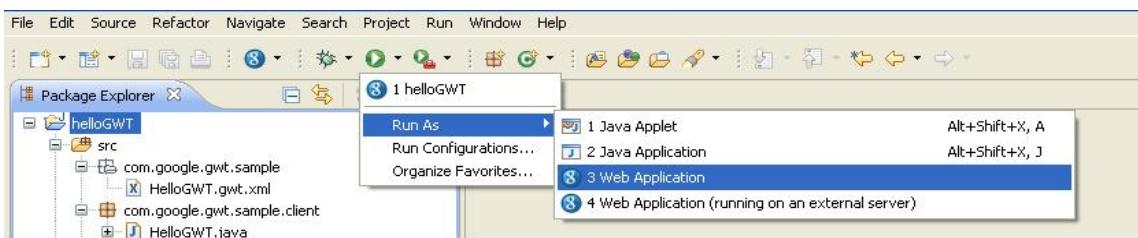


Figura 3.3: Avviamento dell’applicazione in modalità di sviluppo

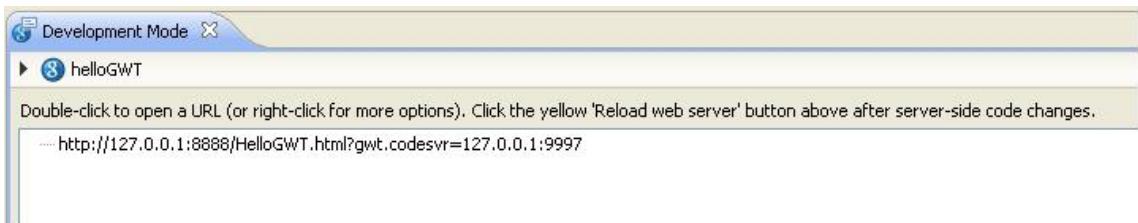


Figura 3.4: URL da copiare nel browser per avviare l’applicazione in modalità di sviluppo

### 3.3.1 Organizzazione del progetto

I progetti GWT possono essere organizzati in una certa varietà di modi. Tuttavia, sono incoraggiate convenzioni particolari atte a rendere più facile identificare quale codice è inteso per l’esecuzione sul browser del client, il server, o entrambi. Questa sezione descrive i principi fondamentali di organizzazione del progetto con GWT e

le convenzioni raccomandate. Usando Eclipse, quest'ultimo crea una serie di file e cartelle, qui sotto elencate.

### Cartelle create

- / **src / com / google / gwt / sample /** : contiene la definizione del modulo GWT e file dell'applicazione;
- / **test / com / google / gwt / sample /** : contiene directory con i file per il test JUnit;
- / **war**: contiene risorse come ad esempio immagini, fogli di stile (CSS), e le pagine HTML;
- / **war / WEB-INF**: contiene i file per il server Java;
- / **war / WEB-INF / lib**: contiene le librerie per il server Java;

### File creati

- **HelloGWT.gwt.xml**: definizione del modulo GWT;
- **HelloGWT.html**: la pagina web;
- **HelloGWT.css**: il CSS relativo;
- **web.xml**: descrizione dell'applicazione java;
- **HelloGWT.java**: classe GWT entry-point;
- **HelloService.java**, **HelloServiceAsync.java**, **HelloServiceImpl.java**: classi che implementano il servizio RPC;
- **gwt-servlet.jar**: libreria GWT runtime per il server;

### 3.4 Costruire l’interfaccia

Le classi di interfaccia utente GWT sono simili a quelli già esistenti come Swing e SWT, tranne che per il fatto che i widget sono resi usando HTML creato dinamicamente piuttosto che tramite la pixel grafica. Nella tradizionale programmazione JavaScript, la creazione dinamica di interfaccia utente viene fatta attraverso la manipolazione DOM del browser. GWT invece, fornisce l’accesso al DOM del browser direttamente utilizzando il pacchetto DOM, è molto più semplice utilizzare le classi dalla gerarchia Widget. Le classi Widget rendono più facile creare rapidamente interfacce che funzionano correttamente su tutti i browser. GWT protegge dal preoccuparsi troppo dell’incompatibilità tra i browser. Se ci si limita all’uso di widget integrati e compositi, le applicazioni funzionano in modo simile sulle versioni più recenti di Internet Explorer, Firefox, Chrome e Safari. (Opera, anche, la maggior parte delle volte). Le interfacce utente DHTML sono molto eccentriche e particolari quindi è bene assicurarsi lo stesso di verificare le applicazioni a fondo su ogni browser. Quando possibile, GWT si rimette agli elementi dell’interfaccia utente nativa dei browser. Per esempio, il widget di GWT Button è un `<button>` HTML vero e proprio piuttosto che un widget sintetico simile ad un bottone da costruire, per esempio, con un `<div>`. Ciò significa che i pulsanti GWT funzionano in modo adeguato sui diversi browser e diversi sistemi operativi. Un aspetto molto positivo è rappresentato dell’uso dei controlli del browser nativo perché sono veloce, accessibile e più familiari agli utenti. Quando si lavora con applicazioni web, CSS è l’ideale. GWT fornisce pochissimi metodi direttamente connessi con lo stile. Piuttosto, gli sviluppatori sono incoraggiati a definire gli stili in fogli di stile che sono collegati al codice dell’applicazione usando nomi di stile. Oltre a separare nettamente lo stile dalla logica applicativa, questa divisione del lavoro aiuta le applicazioni a caricare e rendere più veloce, consumare meno memoria, e anche a renderle più facili da modificare durante i cicli di debug in quanto non c’è bisogno di ricompilare per ritocchi di stile. Tutte le componenti grafiche possono essere raggruppate in due sotto insiemi: i widget di base, e i pannelli per il layout. E’ possibile trovare un’ampia documentazione su widget e panel sia in GWT “showcase”, ovvero una galleria organizzata per funzionalità e contenente molti esempi, sia su JavaDoc incluso nel pacchetto GWT e consultabile anche su internet.

Una volta scelti i widget da inserire nell’interfaccia cominciamo ad inserirne alcuni inizializzando gli attributi della classe HelloGWT. Ad esempio creiamo un oggetto TextBox e un bottone.

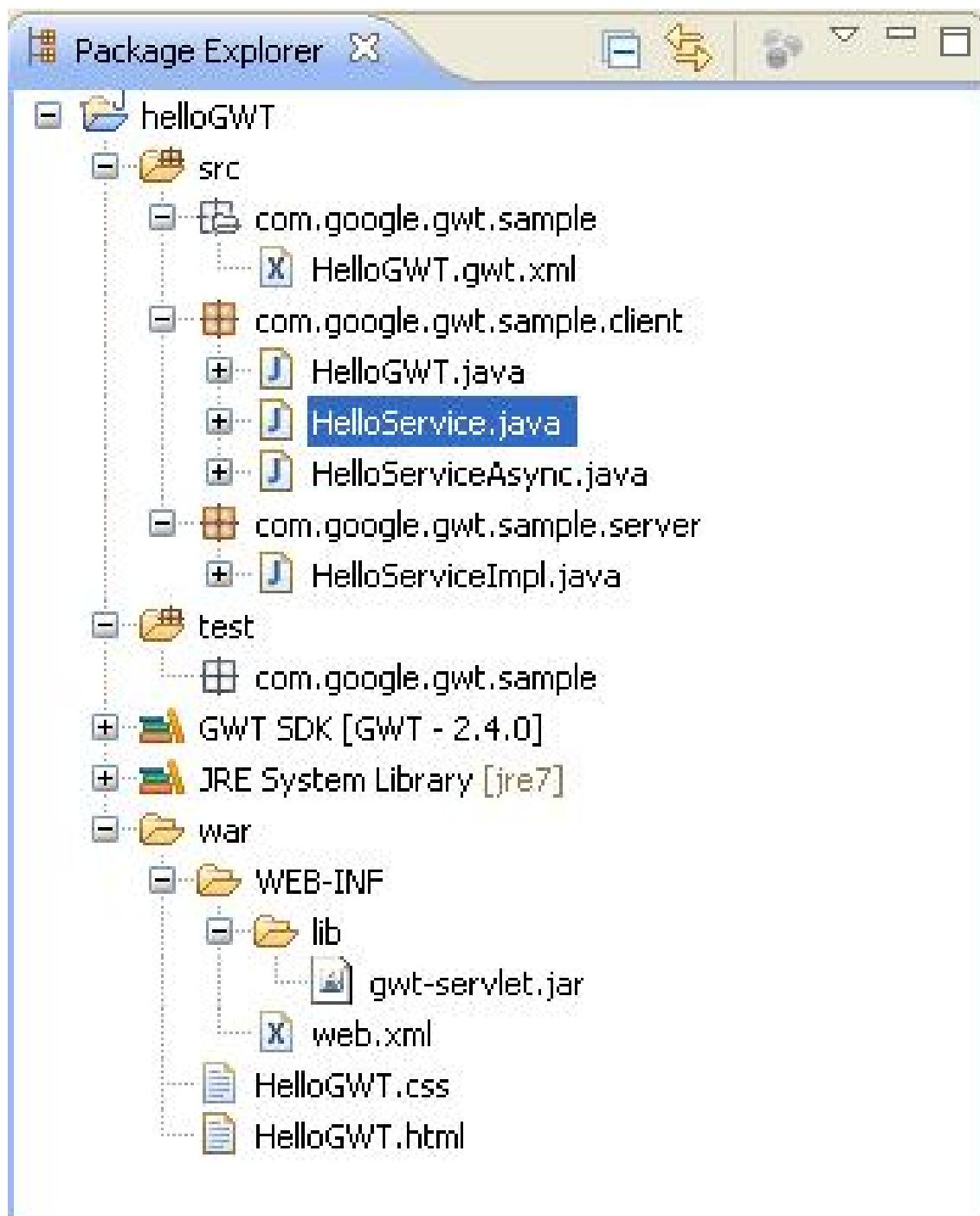


Figura 3.5: Visualizzazione grafica dei package di un progetto

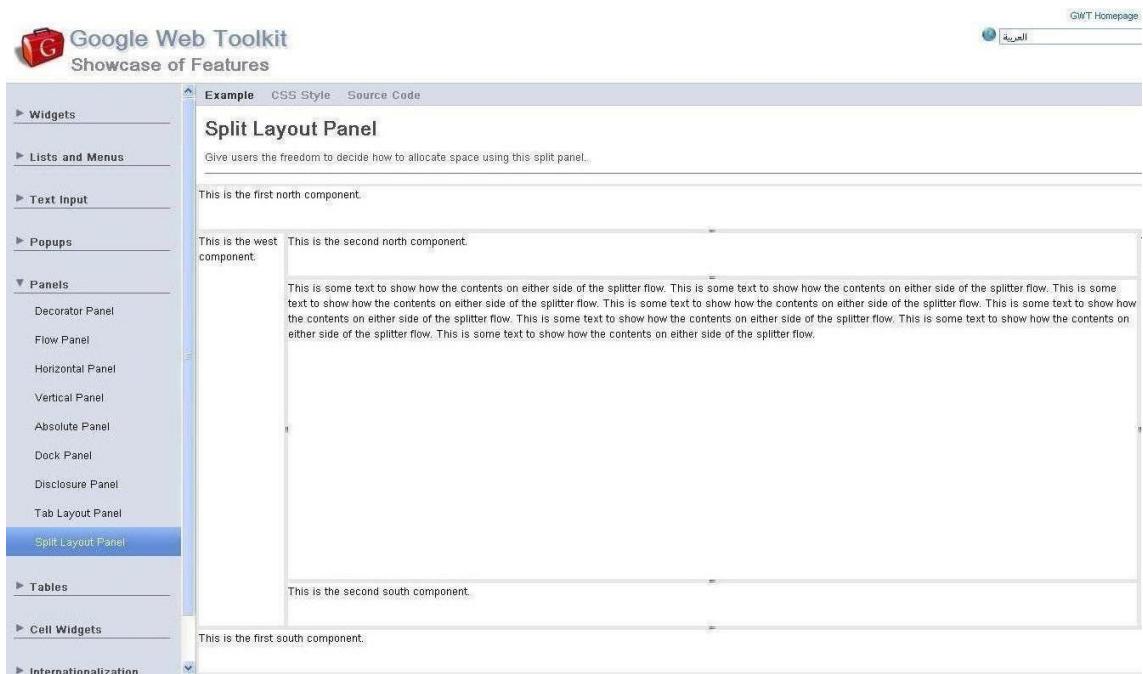


Figura 3.6: Showcase: galleria di widget GWT

```
public class HelloGWT implements EntryPoint {  
  
    private Button send = new Button("send");  
    private TextBox filed = new TextBox();  
    private Label name = new Label("Hello GWT Application");  
    private VerticalPanel vpanel = new VerticalPanel();  
  
    ...  
}
```

Figura 3.7: Codice parziale relativo all’implemtazione del progetto di prova HelloGWT

Decidiamo poi di creare un pannello contenitore con ordinamento verticale in cui inseriamo i widget come segue:

```
public class HelloGWT implements EntryPoint {  
    ...  
    public void onModuleLoad() {  
        vpanel.add(name);  
        vpanel.add(filed);  
        vpanel.add(send);  
        vpanel.add(err);  
        ...  
    }  
    ...  
}
```

Figura 3.8: Codice parziale relativo all’implemtazione del progetto di prova HelloGWT

Per gestire la resa grafica di ogni singolo Widget , GWT permette di definire l’attributo Class o Id da gestire tramite i CSS esterni, oppure gestire gli attributi internamente attraverso metodi propri dell’oggetto.

```
public class HelloGWT implements EntryPoint {  
    ...  
    public void onModuleLoad() {  
        ...  
        send.addStyleName("send-style");  
        name.addStyleName("name-style");  
  
        send.getElement().setId("send-button");  
  
        vpanel.setHorizontalAlignment(HasHorizontalAlignment.ALIGN_CENTER);  
  
        name.setFocus(true);  
    }  
    ...  
}
```

Figura 3.9: Codice parziale relativo all’implemtazione del progetto di prova HelloGWT

Successivamente inseriamo tutto nel DOM, assegnando al body il nostro pannello verticale.

```
RootPanel.get().add(vpanel);
```

Figura 3.10: Codice parziale relativo all’implemtazione del progetto di prova HelloGWT

Aggiungendo altri widget per completare l’interfaccia e inserendo le import all’inizio del file otteniamo:

```
package com.google.gwt.sample.client;

import com.google.gwt.core.client.EntryPoint;
import com.google.gwt.user.client.ui.Button;
import com.google.gwt.user.client.ui.DialogBox;
import com.google.gwt.user.client.ui.HasHorizontalAlignment;
import com.google.gwt.user.client.ui.Label;
import com.google.gwt.user.client.ui.RootPanel;
import com.google.gwt.user.client.ui.TextBox;
import com.google.gwt.user.client.ui.VerticalPanel;

public class HelloGWT implements EntryPoint {

    private Button send = new Button("send");
    private TextBox field = new TextBox();
    private Label name = new Label("Hello GWT Application");
    private VerticalPanel vpanel = new VerticalPanel();
    private Button close = new Button("close");
    private Label message = new Label();
    private DialogBox box = new DialogBox();
    private VerticalPanel dialogvpanel = new VerticalPanel();

    public void onModuleLoad() {

        vpanel.add(name);
        vpanel.add(field);
        vpanel.add(send);

        send.addStyleName("send-style");
        name.addStyleName("name-style");

        send.getElement().setId("send-button");

        vpanel.setHorizontalAlignment(HasHorizontalAlignment.ALIGN_CENTER);

        field.setFocus(true);

        RootPanel.get().add(vpanel);
    }
}
```

Figura 3.11: Codice relativo all’implemtazione del progetto di prova HelloGWT

## 3.5 Gestire gli eventi sul client

Gli eventi in GWT utilizzano un modello simile a quello utilizzato da altri framework di interfaccia utente. Un gestore di interfaccia definisce uno o più metodi che il widget chiama per annunciare un evento. Una classe che desidera ricevere gli eventi di un particolare tipo implementa l'interfaccia gestore associata e poi passa un riferimento a se stesso al widget per sottoscrivere una serie di eventi. La classe Button, ad esempio, pubblica eventi click. L'interfaccia del gestore associato è clickHandler. Vediamo nel nostro esempio come vengono implementati due clickHandler:

## 3.6 Comunicazione col server

Sicuramente, ad un certo punto, la maggior parte delle applicazioni GWT avrà bisogno di interagire con un server di backend. Per questo GWT fornisce un paio di modi diversi per comunicare con un server tramite HTTP. E' possibile utilizzare il framework GWT RPC per rendere trasparente le chiamate al servlet Java e lasciare che sia GWT a prendersi cura dei dettagli di basso livello come la serializzazione degli oggetti. In alternativa, è possibile utilizzare le classi client HTTP di GWT per costruire e inviare richieste HTTP personalizzate.

### Remote Procedure Call

Una differenza fondamentale tra le applicazioni AJAX e le tradizionali applicazioni web HTML è che nelle applicazioni AJAX non è necessario scaricare nuove pagine HTML durante la loro esecuzione. Poichè le pagine AJAX effettivamente funzionano più come applicazioni che girano all'interno del browser, non c'è bisogno di richiedere nuovo HTML dal server e di aggiornare l'interfaccia utente. Tuttavia, come tutte le applicazioni client / server, le applicazioni AJAX di solito hanno bisogno di recuperare i dati da un server. Il meccanismo per interagire con un server attraverso una rete si chiama Remote Procedure Call (RPC). GWT RPC rende più facile per il client e il server scambiarsi gli oggetti Java avanti e indietro su HTTP. Se usato correttamente, RPC vi darà la possibilità di spostare tutta la tua logica dell'interfaccia utente sul client, con conseguente notevole miglioramento delle prestazioni, larghezza di banda ridotta, ridotto il carico del server web, e con un'esperienza utente piacevolmente

```
public class HelloGWT implements EntryPoint {

    private Button send = new Button("send");
    private Button close = new Button("close");
    ...
    public void onModuleLoad() {

        send.addClickHandler(new ClickHandler(){

            @Override
            public void onClick(ClickEvent event) {
                // fai qualcosa
            }
        });

        close.addClickHandler(new ClickHandler(){

            @Override
            public void onClick(ClickEvent event) {
                // fai qualcosa
            }
        });
        ...
    }
}
```

Figura 3.12: Codice parziale relativo all’implemtazione del progetto di prova HelloGWT

fluida. Il codice server-side che viene invocato dal client viene spesso definito come un servizio, quindi l’atto di invocare una procedura remota è a volte indicato come invocare un servizio. Per essere chiari, però, il servizio in questi termini non è lo stesso del “servizio web”, ma un concetto più generale. In particolare, i servizi di GWT non sono legati alla Simple Object Access Protocol (SOAP). Ogni servizio ha una propria piccola famiglia di interfacce e classi di aiuto. Alcune di queste classi, come ad esempio il proxy del servizio, vengono generate automaticamente dietro le

quinte e generalmente non ci si accorgerà mai del fatto che esse esistano. Il modello delle classi di supporto è identico per ogni servizio che si implementa.

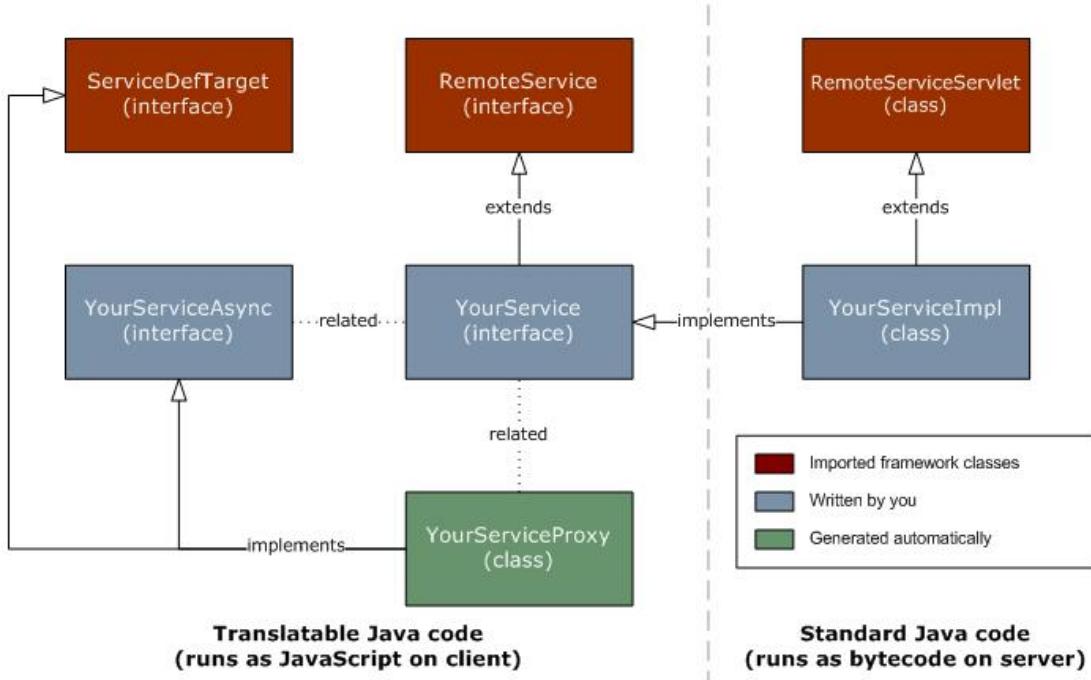


Figura 3.13: Anatomia del servizio RPC

Al fine di definire l’interfaccia RPC, è necessario:

- definire un’interfaccia per il servizio che estende `RemoteService` ed elenca tutti i metodi RPC (figura 3.14);
- definire una classe per l’attuazione del codice lato server che estende `RemoteServiceServlet` e implementa l’interfaccia è stata creata in precedenza (figura 3.15);
- definire un’interfaccia asincrona del vostro servizio che verrà chiamata dal codice sul lato client (figura 3.16);

Si noti l’uso del suffisso `Async` in riferimento alla classe  `AsyncCallback` negli esempi precedenti. La relazione tra un’interfaccia del servizio e la sua controparte asincrona deve seguire determinati standard di denominazione. Il compilatore GWT è influenzato da questi standard di denominazione al fine di generare il codice corretto per implementare RPC. Una interfacce del servizio deve avere un corrispondente

```
package com.google.gwt.sample.client;

import com.google.gwt.user.client.rpc.RemoteService;

/**
 * The client side stub for the RPC service.
 */
@RemoteServiceRelativePath("greet")
public interface helloService extends RemoteService {
    String recuperaDati();
}
```

Figura 3.14: Codice parziale relativo all’implemtazione del progetto di prova HelloGWT

```
package com.google.gwt.sample.server;

import com.google.gwt.sample.client.helloService;

/**
 * The server side implementation of the RPC service.
 */
@SuppressWarnings("serial")
public class helloServiceImpl extends RemoteServiceServlet implements helloService {

    @Override
    public String recuperaDati() {
        return "testo recuperato dal server";
    }

}
```

Figura 3.15: Codice parziale relativo all’implemtazione del progetto di prova HelloGWT

interfaccia asincrona con lo stesso pacchetto e il nome con il suffisso Async aggiunto. Per esempio, se un interfaccia di servizio si chiamasse com.example.cal.client.Spelling Service, l’interfaccia asincrona deve essere chiamata com.example.cal.client.Spelling ServiceAsync. Ogni metodo nell’interfaccia del servizio sincrono deve avere un metodo corrispondente nell’interfaccia del servizio asincrono con un parametro aggiuntivo.

```
package com.google.gwt.sample.client;

import com.google.gwt.user.client.rpc.AsyncCallback;

/**
 * The async counterpart of <code>GreetingService</code>.
 */
public interface helloServiceAsync {

    void recuperaDati(AsyncCallback<String> callback);

}
```

Figura 3.16: Codice parziale relativo all’implemtazione del progetto di prova HelloGWT

vo AsyncCallback come ultimo argomento. Ogni servizio deve in ultima analisi, eseguire alcune elaborazioni per rispondere alle richieste dei client. Tali elaborazioni lato server avvengono nell’implementazione del servizio, che si basa sulla ben nota architettura servlet. Una implementazione del servizio deve estendere RemoteServiceServlet e deve implementare l’interfaccia del servizio associato. Si noti che l’implementazione del servizio non implementa la versione asincrona dell’interfaccia del servizio. Ogni implementazione è in definitiva un servlet, ma piuttosto che estendere HttpServlet, si preferisce estende RemoteServiceServlet. Quest’ultima gestisce automaticamente la serializzazione dei dati passati tra il client e il server e richiama il metodo previsto nella implementazione del servizio. La modalità di sviluppo di GWT include una versione embedded di Jetty che agisce come un contenitore di servlet per il test. Questo consente di eseguire il debug sia del codice lato server che del codice lato client quando si esegue l’applicazione in modalità di sviluppo. Per caricare automaticamente l’implementazione del servizio, bisogna configurare il servlet nel file web.xml. Nell’esempio proposto è stato generato il modulo com.google.gwt.HelloGWT e si è definita l’ interfaccia RPC com.google.gwt.client.HelloService che si è annotata con @ RemoteServiceRelativePath (“greet”). Questa è implementata da un servlet per l’interfaccia tramite la classe com.google.gwt.server.HelloServiceImpl che estende RemoteServiceServlet.

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
        http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
    version="2.5"
    xmlns="http://java.sun.com/xml/ns/javaee">

    <!-- Servlets -->
    <servlet>
        <servlet-name>greetServlet</servlet-name>
        <servlet-class>com.google.gwt.sample.server.helloServiceImpl</servlet-class>
    </servlet>

    <servlet-mapping>
        <servlet-name>greetServlet</servlet-name>
        <url-pattern>/hellogwt/greet</url-pattern>
    </servlet-mapping>

    <!-- Default page to serve -->
    <welcome-file-list>
        <welcome-file>HelloGWT.html</welcome-file>
    </welcome-file-list>

</web-app>
```

Figura 3.17: File web.xml relativo al progetto di prova HelloGWT

Guardando più attentamente il valore di url-pattern si nota che la prima parte deve corrispondere al nome del modulo GWT o, in alternativa, se il modulo è stato rinominato, si può utilizzare il nuovo valore. In entrambi i casi comunque, deve corrispondere alla reale sottodirectory della directory war in cui vive il modulo GWT. La seconda parte deve corrispondere al valore specificato nell'annotazione RemoteServiceRelativePath usata in com.google.gwt.client.HelloService.

Il processo di creazione del servizio RPC dal client comporta sempre la stessa procedura:

- creare un'istanza dell'interfaccia di servizio utilizzando GWT.create();
- creare un oggetto di callback asincrono per essere avvisati quando la chiamata è stata completata;

- effettuare la chiamata;

```
public class HelloGWT implements EntryPoint {
    ...
    private final helloServiceAsync greetingService = GWT.create(helloService.class);

    public void onModuleLoad() {
        ...
        send.addClickHandler(new ClickHandler() {

            @Override
            public void onClick(ClickEvent event) {
                greetingService.recuperaDati(new AsyncCallback<String>() {
                    ...
                    @Override
                    public void onFailure(Throwable caught) {
                        //segnala errore
                    }

                    @Override
                    public void onSuccess(String result) {
                        //recupera i dati e fai qualcosa
                    }
                });
            }
        });
        ...
    }
}
```

Figura 3.18: Codice parziale relativo all’implemtazione del progetto di prova HelloGWT

### Serializzazione

GWT supporta il concetto di serializzazione, il che significa che permette che il contenuto di un oggetto da spostare da un pezzo di codice in esecuzione sia trasmesso a un’altra applicazione o memorizzato al di fuori dell’applicazione per un uso successivo. I parametri del metodo RPC e i tipi restituiti devono essere trasmessi attraverso una rete tra applicazioni client e server e quindi devono essere serializzabili. I tipi serializzabili devono essere conformi a determinate restrizioni. GWT cerca fortemente di rendere la serializzazione più indolore possibile. Mentre le norme relative alla serializzazione sono sottili, in questo caso pratico il comportamento di-

venta molto intuitivo.

### Errori comuni

Di seguito sono elencati alcuni errori che comunemente vengono commessi cercando di far funzionare il servizio RPC:

- quando si avvia l'applicazione in modalità di sviluppo, si vede una eccezione del tipo ClassNotFoundException sulla console, è il server restituisce un errore. Questo significa che molto probabilmente la classe a cui fa riferimento l'elemento servlet nel modulo GWT non in war / WEB-INF / classes. Assicurati di compilare le classi del package server in questa posizione;
- quando si avvia l'applicazione in modalità di sviluppo, si vede un'eccezione del tipo NoClassDefFoundError: com / google / gwt / utente / cliente / rpc / RemoteService sulla console, e il server restituisce un errore. Questo significa che molto probabilmente si è dimenticato di copiare il file gwt-servlet.jar in war / WEB-INF / lib. Se in seguito avrete bisogno di ulteriori librerie lato server, sarà necessario aggiungere le copie di queste anche in war / WEB-INF / lib;
- quando si esegue la chiamata RPC, la modalità di sviluppo mostra un'eccezione del tipo NoServiceEntryPointSpecifiedException: Service implementation URL not specified. Questo errore indica che non si è specificato un @RemoteServiceRelativePath nell'interfaccia del servizio, o che non si è impostato manualmente il percorso di destinazione chiamando la funzione ServiceDefTarget.setServiceEntryPoint ();
- se invocando una chiamata RPC si vede un'eccezione del tipo StatusCodeException 404, il tuo web.xml potrebbe non essere configurato correttamente. In questo caso è bene assicurarsi che sia stato specificato un @RemoteServiceRelativePath e che il campo <url-pattern> specificato nel file web.xml corrisponda con quel valore;

### Richieste HTTP

Se l'applicazione GWT deve comunicare con un server, ma non è possibile utilizzare

i servlet Java sul backend - o se semplicemente si preferisce non utilizzare RPC - è ancora possibile eseguire richieste HTTP manualmente. GWT contiene un numero di classi client HTTP che semplificano, rendendo personalizzate le richieste HTTP al server, l'elaborazione delle risposte in formato JSON o in formato XML. Effettuare richieste HTTP in GWT è molto simile all'effettuarle in qualsiasi lingua o framework, ma ci sono un paio di differenze importanti che dovrebbero conosciute. In primo luogo, a causa del modello a singolo thread di esecuzione delle maggior parte dei browser, lunghe operazioni sincrone come ad esempio chiamate al server possono causare la non risposte dell'interfaccia di un'applicazione JavaScript (e qualche volta del browser stesso). Per evitare problemi di comunicazione di rete o server, GWT permette di effettuare solo chiamate asincrone. Quando si invia una richiesta HTTP, il codice client deve registrare un metodo di callback che gestirà la risposta (o l'errore, se la chiamata non riesce). Per utilizzare i tipi di HTTP in un'applicazione, bisogna ereditare il modulo GWT HTTP aggiungendo il seguente tag <inherits> al file modulo il tuo XML: <inherits name="com.google.gwt.http.HTTP" />.

## 3.7 Compilare un'applicazione GWT

Il cuore di GWT è un compilatore che converte il codice sorgente Java in JavaScript, trasformando l'applicazione Java nell'equivalente applicazione JavaScript. Il compilatore GWT supporta la maggior parte del linguaggio Java. La libreria runtime di GWT emula un rilevante sottoinsieme della libreria di Java runtime. Se una classe JRE o metodo non è supportato, il compilatore emette un errore. E' possibile eseguire il compilatore con il nome del modulo da compilare in una delle seguenti modalità:

- eseguire la classe main com.google.gwt.dev.Compiler utilizzando Java da riga di comando;
- se è stato utilizzato lo script webAppCreator per creare il progetto, è possibile utilizzare per Ant per eseguire il build.xml generato;
- se si utilizza il plugin di Google per Eclipse, è possibile compilare l'applicazione facendo clic sul pulsante Compile GWT Project ;

Una volta completata con successo la compilazione, verranno create le directory contenenti l'implementazione JavaScript del progetto. Il compilatore crea una directory per ciascun modulo si compila.

```
C:\gwt-2.0.0\samples\Hello>ant
Buildfile: build.xml

libs:

javac:

gwtc:
    [java] Compiling module com.google.gwt.sample.hello.Hello
    [java]   Compiling 5 permutations
    [java]     Permutation compile succeeded
    [java]     Linking into war
    [java]       Link succeeded
    [java]   Compilation succeeded -- 20.313s

build:
BUILD SUCCESSFUL
Total time: 22 seconds
```

Figura 3.19: Output del compilatore

Dopo aver eseguito il compilatore GWT la directory war dovrebbe essere simile a questa:

```
C:\gwt-2.0.0\samples\Hello>\bin\find war
war
war\hello
war\hello\18EEC2DA45CB5F0C2050E2539AE61FCE.cache.html
war\hello\813B962DC4C22396EA14405DDEF020EE.cache.html
war\hello\86DA1DCEF4F40731BE71E7978CD4776A.cache.html
war\hello\A37FC20FF4D8F11605B2C4C53AF20B6F.cache.html
war\hello\E3C1ABE83E2E39A126A9194DB727F7742A.cache.html
war\hello\I4A43CD7E24B0A0136C2B8E20D6DF3C0.cache.png
war\hello\548CDF11B6FE9011F3447CA200D7FB7F.cache.png
war\hello\9DA92932034707C17CF15F95086D53F.cache.png
war\hello\A7CD51F9E5A7DED5F65AD1D82BA67A8A.cache.png
war\hello\B8517E9C2E38AA39AB7C0051564224D3.cache.png
war\hello\clear.cache.gif
war\hello\hello.nocache.js
war\hello\hosted.html
war\Hello.html
```

Figura 3.20: Visione generale della cartella war

Nell'esempio qui sopra, war/ hello / hello.nocache.js è lo script che dovrebbe essere incluso in una pagina HTML per caricare l'applicazione. In questo caso, la pagina HTML si trova in war / hello.html e carica lo script di avvio GWT attraverso l'URL relativo ciao / hello.nocache.js. Oltre a questi ci sono un certo numero di altri file generati con l'output del compilatore GWT. Di questi ce ne sono alcuni

che sono fondamentali per la distribuzione dell'applicazione. Chi ha lavorato con GWT prima della versione 1.6, troverà familiari i file nella cartella war / hello sono familiari. L'unica differenza è dove questi file sono generati, e il fatto che la pagina HTML e il CSS non sono nella stessa directory come il resto dei .cache.html / file jpeg. Il percorso in cui vengono generati questi file è controllato modulo XML di GWT.

## 3.8 Librerie esterne

Le librerie API di Google per Google Web Toolkit sono un insieme di librerie che forniscono un legame tra il linguaggio Java e le popolari API JavaScript di Google. Queste librerie rendono semplice e veloce per gli sviluppatori utilizzare le API JavaScript di Google con Google Web Toolkit. Le librerie sono supportate dal team di Google Web Toolkit.

### 3.8.1 Maps

Le API di Google Maps forniscono un modo per accedere alle Mappe Google versione 3 da un progetto GWT senza dover scrivere codice JavaScript addizionale. La versione 3 di questa API è stata appositamente progettata per essere più veloce e più applicabile ai dispositivi mobili, così come alle tradizionali applicazioni browser desktop. Inserire in un progetto GWT l'utilizzo delle Mappe è abbastanza semplice:

- effettuare il download del pacchetto di API;
- dato che si lavora con una libreria, aggiungere gwt-maps3.jar al classpath Java;
- aggiungere al modulo GWT la libreria come segue: <inherits name='com.google.gwt.maps.Maps' />;
- l'applicazione GWT avrà bisogno di accedere alla API di Google Maps, così come all'API key. Per fare questo, è necessario includere un tag <script> nel modulo GWT: <script src="http://maps.google.com/maps/api/js?sensor=false" />;

- includere il tag script indicato nel modulo GWT nel foglio di stile di riferimento generato automaticamente;

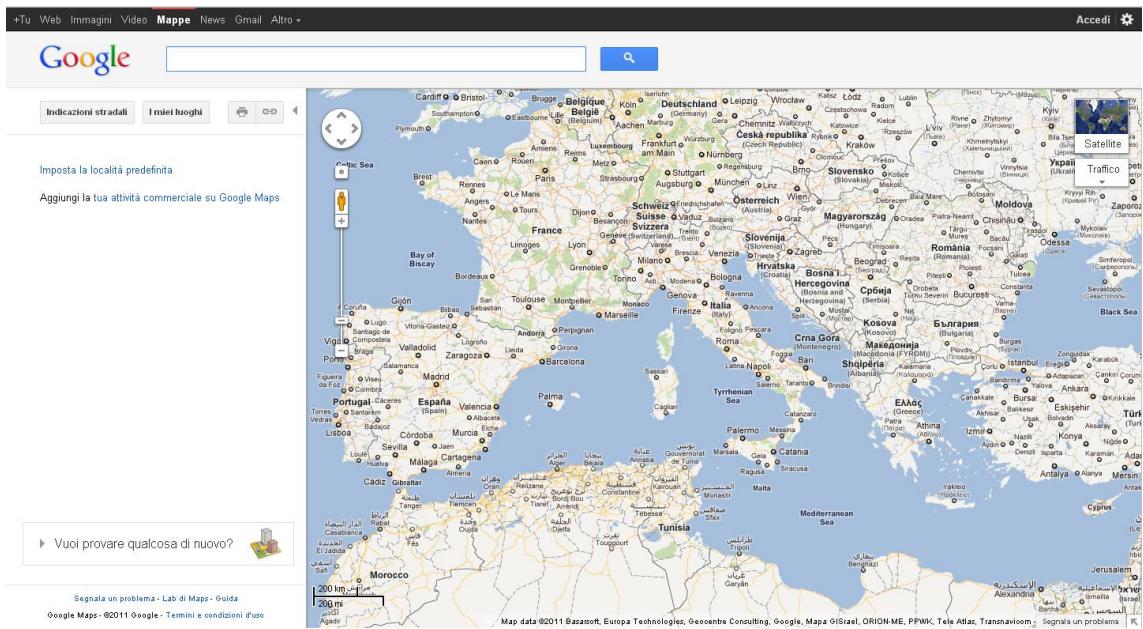


Figura 3.21: Google Maps

# Capitolo 4

## Applicazioni di supporto ad agenzie immobiliari

In questo capitolo andremo a sottolineare il ruolo che il web ha giocato nell'ambito del settore immobiliare prendendo come spunto tre applicazioni esistenti e confrontandole tra di loro.

### 4.1 Generalità

Il web rappresenta sempre più uno strumento fondamentale per il Real Estate, che si sta rapidamente trasformando da nicchia di mercato sul mezzo Internet a uno dei segmenti in più rapida crescita e di maggiore interesse per i navigatori italiani. Nonostante la delicata fase dell'economia globale, il Real Estate Online, sta conoscendo infatti un momento di grande espansione e i contatti sui portali immobiliari continuano a crescere in modo esponenziale, collocando l'area Real Estate tra le categorie web più visitate.

Internet si rivela sempre più un utile strumento a supporto della ricerca di soluzioni immobiliari e offre notevoli benefici sia ad agenti e costruttori, sia ad utenti e potenziali clienti. Da un'indagine Casa.it-Makno emerge che ai fini della ricerca il 54% degli interessati dal processo d'acquisto di un'abitazione ricorre a Internet e il Web si rivela un grande alleato delle agenzie immobiliari proprio in virtù dei vantaggi che è in grado di offrire all'utente finale e che lo rendono sempre più il mezzo

privilegiato per sondare l'offerta del mercato. Gli agenti non possono quindi esimersi dall'essere presenti in Rete ed è fondamentale che acquisiscano consapevolezza delle potenzialità del Web e delle possibili sinergie del mondo immobiliare, aprendosi all'innovazione per migliorare il proprio posizionamento di mercato e cogliere nuove opportunità di business. Ma quali sono effettivamente i benefici del canale online?

- **Convenienza:** il costo contatto è molto più competitivo della carta stampata;
- **Tracciabilità:** il web è molto più controllato e permette di aver chiara visibilità dei risultati di un'inserzione;
- **Presenza continuativa:** gli annunci online sono sempre consultabili, 24 ore su 24, 7 giorni su 7;
- **Aampiezza dell'audience e target mirato:** il web consente di entrare in contatto con un pubblico non solo vasto e diversificato, raggiungendo praticamente chiunque ovunque e in qualunque momento, ma anche realmente interessato, rivolgendosi a un target selezionato con esigenze in linea alle caratteristiche dell'immobile oggetto dell'annuncio;
- **Visibilità delle inserzioni:** gli annunci travalicano i confini del singolo portale immobiliare e grazie alla natura ipertestuale del web, possono essere facilmente disponibili su altri siti. Inoltre un portale immobiliare affermato può amplificare la visibilità delle inserzioni, cercando di posizionarsi al meglio sui motori di ricerca e sviluppando partnership in esclusiva con i principali portali generalisti;
- **Ricchezza di informazioni e aggiornamento costante:** gli agenti immobiliari possono descrivere nel dettaglio gli immobili e inserire foto, video, piantine, mappe e commenti per rendere più accattivanti i propri annunci, che possono essere facilmente aggiornati, modificando o aggiungendo particolari ed eliminando offerte non più disponibili;
- **Interattività:** gli agenti immobiliari possono relazionarsi in modo diretto e immediato con gli utenti e possono servirsi degli strumenti del Web 2.0 (Facebook, Twitter, Blogs, ecc.) per dialogare con loro;

- **Passaparola:** grazie alla semplicità e immediatezza delle interazioni, le reti di relazioni fondamentali nel mondo immobiliare acquisiscono sul web una portata senza precedenti. Gli utenti soddisfatti diventano facilmente portavoce della qualità del servizio e possono attivare un efficace passaparola.

Le agenzie immobiliari che desiderano crescere e competere con successo nell'epoca di Internet devono quindi prendere coscienza di questi benefici e informazione e formazione sono essenziali per porre le basi di un cambiamento culturale che possa contribuire al rilancio del mercato Real Estate. Lo sviluppo del mercato passa proprio dalla consapevolezza delle opportunità offerte dalla rete e dall'utilizzo strategico e business-oriented degli strumenti del Web 2.0 (tool multimediali, blog, social network e community).

## 4.2 Confronto tra diverse applicazioni esistenti

Il web è stracolmo di applicazioni web che sono di supporto ad agenzie immobiliari e generalmente ognuna offre un servizio leggermente diverso. Possiamo distinguere quindi tre grosse categorie di applicazioni che si differenziano soprattutto per il tipo di servizio offerto e la grandezza dell'agenzia che rappresentano. Per approfondire meglio questo concetto prenderemo in considerazione tre applicazioni attualmente esistenti sul web: “<http://www.immobiliare.it>”, “<http://www.tecnocasa.it>” e “<http://www.immobiliareilpanda.it>”. Possiamo definire il primo caso il più generale possibile in quanto, non si è vincolati ad un'agenzia immobiliare in particolare ma piuttosto l'applicazione permette a qualsiasi di esse di iscriversi, pagando un canone mensile.

Ogni agenzia iscritta a questo punto può inserire tutti gli annunci che desidera. L'utente che accede all'applicazione può cercare un annuncio e/o contattare direttamente l'agenzia che lo ha inserito. Sempre pagando, anche un utente può inserire un annuncio. In questo caso specifico all'agenzia vengono anche offerte la possibilità di creare un mini-sito proprio oltre che altri servizi di diversa natura. Nel secondo caso, invece, stiamo considerando un'agenzia immobiliare vera e propria di grosse dimensioni tanto da essere divisa in filiali. In questo caso il servizio offerto è più limitato in quanto gli annunci presenti nell'applicazione saranno solo quelli che gestisce

l'agenzia. L'utente anche in questo caso può visionare gli annunci e a contattare l'agenzia se interessato. L'ultimo esempio proposto, invece, si riferisce invece ad una singola agenzia immobiliare di piccole dimensioni. Questo caso è analogo al precedente, con l'unica differenza che l'agenzia non è divisa in filiali e quindi sarà tutto centralizzato su di essa.

Come detto in precedenza ogni applicazione offre un servizio leggermente diverso, ma nessuna può prescindere da quelle che sono le funzionalità base. Una di queste, sicuramente la più importante è quella della ricerca. In immobiliare.it, ad esempio, viene implementata tramite una cartina dell'Italia cliccabile, molto intuitiva e funzionale per lo scopo. Altre funzionalità fondamentali sono quella della gestione degli annunci e la possibilità di contattare l'agenzia.

Accanto a queste ci sono diverse funzionalità, più o meno utili, che possono fare da sfondo a quelle principali. Un chiaro esempio di questo è offerto da sempre da Immobiliare.it che mette a disposizione un area del sito dedicata allo scambio di informazioni riguardanti il campo della compravendita di immobili. Altri servizi generalmente offerti possono essere quello del calcolo del mutuo o del prestito e l'internazionalizzazione.



Figura 4.1: immobiliare.it

## Capitolo 4

The screenshot shows the Tecnocasa website homepage. At the top, there's a navigation bar with links to the website, Area Affiliati, Glossario, Lavora con noi, Contatti, and MY Home. The main header features the Tecnocasa logo with the tagline "GROUP". Below the header, there are four main sections: "IL TUO IMMOBILE" (with a smartphone icon), "IL TUO FINANZIAMENTO" (with a house icon), "PROGETTO CASA" (with a person icon), and "PROGETTO IMPRESA" (with a factory icon). Each section contains descriptive text and links. Below these are two large forms: "Calcola il tuo mutuo" (Calculate your mortgage) and "Compro" (Buy). The "Calcola il tuo mutuo" form has fields for Type of mortgage (Tasso Variabile), Capital requested in € (100000), and Duration in years (5). The "Compro" form has fields for Destination Immobile (Immobile residenziale), Comune (Fornaci di Barga), Località con immobili (Eglio), Tipologia immobile (Tutte), and Prezzo € (0). Both forms have "CALCOLA" and "CERCA" buttons.

Figura 4.2: tecnocasa.it

The screenshot shows the Immobiliare il Panda website homepage. The header features the logo "il Panda" with a cartoon panda character. Navigation links include home, chi siamo, modulo contatti, english version, area riservata, and BACHECA ULTIMI ARRIVI. The main content area has a yellow gradient background. On the left, there's a sidebar with categories: APPARTAMENTI, VILLE, VILLE BIFAMILIARI, RUSTICI/COLONICHE, TERRENI, and COMMERCIALI. Below this is the FIMAAC logo. In the center, there's a large image of a rustic stone building with green doors and windows. To the right of the image, there's descriptive text about the property (Rif. R/94, Tipologia: RUSTICO, Località: MOLAZZANA/EGLIO, Prezzo: 100000-250000), a "scheda completa" link, and a "DOVE SIAMO" section with maps. At the bottom, there's contact information for Agenzia Immobiliare Il Panda (Via dell'Angeletto 1/B, Fornaci di Barga 55052 Lucca, Tel/Fax (+39) 0583-75149, Cel. 348-3701027) and an email address (info@immobiliarelpanda.it). There are also links to various websites and services like goutos.com, clickasa.it, dbcasa.net, and casattiva.it.

Figura 4.3: immobiliareilpanda.it

Capitolo **5**

## Applicazione : Analisi

In questo capitolo e nei successivi considereremo la progettazione e la realizzazione mediante il toolkit GWT di un'applicazione software su architettura web di supporto ad agenzie immobiliari. Vedremo in particolare un'analisi dei requisiti del progetto, il disegno delle classi secondo il pattern Model-View-Controller e la codifica in Java del progetto.

### 5.1 Specifiche e requisiti

L'intento è quello di realizzare un applicazione che fornisca supporto ad un'agenzia immobiliare di piccole dimensioni. In generale l'applicazione deve permettere all'amministratore dell'agenzia di poter gestire in modo semplice gli annunci. Questo comprende il loro inserimento, modifica, cancellazione e ricerca. Gli utenti dell'applicazione invece, devono poter accedere facilmente e velocemente agli annunci e devono poter contattare l'agenzia, quindi devono poter reperire le informazioni ad essa relativa.

Trattandosi di una piccola agenzia è conveniente inserire i dati di quest'ultima direttamente in fase di installazione, in modo che una volta installata, l'applicazione sia già fornita dei dati necessari, evitando di doverli inserire successivamente. I dati in questione sono : societari (regione sociale, telefono fisso, cellulare, fax, email), indirizzo(nazione, regione, comune, numero civico, CAP), altri dati (logo, descrizione ecc.). Sempre in questa fase deve essere possibile customizzare l'interfaccia dell'ap-

plicazione, in modo da poterla vendere a più aziende che, modificandone la veste grafica possono dichiarare di avere ognuna un prodotto diverso.

Deve essere possibile modificare oltre che lo sfondo, il colore, il font ed altri elementi puramente grafici anche la posizione degli elementi sullo schermo. Così facendo, una volta completata questa fase, gran parte dei contenuti saranno inseriti e all'amministratore resterà solo da inserire gli annunci. Come detto in precedenza, deve essere anche possibile modificare, cancellare o cercare annunci e deve essere possibile modificare i dati relativi all'agenzia stessa. Per poter accedere a queste funzionalità, l'amministratore deve necessariamente autenticarsi. Un annuncio deve contenere diverse informazioni che dipendono però dalla tipologia dell'immobile. Una possibile divisione per categorie potrebbe essere : immobili residenziali e commerciali. La prima categoria comprende appartamenti, attici/mansarde/loft, ville singole o a schiera e rustici/casali mentre la seconda box/garage, terreni edificati e agricoli e altro.

Gli immobili residenziali in vendita sono caratterizzati da: prezzo, superficie(mq), numero di locali, stato( buono, da ristrutturare, discreto, nuovo/in costruzione, ottimo, ristrutturato), piano, totale piani, presenza ascensore, classe energetica, riscaldamento( assente, autonomo, centralizzato), cucina( abitabile, angolo cottura, cucinotto, semi-abitabile), box/garage( singolo, doppio, posto auto, assente), numero di bagni, giardino( privato, condiviso, assente), altro( condizionatore, terrazzo), balcone. Quelli in affitto hanno le stesse caratteristiche a differenza del prezzo che viene sostituito dal canone e dell'aggiunta di informazioni sul contratto e sull'arredamento( arredato, parzialmente, no). Gli immobili commerciali invece sono caratterizzate da : prezzo e superficie più nel caso di terreni agricoli da informazioni su proprietà e colture, nel caso di box/garage dal numero dei posti auto. Ad ogni annuncio deve essere associato un codice univoco e deve presente una breve descrizione, l'indirizzo e opzionalmente una o più foto.

Uno dei servizi che l'applicazione deve offrire è quello della ricerca di uno o più immobili. Questa è conveniente che sia effettuata mediante due step: il primo consiste nella ricerca per regione, provincia ed eventualmente città e/o tipologia per poi andare ad effettuare dei raffinamenti nel secondo passo. La prima ricerca viene effettuata tramite l'utilizzo di una mappa dell'Italia con le regioni cliccabili. Cliccandone una, la mappa viene sostituita con quella della regione corrispondente,

viene selezionata la regione dalla lista e viene sbloccata una nuova lista contenente le provincie. Cliccando su una provincia, viene selezionata una provincia e viene sbloccata una lista con le città. Questo approccio permette agli utenti che si interfacciano con l'applicazione di poter effettuare velocemente e agilmente una ricerca. Può essere utile per l'utente tenere traccia di alcuni degli annunci che ha visionato, mettendoli in una specie di bacheca virtuale. Questa dà la libertà agli utenti di visionare tutti gli annunci che si vogliono per poi visualizzare in seconda battuta solo quelli che ha ritenuto più interessanti senza doverli cercare nuovamente.

Inoltre, un utente potrebbe voler confrontare due annunci, l'applicazione deve quindi essere in grado di mettere in parallelo due immobili e mostrare le differenze tra le due soluzioni scelte. La visualizzazione degli annunci deve essere realizzata in due modi: tramite una lista oppure una mappa. Nella lista per ogni annuncio sono presenti una foto, una breve descrizione e i dati più importanti come prezzo, superficie e città. Nella seconda modalità di visualizzazione invece sono mostrati tutti gli annunci come punti sulla mappa della città o regione selezionate. In entrambe i casi cliccando su di un singolo annuncio verranno mostrati i dettagli: una galleria fotografica, una descrizione nei particolari dell'immobile e la mappa che ne mostra la posizione. In modalità mappa abbiamo la possibilità di zoomare delle aree precise; quando ciò accade se si torna alla lista, deve essere creata una divisione di quest'ultima in "presenti nell'area zoomata" e non. Per qualsiasi informazione, l'utente deve essere in grado di contattare l'agenzia, quindi deve essere presente una sezione dove sono messi in mostra il numero di telefono, di fax e l'indirizzo dell'agenzia.

## 5.2 Casi d'uso

I casi d'uso presi in considerazione sono tre, e si riferiscono alle funzionalità base che l'applicazione deve implementare sia dal punto di vista dell'amministratore che da quello dell'utente. Di seguito è riportato il diagramma dei casi d'uso che mostra in maniera estremamente sintetica ma efficace la situazione sopra descritta.

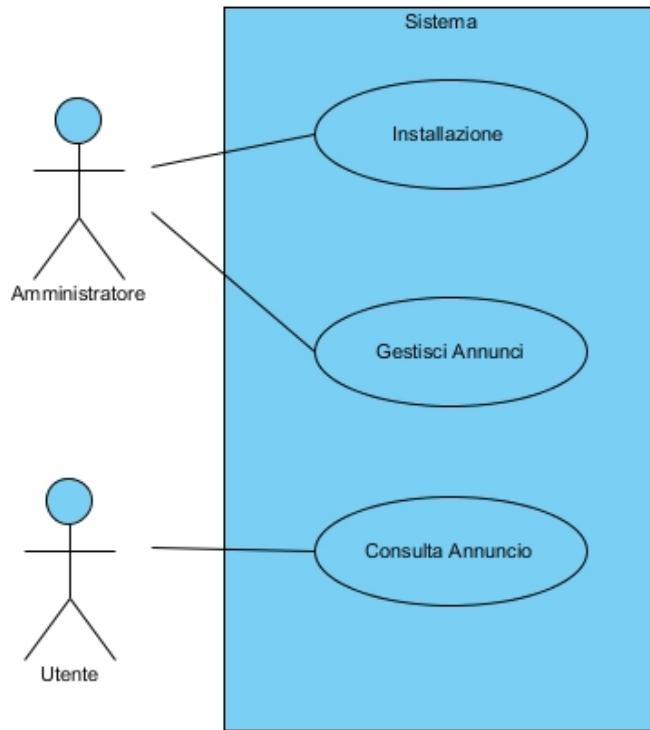


Figura 5.1: Diagramma dei casi d'uso

### 5.2.1 Caso d'uso Installazione

Questo è un caso d'uso di business che permette di installare l'applicazione oltre che tramite l'inserimento di dati, sia tecnici che riguardanti l'agenzia, anche tramite la scelta di uno stile grafico. L'attore principale coinvolto è l'amministratore. Vediamo ora in dettagli lo scenario base:

1. l'utente accede per la prima volta all'applicazione;
2. il sistema chiede il dati relativi al database e al servizio mail: nome, posizione, numero di porta, username e password del database e server, numero di porta, username e password del servizio mail;
3. l'utente inserisce i dati relativi;

4. il sistema chiede i dati relativi all'agenzia: nome, indirizzo, username e password dell'amministratore, logo, numero di telefono, numero di fax e indirizzo email;
5. l'utente inserisce i dati relativi;
6. il sistema mostra una serie di temi grafici;
7. l'utente sceglie il tema che preferisce;
8. il sistema scrive i dati ricevuti su un file di configurazione e crea il database coerentemente con i dati inseriti;

### **5.2.2 System Sequence Diagram per Installazione**

Possiamo schematizzare il caso d'uso in esame tramite un diagramma di sequenza tra l'attore principale “Amministratore” e il sistema.

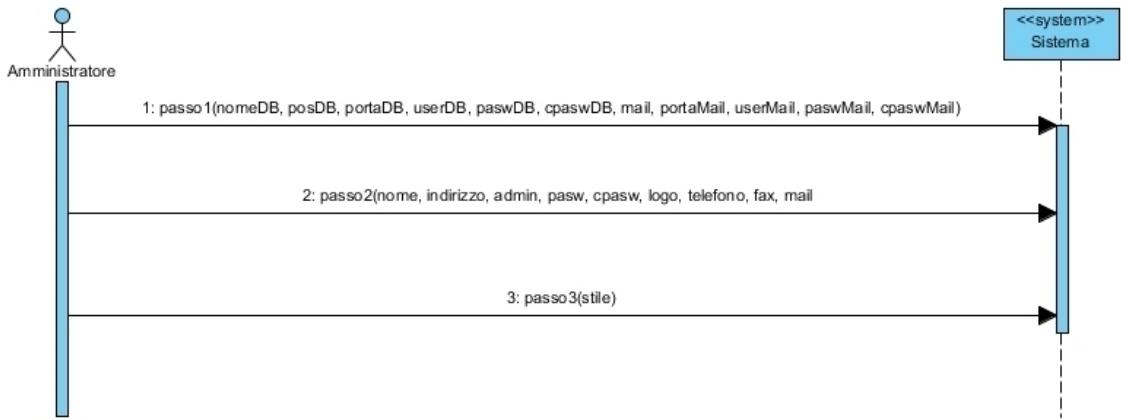


Figura 5.2: System Sequence Diagram per Installazione

### 5.2.3 Caso d'uso Consulta Annuncio

Questo è un caso d'uso di business e prevede oltre allo scenario principale anche alcune varianti da trattare a parte. Il caso d'uso Consulta Annuncio serve agli utenti per :

- cercare gli annunci;
- salvare gli annunci in bacheca;
- confrontare due annunci;

Vediamo ora in dettagli lo scenario base:

1. l'utente vuole consultare un annuncio;
2. l'utente seleziona regione, categoria, tipologia e opzionalmente provincia e città;
3. l'utente seleziona un annuncio;
4. il sistema mostra i dettagli relativi all'annuncio;

Allo scenario principale si affiancano varie alternative come elencato di seguito:

**4.a L'utente decide di raffinare la ricerca:**

1. l'utente inserisce ulteriori parametri a seconda della categoria e della tipologia dell'immobile descritto nell'annuncio;
2. il sistema mostra una lista di annunci coerentemente con i nuovi parametri di ricerca inseriti;

**4.b L'utente vuole salvare un annuncio in bacheca:**

1. l'utente seleziona l'opzione salva in bacheca;
2. il sistema inserisce l'annuncio in bacheca;

**4.c L'utente vuole confrontare l'annuncio:**

1. il sistema chiede con quale annuncio si vuole confrontare;
2. l'utente seleziona l'annuncio
3. il sistema mette in parallelo i due annunci e ne mostra le differenze;

### **5.2.4 System Sequence Diagram per Consulta Annuncio**

Possiamo schematizzare il caso d'uso in esame tramite tre diagramma di sequenza che coinvolgono l'attore principale “utente” e il sistema.

### **5.2.5 Caso d'uso Gestisci Annuncio**

Anche questo è un caso d'uso di business che permette di inserire, cancellare e modificare gli annunci. L'attore principale coinvolto è l'Amministratore che per poter assolvere i suoi compiti dev'essere necessariamente autenticato tramite l'inserimento di username e password. Vediamo ora in dettagli lo scenario base:

1. l'Amministratore vuole gestire un annuncio;

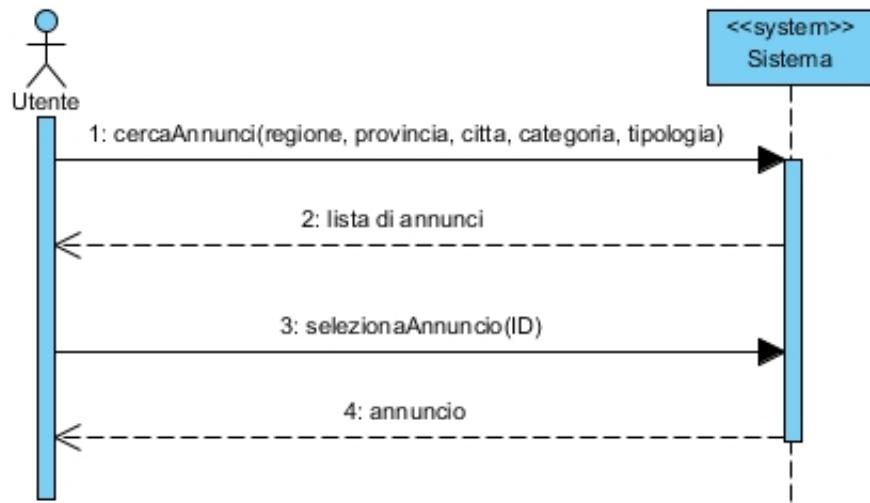


Figura 5.3: System Sequence Diagram per Consulta Annuncio: caso principale

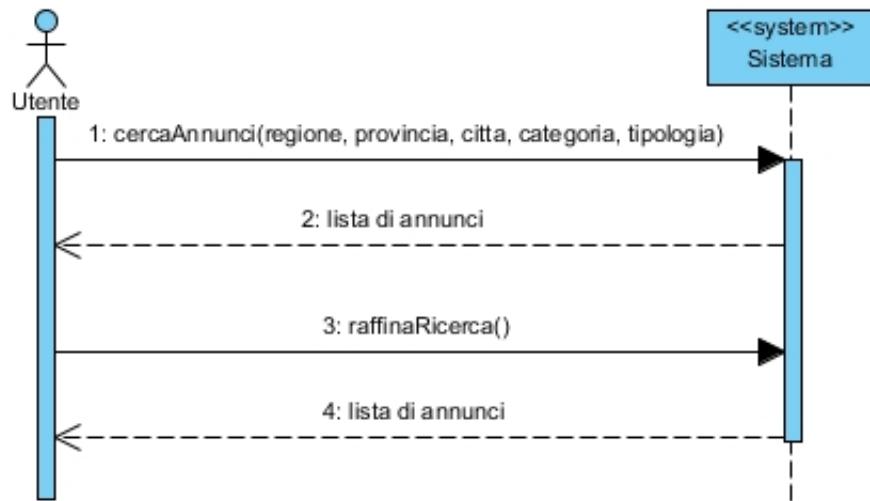


Figura 5.4: System Sequence Diagram per Consulta Annuncio: estensione

2. l'Amministratore vuole inserire un annuncio;
3. il sistema chiede i dati relativi al tipo d'immobile selezionato;
4. l'utente inserisce i dati;
5. il sistema salva l'annuncio;

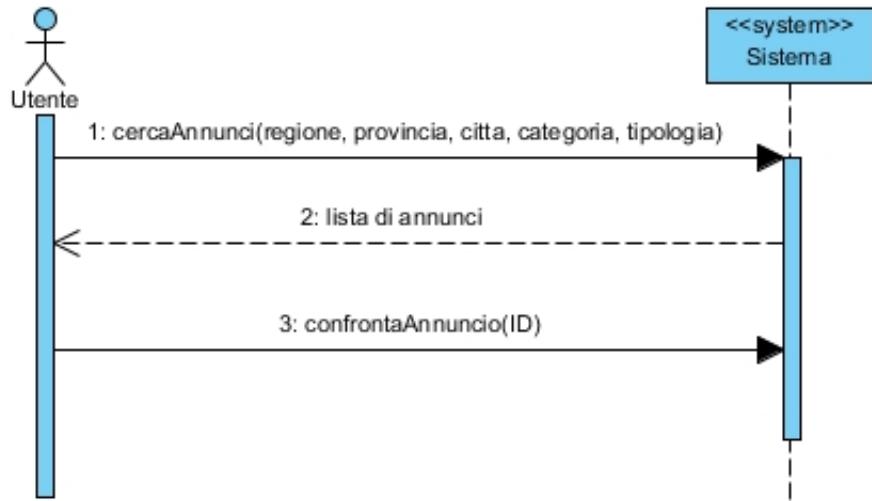


Figura 5.5: System Sequence Diagram per Consulta Annuncio: estensione

Allo scenario principale si affiancano varie alternative come elencato di seguito:

#### **2.a L'utente vuole cancellare un annuncio:**

1. l'Amministratore seleziona l'ID dell'annuncio da cancellare;
2. il sistema cancella l'annuncio;

#### **2.b L'utente vuole modificare un annuncio:**

1. l'Amministratore seleziona l'ID dell'annuncio da modificare;
2. il sistema chiede i dati relativi al tipo di immobile selezionato;
3. il sistema modifica i dati dell'annuncio;

#### **5.2.6 System Sequence Diagram per Gestisci Annuncio**

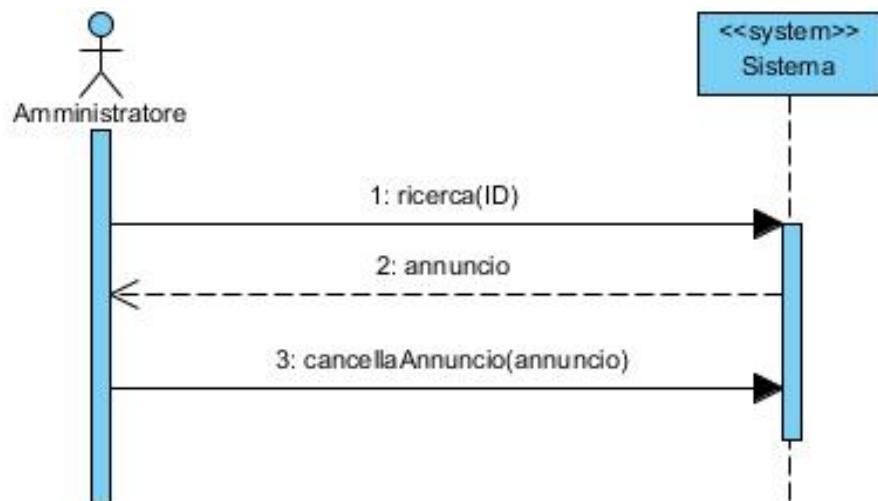


Figura 5.6: System Sequence Diagram per Gestisci Annuncio: caso principale

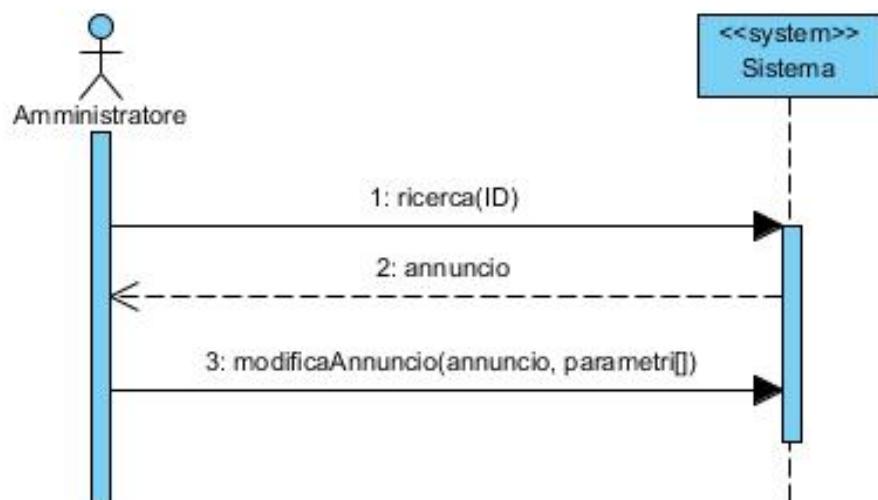


Figura 5.7: System Sequence Diagram per Gestisci Annuncio: estensione

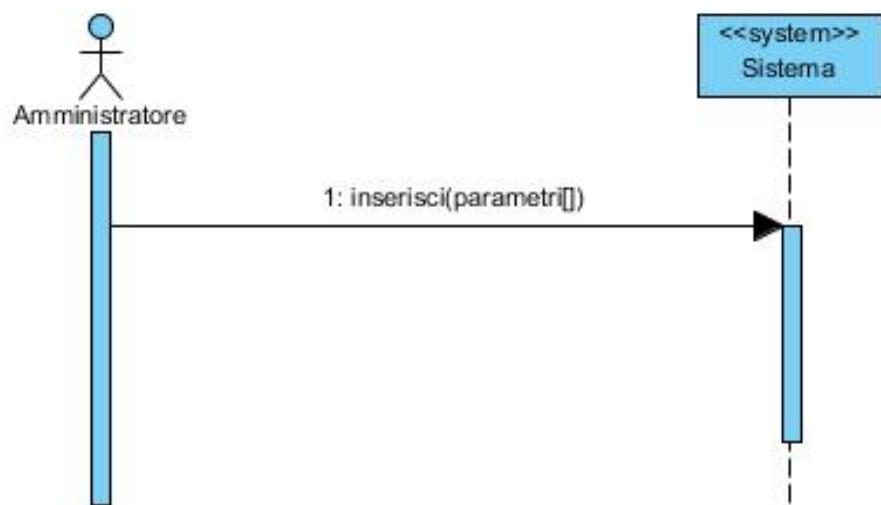


Figura 5.8: System Sequence Diagram per Gestisci Annuncio: estensione

Capitolo **6**

## Applicazione : Progettazione

In questo capitolo si introduce l'utilizzo di UML per descrivere mediante diagrammi il progetto in questione, attraverso i passi che caratterizzano lo sviluppo del software: in particolare analizzeremo il modello di dominio, il diagramma delle classi, ed alcuni pattern utilizzati nel diagramma delle classi per avvicinarsi alla fase di implementazione attraverso piccoli salti rappresentazionali.

### 6.1 Modello di dominio

Con il modello di dominio si ha una rappresentazione visuale di oggetti nel mondo reale o classi concettuali del dominio in analisi, e non classi software. Relativamente al progetto in esame possiamo ricavare un primo modello di dominio come si vede in figura 6.1 per rappresentare lo scenario base già precedentemente preso in considerazione.

- un utente può cercare più annunci;
- un utente può confrontare due annunci;
- un'agenzia può gestire più annunci;
- un'agenzia è descritta da logo, descrizione, numero di telefono, numero di fax, indirizzo, nome, indirizzo email, username e password;

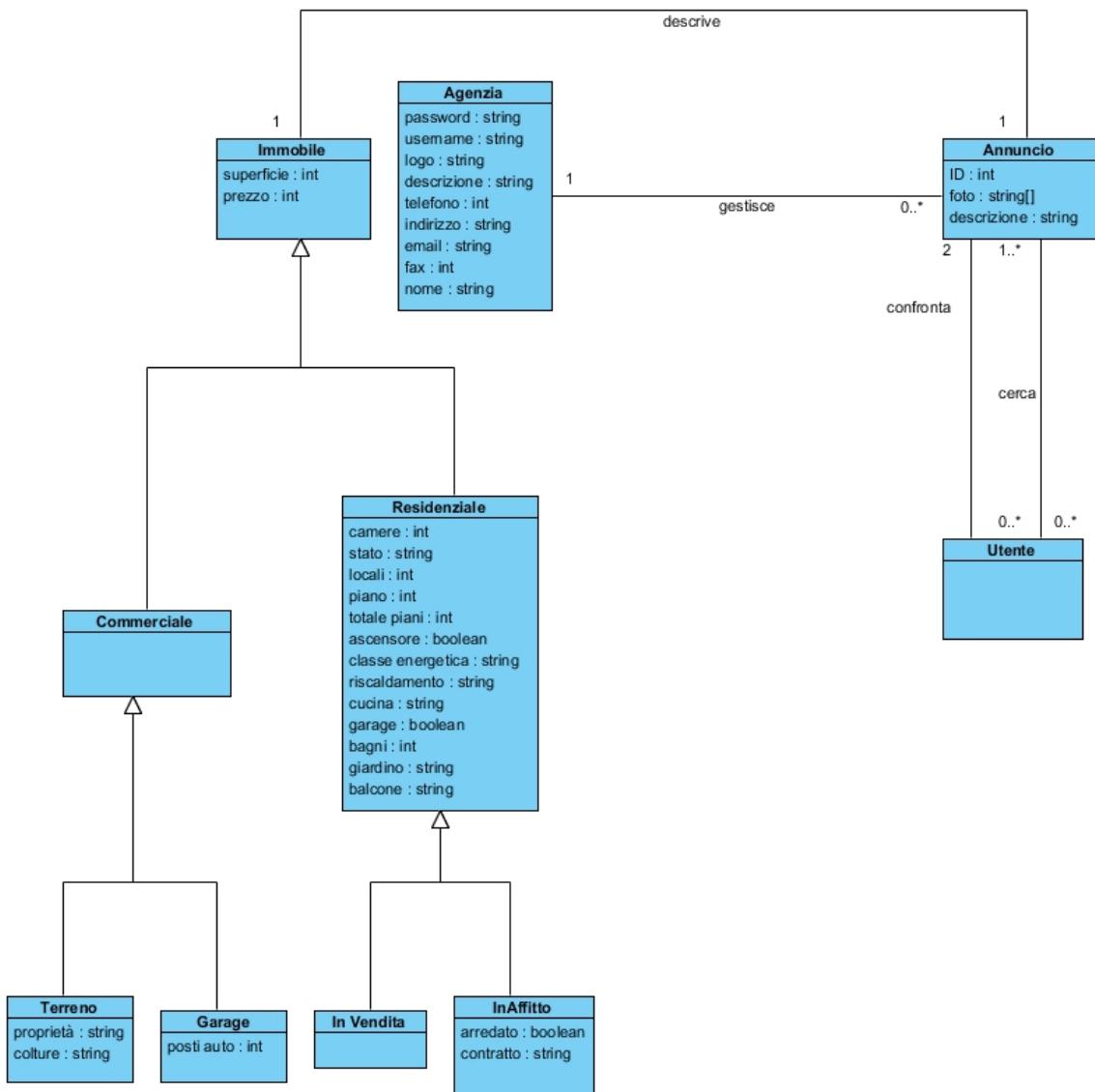


Figura 6.1: Modello di dominio

- un annuncio descrive uno ed un solo immobile;
- un annuncio è descritto da un id univoco, una o più foto e una descrizione;
- un immobile è descritto da superficie e prezzo ed ha uno ed un solo indirizzo;

- in un indirizzo ci possono essere più immobili, come ad esempio nel caso degli appartamenti in una palazzina;
- un immobile può essere di tipo commerciale o residenziale;
- un immobile residenziale è descritto dal numero di locali, dallo stato( buono, da ristrutturare, discreto, nuovo/in costruzione, ottimo, ristrutturato), dal piano, dal totale dei piani, della presenza o meno dell'ascensore, dalla classe energetica, dal riscaldamento( assente, autonomo, centralizzato),dalla cucina( abitabile, angolo cottura, cucinotto, semi-abitabile),dal box/garage( singolo, doppio, posto auto, assente), dal numero di bagni, dal giardino( privato, condiviso, assente) e altro( condizionatore, terrazzo), balcone;
- un immobile residenziale può essere in vendita o in affitto;
- un immobile residenziale in vendita ha le stesse caratteristiche dell'immobile residenziale;
- un immobile residenziale in affitto ha le stesse caratteristiche dell'immobile residenziale più informazioni sul contratto e sul fatto che l'immobile sia arredato a meno;
- un immobile commerciale può essere o un garage o un terreno;
- un garage è descritto dal numero di posti\_auto;
- un terreno è descritto dal tipo di colture e dalle sue proprietà;

## 6.2 Diagramma delle classi

### 6.2.1 Architettura logica

Dai modelli di analisi concettuale si passa ai diagrammi delle classi software. Prima ancora di analizzarli però, vediamo come sono organizzate le classi e i package. L'architettura utilizzata è quella a due livelli Client Server, con un client pesante, ovvero capace di elaborare i dati e presentarli graficamente all'utente. Il server sarà molto leggero ed avrà il solo compito di gestire il problema della persistenza dei

dati. Il pattern architettonale più utilizzato per applicazioni web è l'MVC (Model-View-Controller), implementato in questa applicazione come si vede nella figura. Nel diagramma mostrato nella figura 6.2 il package Model è stato chiamato Entity, due nomi che in letteratura sono assolutamente equivalenti.

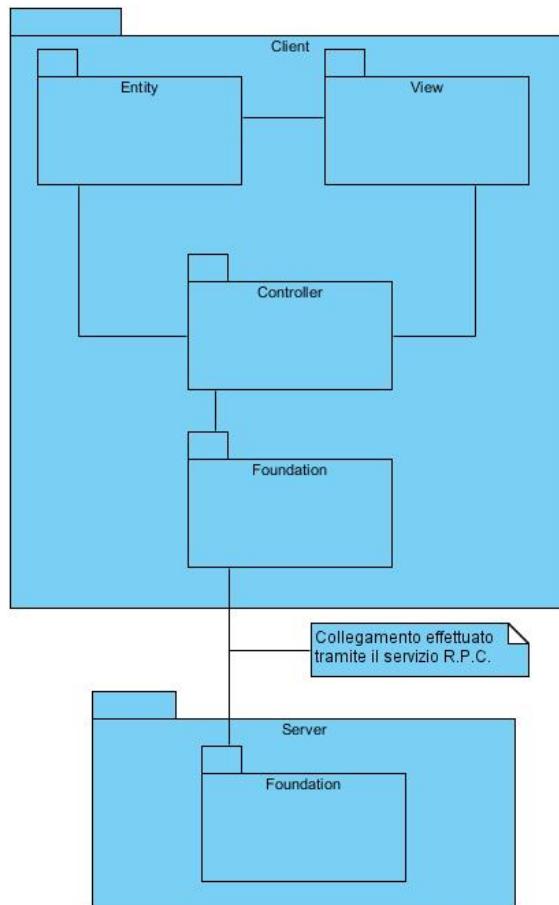


Figura 6.2: Architettura logica, pattern MVC

Analizzeremo nelle sezioni successive i package come sono strutturati al loro interno, per poi approfondire nella sezione “Design Pattern” le interazioni fra essi.

### 6.2.2 Entity package

Il diagramma delle classi in figura rappresenta il package Entity. Tutte le classi al suo interno iniziano con il prefisso “E” e la stessa cosa sarà riproposta poi nei

package successivi ovvero per il package Controller avremo il prefisso “C” e per il View “V”. Inoltre, coerentemente con ciò che è stato detto nel capitolo 3, riguardo il servizio RPC, tutte le classi sono Serializzabile, quindi in grado di trasportare dati dal client al server. Possiamo notare come nel modello di dominio sono rimasti quasi tutti i concetti che erano presenti: questo in virtù del fatto di aver compiuto un salto rappresentazionale basso.

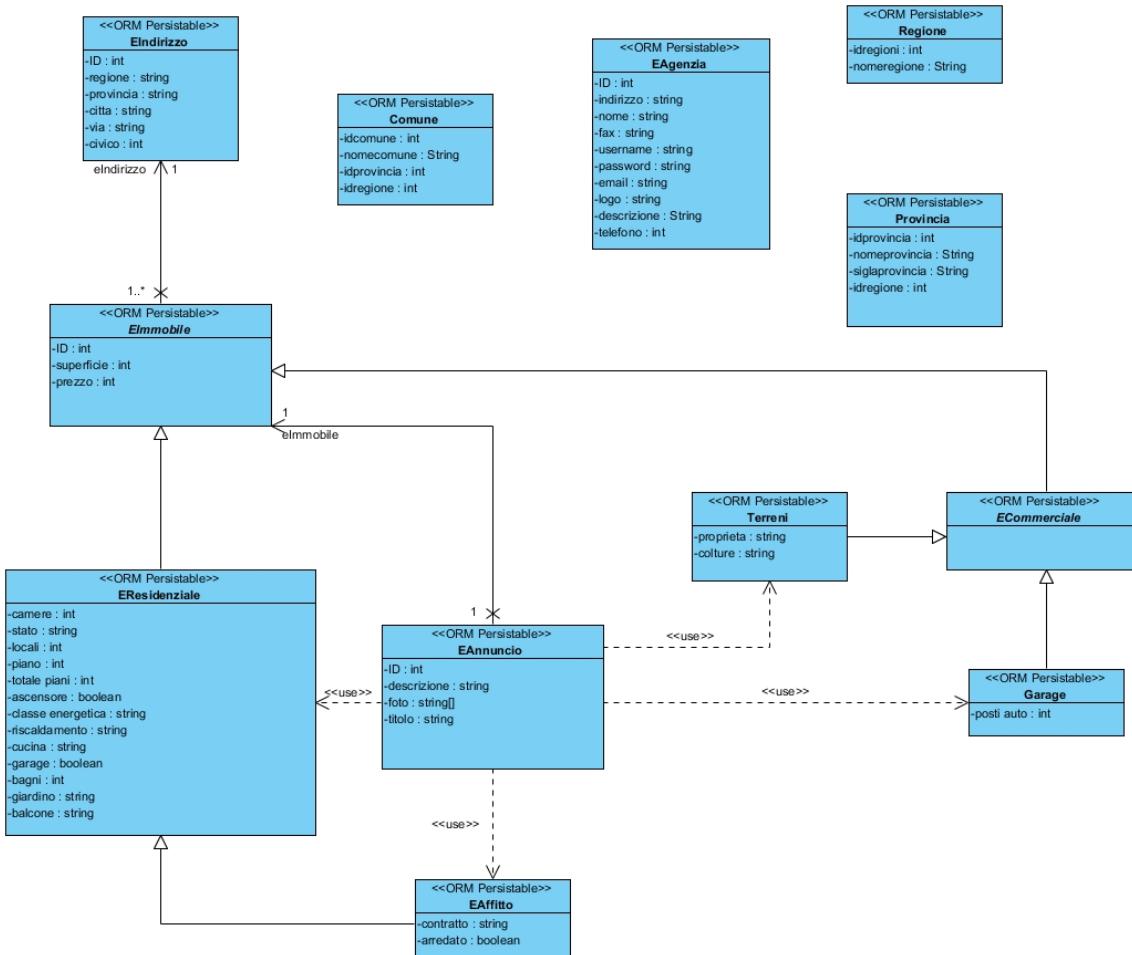


Figura 6.3: Diagramma delle classi per il package Client.Entity

Vediamo in particolare le classi e le loro peculiarità:

- **EAnnuncio:** una delle classi più importanti del diagramma in quanto tutta l'applicazione si basa sulla visione e sulla gestione degli annunci. Tra i suoi

attributi troviamo un id univoco che la identifica oltre che un titolo, un array di stringhe che contiene i path relativi delle foto e una descrizione. Inoltre è la classe responsabile di mantenere in sé i dati relativi all'immobile descritto nell'annuncio mediante l'attributo “eImmobile”. Da notare che l'attributo è di tipo EImmobile, classe posta in cima alla gerarchia che descrive gli immobili, in modo da poter utilizzare al suo interno una qualsiasi delle classi da essa derivata;

- **EImmobile:** classe che raccoglie gli attributi comuni a tutti i tipi di immobili ovvero superficie e prezzo. Anche in questo caso alla classe è stato assegnato un identificatore univoco in modo da rendere un immobile immediatamente identificabile. È la classe responsabile di mantenere in sé i dati relativi all'indirizzo in cui è posizionata tramite l'attributo “eIndirizzo”;
- **EResidenziale:** classe derivata da EImmobile che la specializza inserendo gli attributi relativi ad un immobile di tipo residenziale quale può essere un appartamento, una casa singola o una villetta;
- **EAffitto:** classe che a sua volta deriva da EResidenziale e che la specializza in quanto EAffitto ha in più due attributi, che sono nello specifico contratto e arredato;
- **ECommerciale:** classe derivata anch'essa da EImmobile, come nel caso di EResidenziale. Da notare che non ha nessun attributo, ma è stata creata lo stesso sia per mantenere traccia della logica che prevede la suddivisione di un immobile in commercial e residenziale sia per evitare un collegamento diretto tra le classi che deriveranno da ECommerciale con EImmobile;
- **EGarage:** classe derivata da ECommerciale e che la specializza tramite l'aggiunta dell'attributo posti\_auto;
- **ETerreni:** come EGaraage con la sola differenza dell'aggiunta degli attributi proprietà e colture invece che dell'attributo posti\_auto;
- **EIndirizzo:** classe che rappresenta la posizione geografica di un immobile infatti conoscendone, regione, provincia, città, via e numero civico si riesce

ad assegnarli una posizione univoca. Da notare la presenza dell'attributo ID, apparentemente superfluo, che è stato inserito per questioni derivanti l'utilizzo di hibernate per la creazione e gestione del database;

- **ERegione:** classe che rappresenta una regione italiana tramite il nome ed un id;
- **EProvincia:** classe che rappresenta una provincia italiana tramite il nome un id e l'id della regione a cui appartiene;
- **ECittà:** classe che rappresenta una citta italiana tramite il nome un id e gli id della provincia e della regione a cui appartiene;
- **EAgenzia:** classe che rappresenta l'agenzia immobiliare con tutti i suoi attributi compresi username e password, necessari per avere l'accesso alla parte di back-end dell'applicazione;

### 6.2.3 Controller Package

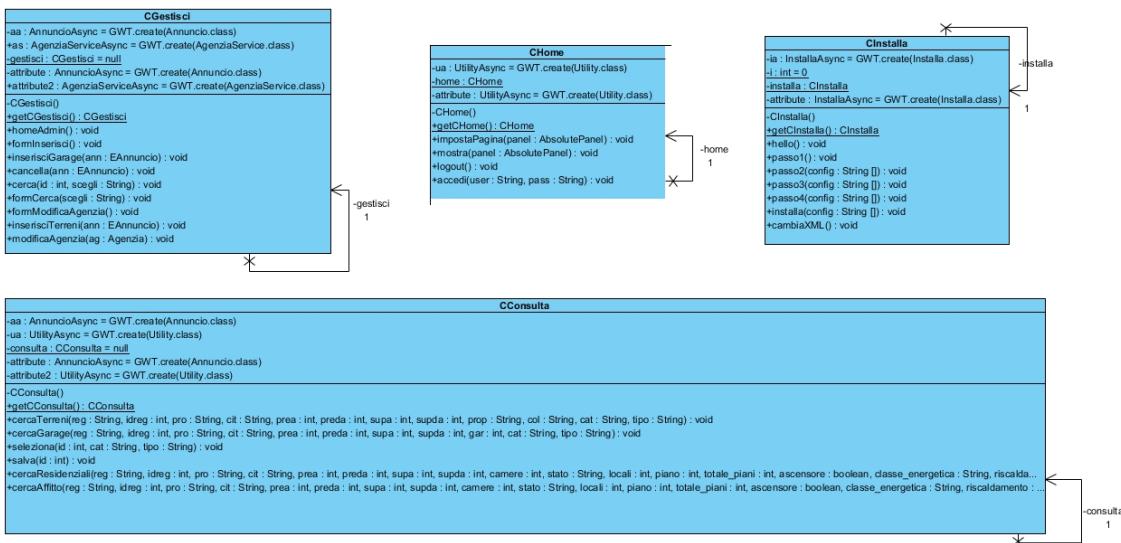


Figura 6.4: Diagramma delle classi per il package `Client.Controller`

La logica seguita per la realizzazione del package Controller è stata quella di creare delle classi, ognuna candidata alla gestione di un caso d'uso. Questo traspare

infatti anche dai loro nomi che sono formati dal prefisso C e il nome del caso d'uso che ognuna controlla. La maggior parte delle classi che compongono il package hanno la funzione di gestire una View poichè come scopo principale hanno esattamente quello della gestione degli eventi che scaturiscono dall'interazione dell'utente con l'interfaccia grafica. Tutte le classi del package sono state create secondo il pattern Singleton, che vedremo nel capitolo sui Design Pattern, in quanto di ogni controller è necessaria una sola istanza. L'unica classe che non controlla alcun caso d'uso è CHome alla quale però sono assegnati altri compiti non di minor importanza. Il diagramma mostra appunto le classi del package e come fatto nella sezione precedente, ne vediamo i dettagli:

- **CHome**: il controller di riferimento di tutto il progetto è proprio questa classe poichè ha il ruolo di gestire di volta in volta il contenuto della pagina, tramite il metodo impostaPagina. Questo ogni volta che invocato, oltre che a delegare la stampa della pagina alla View associata, in questo caso VHome, controlla anche che i cookie siano sempre attivati altrimenti lo segnala ed impedisce di continuare l'utilizzo dell'applicazione;
- **CConsulta**: controller relativo al caso d'uso Consulta Annuncio. La classe ha sostanzialmente tre compiti:
  - catturare gli eventi che derivano dalle azioni di ricerca o di raffinamento della stessa e richiede alla classi di foundation di recuperare i dati che poi passerà alla view associata, VConsulta;
  - catturare gli eventi che derivano dai click su un determinato annuncio in modo da andare a recuperare su database l'oggetto corrispondente e mandarlo a VConsulta;
  - catturare gli eventi che derivano dall'inserimento di un annuncio in bacheca. In questo caso il controller provvederà a salvare l'annuncio chiamando una funzione sul server delegata al suo salvataggio nella sessione;
- **CGestisci**: controller relativo al caso d'uso Gestisci Annuncio. La classe ha il compito di gestire tutti gli eventi relativi alla parte di back-end dell'applicazione;

- **CInstalla:** controller relativo al caso d'uso Installazione. La classe ha il compito di gestire tutti gli eventi generati durante la fase di installazione dell'applicazione;

### 6.2.4 View Package

La logica che ha portato alla realizzazione del package View è la stessa della precedente ovvero è stata realizzata una classe per ogni caso d'uso. Ognuna di esse inoltre è collegata direttamente al controller del caso d'uso, poichè come già accennato in precedenza, la classi view hanno il compito di catturare gli eventi derivanti dall'interazione tra gli utenti e l'interfaccia grafica e smistare la richiesta al controller relativo. Non solo, queste classi hanno anche il compito di costruire dinamicamente l'interfaccia a seconda degli input derivanti sempre dai controller associati. Anche in questo caso e per le stesse ragioni, tutte le classi del package sono state implementate come singleton. Il diagramma in figura mostra le classi del package in dettaglio:

- **VHome:** classe responsabile del layout della home page dell'applicazione, nonchè classe delegata alla stampa di tutte le pagine dell'applicazione;
- **VInstalla:** classe responsabile del layout della parte dell'applicazione che concerne l'installazione. Genera una pagina per ogni step previsto in questa fase;
- **VGestisci:** classe responsabile del layout della parte di back-end dell'applicazione;
- **VConsulta:** classe responsabile del layout per quanto riguarda la lista di risultati della ricerca e la visualizzazione dei dettagli di un singolo annuncio. E' l'unica view che fa uso delle mappe per questo è in relazione con la classe VMapsAdapter;
- **IVMapsAdapter:** interfaccia al vertice della gerarchia delle classi responsabili della creazione della mappa;
- **VMapsAdapter:** classe concreta responsabile dalla creazione della mappa, utilizzando la API di GoogleMaps;

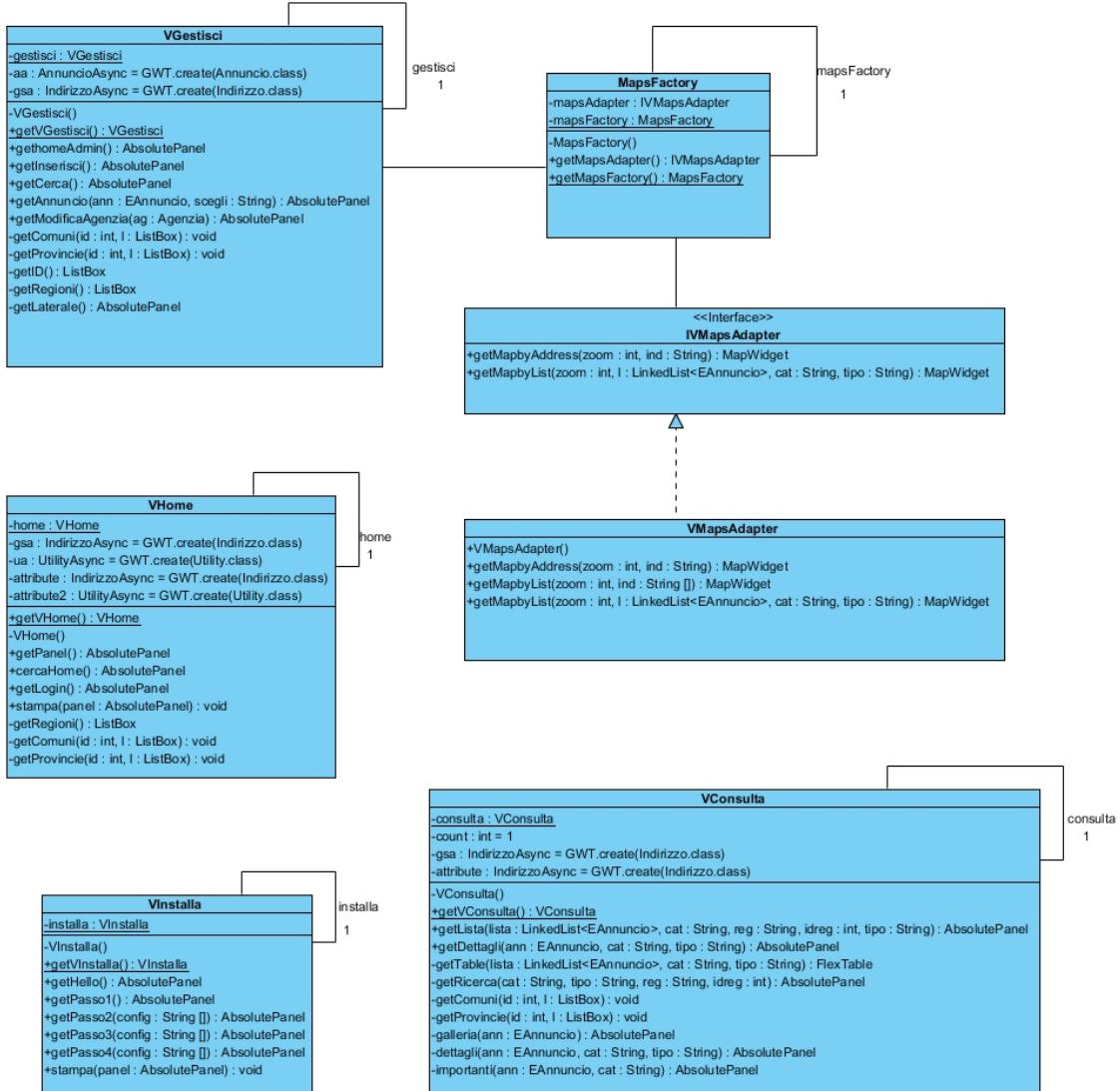


Figura 6.5: Diagramma delle classi per il package Client.View

- **MapsFactory**: classe responsabile della creazione di un oggetto di tipo `IVMapsAdapter`;

### 6.2.5 Foundation package: Client

Il package contiene al suo interno le interfacce che serviranno poi alla realizzazione del servizio RPC. Da notare come per ogni classe, coerentemente con quello che è

stato detto nella sezione riguardante l'RPC, sia presente la controparte asincrona, facilmente identificabile tramite il suffisso "Async". Di seguito è solo riportato il diagramma poichè lasceremo i dettagli riguardanti le singole classi nella prossima sezione, dove verrà presentato il package relativo all'implementazione del servizio.

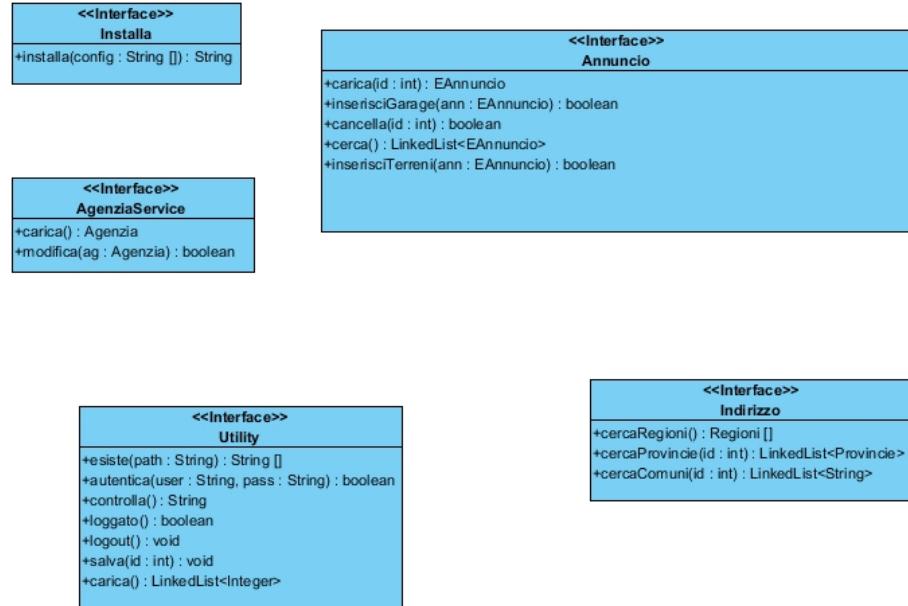


Figura 6.6: Diagramma delle classi per il package Client.Foundation

### 6.2.6 Foundation Package: Server

Il package Foundation lato server ospita le classi che implementano l'interfaccia lato client per la realizzazione del servizio RPC. Ogni classe è da considerarsi come un servlet Java delegata al recupero e alla gestione dei dati sul database. Da notare come sul server siano presenti solo cinque classi, in linea con la scelta fatta inizialmente di avere un client pesante e un server estremamente leggero. Andremo ora ad analizzare il package in figura 6.8:

- AnnuncioImpl: classe che ha il compito di caricare, cercare, eliminare ed inserire annunci. È la classe del package foundation che svolge più lavoro in quanto tutta l'applicazione è incentrata proprio sulla gestione degli annunci;

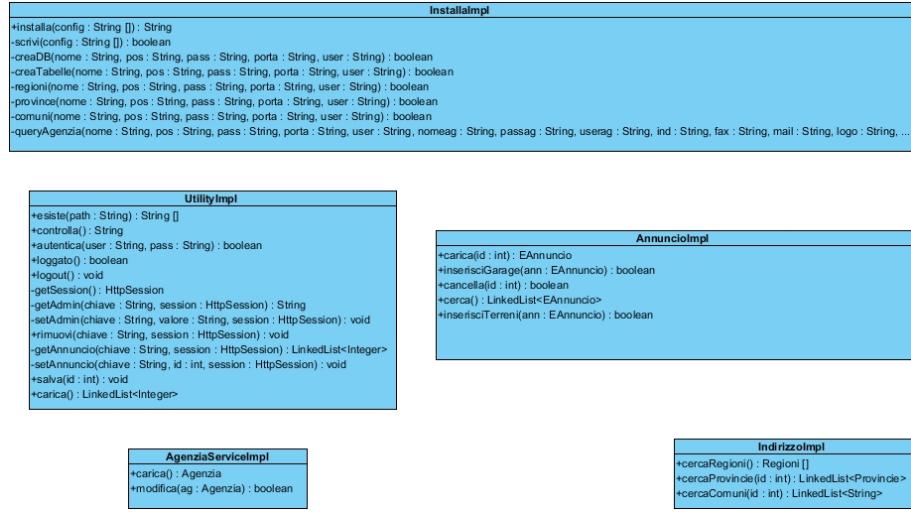


Figura 6.7: Diagramma delle classi per il package Client.Foundation

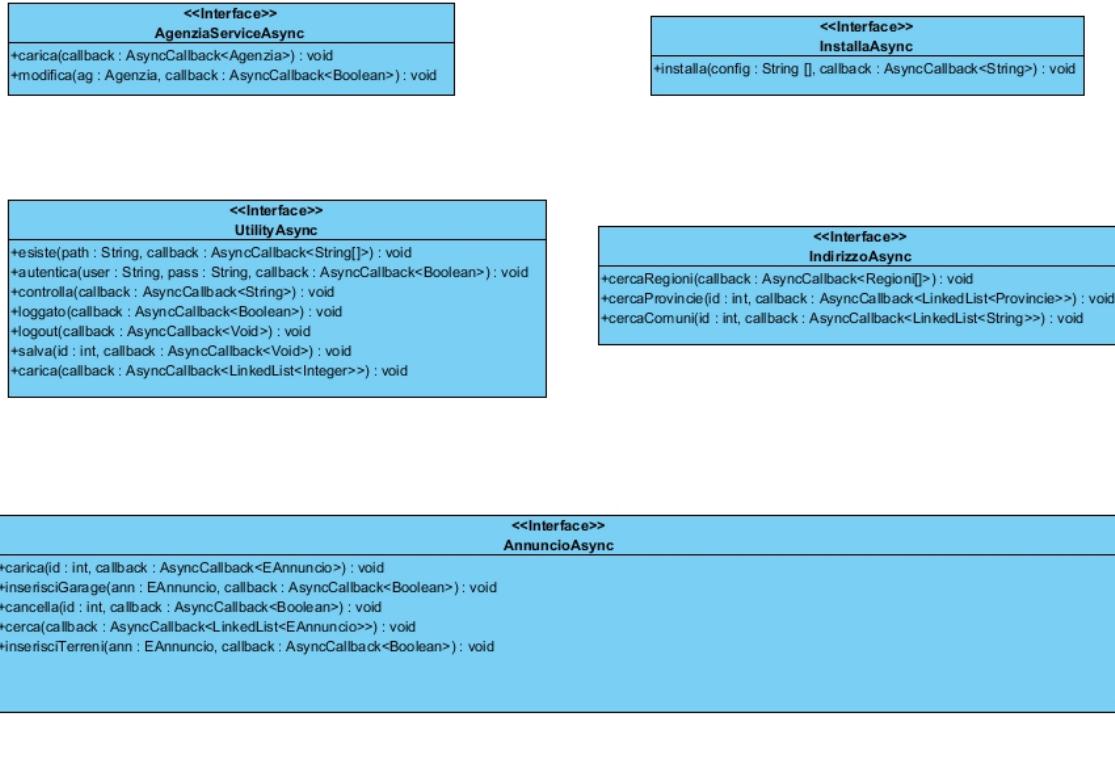


Figura 6.8: Diagramma delle classi per il package Server.Foundation

- IndirizzoImpl: classe che svolge tre compiti molto simili tra loro:
  - trovare tutte le regioni;
  - trovare tutte le provincie di una regione, dato l'id di una regione;
  - trovare tutte le città di una provincia, dato l'id di una provincia;
- AgenziaImpl : classe che il compito di gestire (inserire, modificare, caricare) i dati relativi all'agenzia;
- UtilityImpl : classe che svolge alcuni funzioni di utilità per il sistema come ad esempio il controllo dell'esistenza di un file, gestione login e logout, e controllo della sessione.
- InstallaImpl: fondamentalmente svolge l'unica funzione di installare l'applicazione ma questa operazione è divisa per una mera questione di praticità e leggibilità del codice in più parti:
  - scrittura dei dati sul file di configurazione;
  - creazione del database;
  - popolazione del database;

### 6.2.7 Database design

Lo sviluppo di un'applicazione client-server con tool per l'ORM permette di semplificare molto lo sviluppo software da parte del programmatore. Proprio per questo, nel costruire l'applicazione in esame ci si è avvalsi di Visual Paradigm, un case che tra le sue molteplici funzionalità presenta anche quella descritta qualche riga più su. Grazie a questo strumento partendo dal diagramma che rappresenta il package Entity è possibile con un click generare e successivamente, sincronizzare a fronte di eventuali cambiamenti il modello entità relazioni. Di seguito riportato il diagramma (figura 6.9), abbastanza chiaro da non necessitare di ulteriori commenti:

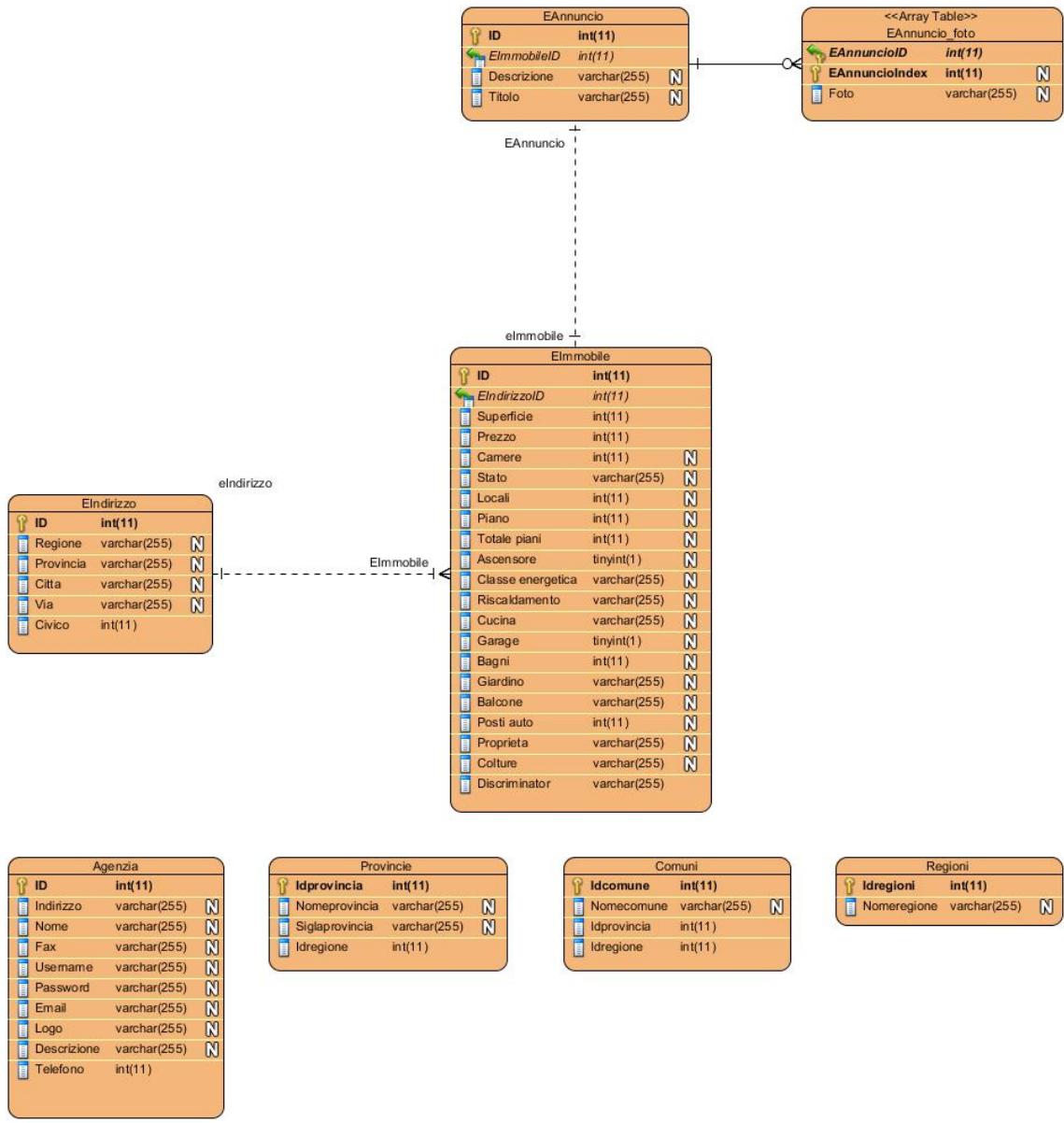


Figura 6.9: Diagramma Entità relazioni

## 6.3 Design Pattern

Nell'ambito dell'ingegneria del software, un design pattern può essere definito “una soluzione progettuale generale a un problema ricorrente”. Esso non è una libreria o un componente di software riusabile, quanto piuttosto una descrizione o un modello da applicare per risolvere un problema che può presentarsi in diverse situazioni durante la progettazione e lo sviluppo del software. Al fine di rendere l'applicazione più gestibile e robusta è risultato necessario applicare ed implementare alcuni design pattern noti in letteratura.

### 6.3.1 Singleton

Tutte le classi dei package View e Controller sono state implementate secondo il pattern singleton, permettendo quindi, a tutte le classi del progetto di accedervi mediante il metodo di classe “getNome della Classe()”. Questi metodi alla prima chiamata instanziano un oggetto che sarà memorizzato come attributo statico della classe, in modo tale che le chiamate seguenti al metodo restituiscano l'unico oggetto della classe che è stato instanziato. In questo modo si garantisce l'accesso univoco sempre allo stesso oggetto.

```
package com.agenzia.immobiliare.client.controller;

public class CHome {

    private CHome() {
    }

    private static CHome home;

    public static CHome getCHome() {
        if (home == null) {
            home = new CHome();
        }
        return home;
    }

}
```

Figura 6.10: Esempio di classe che implemtenta il pattern Singleton

Nell'esempio proposto si può accedere all'unica istanza della classe tramite il metodo `getCHome()`: se è la prima volta che viene chiamato, il metodo crea un'istanza e assegna tale valore all'attributo di classe “home” che sarà poi ritornato. Le volte successive alla prima il metodo si limiterà a ritornare “home”. Da notare come il costruttore viene dichiarato privato, in modo da non dare la possibilità all'esterno della classe di creare altre sue istanze.

### 6.3.2 Adapter

L'Adapter è un pattern strutturale che può essere basato sia su classi che su oggetti. A volte viene chiamato anche wrapper (ovvero involucro) per il suo schema di funzionamento. Il suo fine è quello di fornire una soluzione astratta al problema dell'interoperabilità tra interfacce differenti. Il problema si presenta ogni qual volta nel progetto di un software si debbano utilizzare sistemi di supporto ,come per esempio librerie, la cui interfaccia non è perfettamente compatibile con quanto richiesto da applicazioni già esistenti. Invece di dover riscrivere parte del sistema, compito oneroso e non sempre possibile se non si ha a disposizione il codice sorgente, può essere comodo scrivere un adapter che faccia da tramite. L'uso del pattern Adapter risulta utile quando interfacce di classi differenti devono comunque poter comunicare tra loro. Alcuni casi possono includere:

- l'utilizzo di una classe esistente che presenta un'interfaccia diversa da quella desiderata;
- la scrittura di una determinata classe senza poter conoscere a priori le altre classi con cui dovrà operare, in particolare senza poter conoscere quale specifica interfaccia sia necessario che la classe debba presentare alle altre;

Un altro contesto è quello in cui si desidera che l'invocazione di un metodo di un oggetto da parte dei client avvenga solo in maniera indiretta: il metodo “target” viene incapsulato all'interno dell'oggetto, mentre uno o più metodi “pubblici” fanno da tramite con l'esterno. Questo consente alla classe di subire modifiche future mantenendo la retro compatibilità, oppure di implementare in un unico punto una funzionalità alla quale i client accedono tramite metodi più “comodi” da usare e

con signatures differenti. Nell'applicazione che si sta costruendo il pattern è utilizzato per interfacciare il sistema con le librerie di GoogleMaps. In questo modo un eventuale cambiamento della libreria o l'uso di una libreria differente comporterà solamente il dover cambiare il codice della classe concreta VMapsAdapter, senza dover mettere le mani su tutti i frammenti di codice che coinvolgono le librerie sopra citate. Nel diagramma in figura abbiamo un esempio concreto dell'utilizzo del pattern in quanto viene creata un interfaccia, IVMapsAdapter, che viene poi specializzata dalla classe VMapsAdapter. Questa utilizza effettivamente le librerie API di GoogleMaps, implementando i metodi dichiarati nell'interfaccia. Tutte le classi che avranno bisogno di utilizzare una mappa non dovranno interfacciarsi direttamente con le librerie ma chiameranno un metodo della classe VMapsAdapter.

### 6.3.3 Simple Factory

La logica di fondo dietro al pattern Simple Factory è quella di implementare una classe che crea di volta in volta diverse istanze di altre classi appartenenti ad una gerarchia, a seconda dei dati che le vengono passati. Di solito tutte le classi che la Factory restituisce hanno come genitore una classe comune: questo comporta che tutti i suoi figli avranno gli stessi metodi, che poi però eseguiranno compiti diversi a seconda di come verranno implementati. Detto questo sembra naturale combinare l'uso di questo pattern con quello visto nel capitolo precedente, l'Adapter: le classi IVMapsAdapter e VMapsAdapter costituiranno la gerarchia di classi e la MapsFactory avrà quindi il compito di creare un'istanza di una classe scelta tra i figli di IVMapsAdapter, che però in questo caso si limitano alla sola classe VMapsAdapter. Da notare che la classe MapsFactory è stata implementata come Singleton, altrimenti sarebbe sorto il problema di trovare una classe alla quale assegnare il compito di creare la factory.

Dal diagramma in figura 6.11 è chiaro il principio fondamentale del pattern Simple Factory. Si crea un astrazione che decide quale delle diverse classi possibile tornare, e ne restituisce una. Allora si chiamano i metodi di tale istanza di classe, senza mai sapere quali sottoclassi in realtà si sta usando.

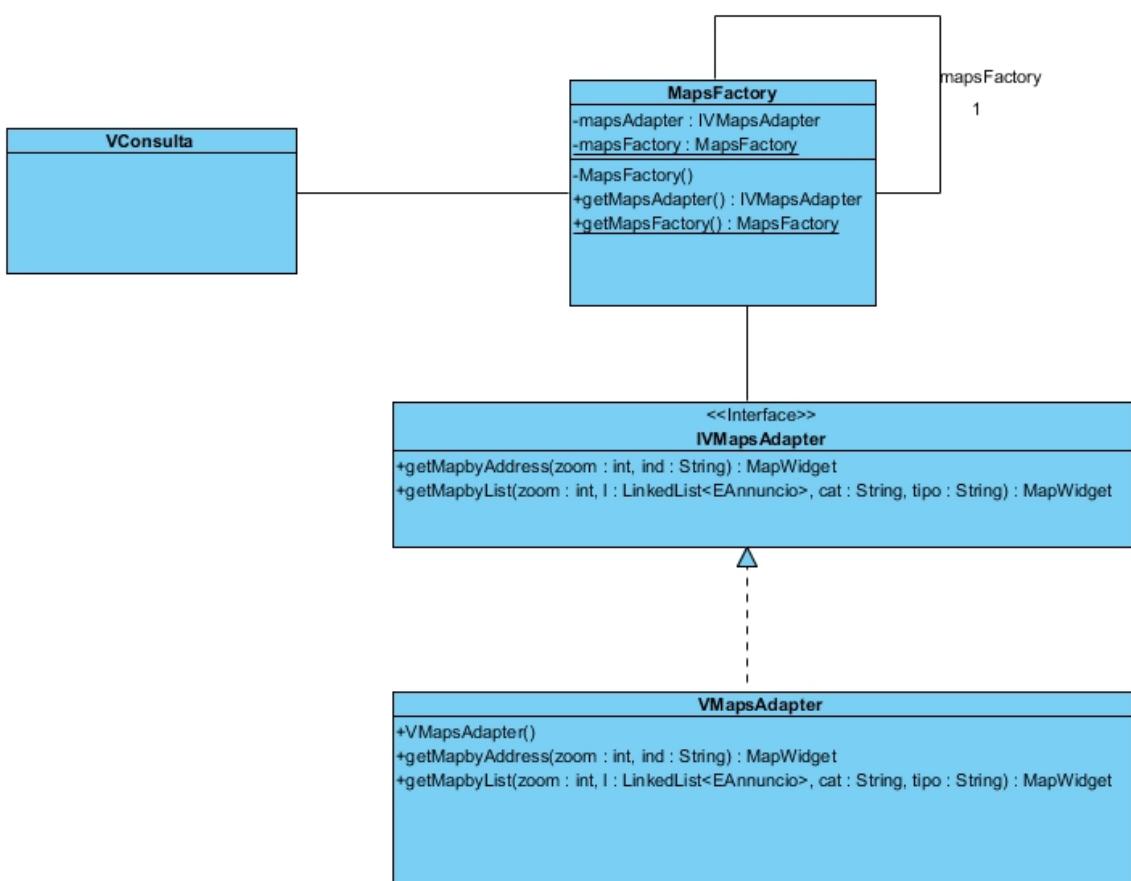


Figura 6.11: Diagramma delle classi relativamente il pattern Adapter

# Capitolo 7

## Applicazione : Implementazione

In questi capitolo analizzeremo alcune soluzioni implementative, appartenenti a vari package dell'applicazione, selezionati per evidenziare alcune peculiarità del linguaggio. Non verrà fatta un'analisi del codice implementativo puntuale, perchè non è questo l'obiettivo del capitolo e risulterebbe pesante e poco interessante. Saranno analizzate piuttosto le interazioni fra le classi e alcune sequenze di input dell'utente e verranno mostrati dei frammenti di codice riguardo le API utilizzate: servlets con le RPC e GoogleMaps.

### 7.1 Comunicazione con il server

Oggiorno è raro trovare un'applicazione che non abbia bisogno di immagazzinare dati persistenti in qualche modo, su file nei casi più semplici fino ad arrivare ai più sofisticati Database Management Systems (DBMS) e l'applicazione che stiamo esplorando non fa sicuramente eccezione . Quando si scrivono applicazioni con database ci si trova di fronte a due possibili scenari: uno in cui il database esiste già e uno in cui va progettato da zero. Nel primo caso si può procedere con un reverse engineering per identificare un modello dei dati a partire dalle tabelle esistenti. Nel secondo caso l'approccio tradizionale consiste nell'identificare entità e relazioni coinvolte e creare le tabelle sul database. In entrambi i casi si parte dalla definizione dei dati e vi si costruisce sopra l'applicazione. Questo approccio, indubbiamente immediato quando si usano strumenti di sviluppo RAD (Rapid Application Development), di-

viene immediatamente svantaggioso con metodologie di sviluppo più sofisticate, in quanto vincola le entità del modello dati (nell'applicazione di studio in linguaggio Java) alla struttura relazionale tipica dei tradizionali database. I due modelli, quello relazionale e quello ad oggetti, sono fortemente diversi per cui è necessario escogitare una forma di “traduzione” da un modello all’altro, detta Object/Relational Mapping (ORM). Questa traduzione è sempre necessaria ed introduce uno strato aggiuntivo di logica, con tutto ciò che ne consegue in termini di sviluppo, debugging, manutenzione. Inoltre l'applicazione conserverà inevitabilmente la natura relazionale del modello di partenza, risultando strutturata prevalentemente su tale modello piuttosto che sul paradigma di sviluppo ad oggetti, con lo sviluppatore sempre costretto a fare i conti con una traduzione più o meno esplicita da un modello all’altro.

### 7.1.1 Hibernate

Hibernate (talvolta abbreviato in H8) è una piattaforma middleware open source per lo sviluppo di applicazioni Java che fornisce un servizio di Object-relational mapping (ORM). Hibernate è stato originariamente sviluppato da un team internazionale di programmatore volontari coordinati da Gavin King; in seguito il progetto è stato proseguito sotto l'egida di JBoss, che ne ha curato la standardizzazione rispetto alle specifiche Java EE. L'idea generale è quella di avere uno strato di programmazione che si interpone tra l'applicazione ed il database, occupandosi di popolare con le informazioni corrette il proprio database. Hibernate nasce con questo scopo e si evolve fino a diventare forse la più robusta piattaforma di middleware per la persistenza attualmente presente. Il concetto è quella di indicare al middleware in maniera dichiarativa (con dei descrittori testuali, xml) l'associazione tra la classe del software e la tabella in cui risiedono i dati. Una volta definite queste associazioni, l'infrastruttura che si utilizza si occuperà di recuperare dinamicamente le informazioni associate, leggendo i descrittori, e creare automaticamente le query necessarie, in base all'operazione si utilizza.

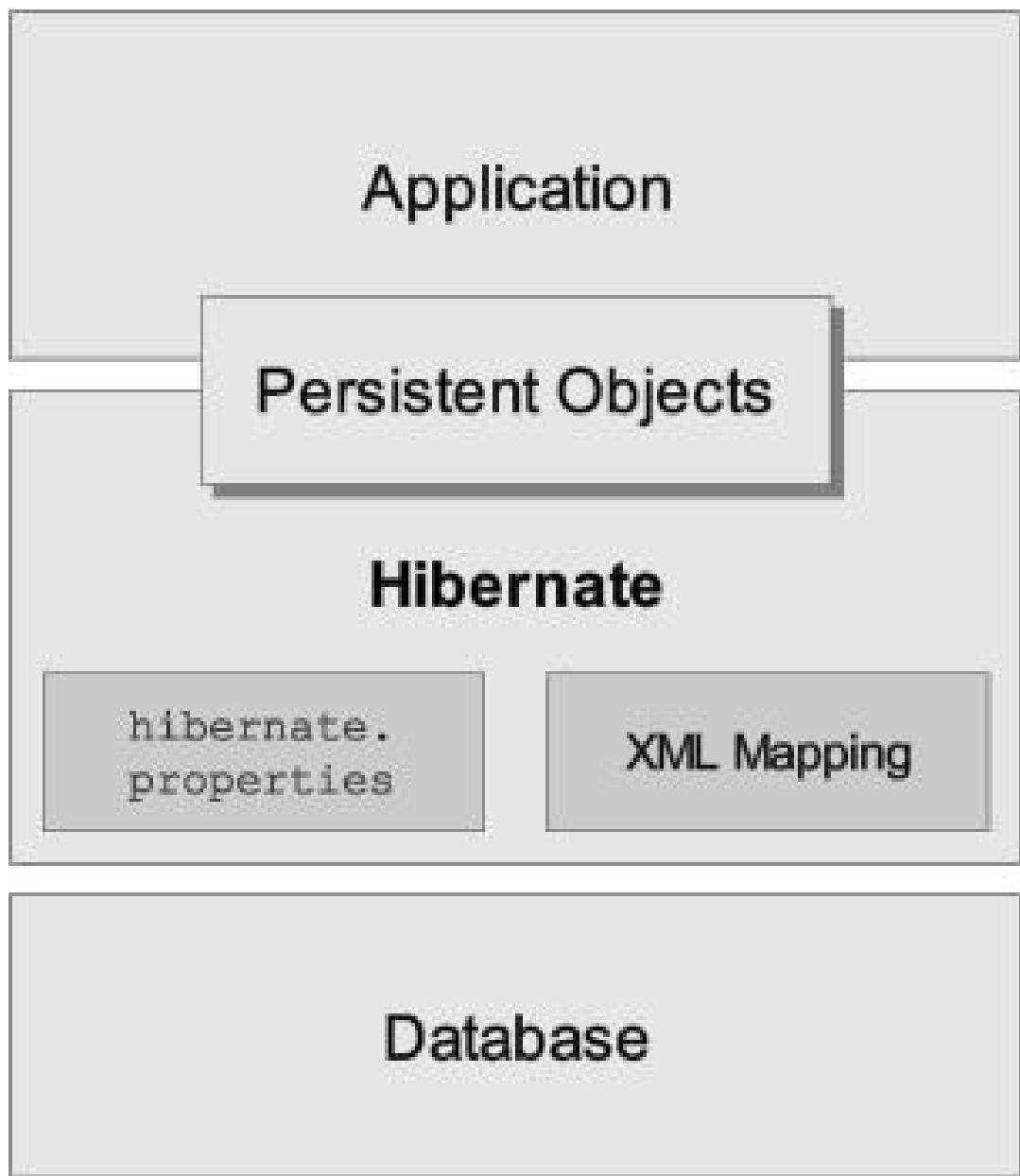


Figura 7.1: Interazione tra il database ed ahibernate

### 7.1.2 Esempi

Dopo aver visto nel capitolo sulla tecnologia GWT come funziona il meccanismo RPC analizziamo come è stato implementato nell'applicazione mostrando degli esempi. Iniziamo mostrando l'interfaccia “Annuncio”:

```
package com.agenzia.immobiliare.client;

import java.util.LinkedList;

@RemoteServiceRelativePath("annuncio")
public interface Annuncio extends RemoteService{

    EAnnuncio carica(int id);

    boolean inserisciGarage(EAnnuncio ann);

    boolean cancella(int id);

    LinkedList<EAnnuncio> cerca();

    boolean inserisciTerreni(EAnnuncio ann);

}
```

Figura 7.2: Implementazione della classe Annuncio.java contenuta nel package Client.Foundation

Le funzioni sono già molto chiare: inserimento, cancellazione e ricerca di annunci. L’implementazione lato server può essere interessante da osservare in quanto per la realizzazione dei metodi sono state usate le funzioni messe a disposizione da hibernate.

```
package com.agenzia.immobiliare.server;

import java.util.LinkedList;

@SuppressWarnings("serial")
public class AnnuncioImpl extends RemoteServiceServlet implements Annuncio {

    @Override
    public EAnnuncio carica(int id) {
        EAnnuncio ann = new EAnnuncio();
        try {
            ann = EAnnuncioDAO.loadEAnnuncioByORMID(id);
        } catch (PersistentException e) {
            e.printStackTrace();
        }
        return ann;
    }

    @Override
    public boolean inserisciGarage(EAnnuncio ann) {
        try {
            PersistentSession s = AgeziaPersistentManager.instance().getSession();
            s.beginTransaction();
            s.save(ann.geteGarage().geteIndirizzo());
            s.save(ann.geteGarage());
            s.save(ann);
            s.getTransaction().commit();
        } catch (PersistentException e) {
            e.printStackTrace();
        }
        return true;
    }
}
```

Figura 7.3: Implementazione della classe AnnuncioImpl.java contenuta nel package Server.Foundation

```
@Override
public boolean cancella(int id) {
    try {
        EAnnuncio eAnnuncio = carica(id);
        EAnnuncioDAO.delete(eAnnuncio);
    } catch (PersistentException e) {
        e.printStackTrace();
    }
    return false;
}

@Override
public boolean inserisciTerreni(EAnnuncio ann) {
    try {
        PersistentSession s = AgeziaPersistentManager.instance().getSession();
        s.beginTransaction();
        s.save(ann.geteTerreni().geteIndirizzo());
        s.save(ann.geteTerreni());
        s.save(ann);
        s.getTransaction().commit();
    } catch (PersistentException e) {
        e.printStackTrace();
    }
    return true;
}
```

Figura 7.4: Implementazione della classe AnnuncioImpl.java contenuta nel package Server.Foundation

Altro esempio interessante può essere quello dell’interfaccia “Installazione”:

```
package com.agenzia.immobiliare.client;

import com.google.gwt.user.client.rpc.RemoteService;
import com.google.gwt.user.client.rpc.RemoteServiceRelativePath;

@RemoteServiceRelativePath("installa")
public interface Installa extends RemoteService{

    String installa(String[] config);

}
```

Figura 7.5: Implementazione della classe Installa.java contenuta nel package Client.Foundation

Anche in questo caso le funzioni sono abbastanza esplicative ma come possiamo vedere, qui tutto il codice è scritto in linguaggio Java puro.

```
package com.agenzia.immobiliare.server;

import java.io.BufferedReader;
import java.io.DataInputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.util.Properties;

import com.agenzia.immobiliare.client.Installa;
import com.agenzia.immobiliare.shared.Controllo;
import com.google.gwt.user.server.rpc.RemoteServiceServlet;

@SuppressWarnings("serial")
public class InstallaImpl extends RemoteServiceServlet implements Installa{

    @Override
    public String installa(String[] config) {
        //installa applicazione
    }
}
```

Figura 7.6: Implementazione della classe InstallaImpl.java contenuta nel package Server.Foundation

```
private boolean scrivi(String[] config) {
    try {
        Properties properties = new Properties();
        properties.setProperty("nomedb",config[0]);
        properties.setProperty("passworddb",config[2]);
        properties.setProperty("posizionedb",config[1]);
        properties.setProperty("usernamedb",config[9]);
        properties.setProperty("portadb",config[10]);
        properties.setProperty("serviziomail",config[4]);
        properties.setProperty("portamail",config[5]);
        properties.setProperty("usernamemail",config[8]);
        properties.setProperty("passwordmail",config[6]);
        properties.setProperty("stile",config[21]);
        File file = new File("config.properties");
        file.createNewFile();
        FileOutputStream fileOut = new FileOutputStream(file);
        properties.store(fileOut, "Configurazione");
        fileOut.close();
    } catch (FileNotFoundException e) {
        return false;
    } catch (IOException e) {
        return false;
    }
    return true;
}

private boolean creaDB(String nome, String pos, String pass, String porta, String user) {
    try{
        Class.forName("com.mysql.jdbc.Driver");
        java.sql.Connection con = DriverManager.getConnection("jdbc:mysql://" + pos + ":" + porta, user, pass);
        try{
            java.sql.Statement st = con.createStatement();
            st.executeUpdate("CREATE DATABASE " + nome);
        }
        catch (SQLException s){
            return false;
        }
    }catch (Exception e){
        return false;
    }
    return true;
}
```

Figura 7.7: Implementazione della classe InstallaImpl.java contenuta nel package Server.Foundation

```
private boolean regioni(String nome, String pos, String pass, String porta, String user){
    try{
        Class.forName("com.mysql.jdbc.Driver");
        java.sql.Connection con = DriverManager.getConnection("jdbc:mysql://" + pos + ":" + porta + "/" + nome, user, pass);
        try{
            java.sql.Statement st = con.createStatement();
            InputStream fstream = new FileInputStream("sql/regioni.sql");
            DataInputStream in = new DataInputStream(fstream);
            BufferedReader br = new BufferedReader(new InputStreamReader(in));
            String strLine;
            while ((strLine = br.readLine()) != null)
                st.executeUpdate(strLine);
            in.close();
        }catch (SQLException s){
        }
        }catch (Exception e){
        }
        return true;
    }

private boolean province(String nome, String pos, String pass, String porta, String user){
    //inserisci provincie
}

private boolean comuni(String nome, String pos, String pass, String porta, String user){
    //inserisci città
}

private boolean queryAgenzia(String nome, String pos, String pass, String porta, String user, String nomeag,
    String passag, String userag, String ind, String fax, String mail, String logo, int tel, String desc) {
    try {
        Class.forName("com.mysql.jdbc.Driver");
        java.sql.Connection con = DriverManager.getConnection("jdbc:mysql://" + pos + ":" + porta + "/" + nome, user, pass);
        java.sql.Statement st = con.createStatement();
        st.executeUpdate("INSERT INTO agenzia VALUES (1, '" + ind + "', '" + nomeag + "', '" + fax + "', '" + userag + "', '" + passag +
            "', '" + mail + "', '" + logo + "', "+tel+", '" + desc + "', 12)");
    } catch (SQLException e) {
    } catch (ClassNotFoundException e) {
    }
    return true;
}
}
```

Figura 7.8: Implementazione della classe InstallaImpl.java contenuta nel package Server.Foundation

## 7.2 Mappa

La mappa è una classe propria delle API di GoogleMaps contenuta nel seguente package: “com.gwt.maps.client.MapWidget.MapWidget”. La ritroviamo in due scenari dell'applicazione : nella visualizzazione della lista dei risultati e nella visualizzazione dei dettagli riguardanti un annuncio. In entrambi i casi il widget corrispondente alla mappa è gestito dalla classe VMapsAdapter, con questo codice:

```
package com.agenzia.immobiliare.client.view;

import java.util.LinkedList;
import java.util.List;

import com.agenzia.immobiliare.client.entity.EAnnuncio;
import com.google.gwt.maps.client.MapOptions;
import com.google.gwt.maps.client.MapTypeId;
import com.google.gwt.maps.client.MapWidget;
import com.google.gwt.maps.client.base.HasLatLng;
import com.google.gwt.maps.client.base.InfoWindow;
import com.google.gwt.maps.client.event.EventCallback;
import com.google.gwt.maps.client.event.impl.EventImpl;
import com.google.gwt.maps.client.geocoder.Geocoder;
import com.google.gwt.maps.client.geocoder.GeocoderCallback;
import com.google.gwt.maps.client.geocoder.GeocoderRequest;
import com.google.gwt.maps.client.geocoder.HasGeocoderRequest;
import com.google.gwt.maps.client.geocoder.HasGeocoderResult;
import com.google.gwt.maps.client.overlay.Marker;
import com.google.gwt.user.client.Window;
import com.google.gwt.user.client.ui.AbsolutePanel;
import com.google.gwt.user.client.ui.Image;
import com.google.gwt.user.client.ui.Label;

public class VMapsAdapter implements IVMapsAdapter{

    public VMapsAdapter(){}
}
```

Figura 7.9: Implementazione della classe VMapsAdapter.java contenuta nel package Client.View

```
public MapWidget getMapbyAddress(int zoom, String ind) {
    MapOptions options = new MapOptions();
    options.setZoom(zoom);
    options.setMapTypeId(new MapTypeId().getRoadmap());
    options.setDraggable(true);
    options.setNavigationControl(true);
    options.setMapTypeControl(true);
    options.setScrollwheel(true);
    final MapWidget map = new MapWidget(options);
    Geocoder geo = new Geocoder();
    HasGeocoderRequest req = new GeocoderRequest();
    req.setAddress(ind);
    geo.geocode(req, new GeocoderCallback() {
        @Override
        public void callback(List<HasGeocoderResult> responses, String status) {
            if (status.equals("OK")){
                HasGeocoderResult res = responses.get(0);
                HasLatLng gLatLng = res.getGeometry().getLocation();
                final Marker marker = new Marker();
                marker.setPosition(gLatLng);
                marker.setMap(map.getMap());
                marker.setClickable(true);
                map.getMap().setCenter(gLatLng);
            }else{
                Window.alert("Geocoder failed with response : " + status);
            }
        }
    });
    return map;
}
```

Figura 7.10: Implementazione della classe VMapsAdapter.java contenuta nel package Client.View

Capitolo 8

## Conclusioni

Dopo aver descritto quello che è il concetto di web oggi e aver delineato il contesto in cui la nostra applicazione si colloca, abbiamo visto come progettare un'applicazione e realizzarla mediante la programmazione ad oggetti, grazie al framework GWT.



Figura 8.1: GWT

Lo sviluppo software è stato ampiamente assistito dal plug-in per Eclipse, che ci

ha assistito durante tutta la stesura del codice grazie all'ampio e dettagliato JavaDoc GWT.

Un grande vantaggio è stato quello di poter utilizzare gli strumenti della programmazione ad oggetti nonostante Javascript non sia ad oggetti. Questo oltre a semplificare notevolmente la soluzione, ci ha permesso di organizzare l'intero sviluppo software mediante modelli tipici della programmazione ad oggetti, come il modello iterativo incrementale Unified Process.

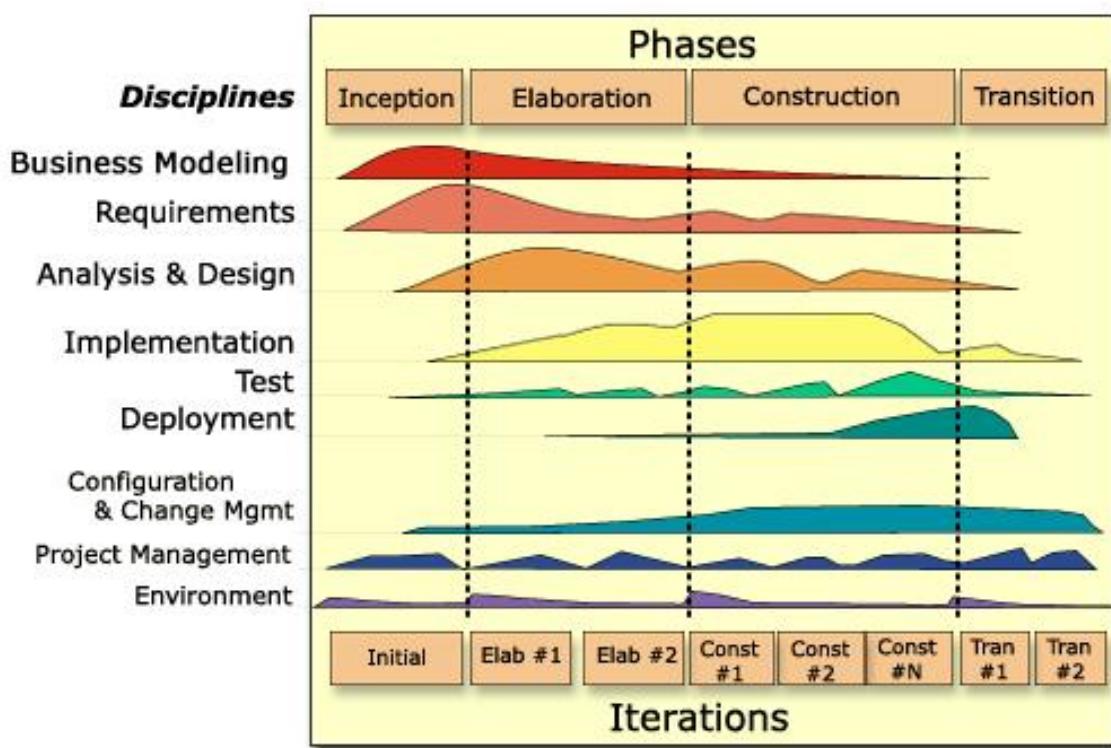


Figura 8.2: Fasi di sviluppo del modello iterativo incrementale Unified Process

L'applicazione fornita come soluzione al problema di fornire supporto ad agenzie immobiliari di piccole dimensioni, rispecchia le specifiche iniziali e sfrutta le potenzialità offerte dalle mappe di google. Il progetto è composto da 67 classi Java, ed oltre 8000 linee di codice effettive (tolti commenti e linee vuote), i file sorgenti anno in totale una dimensione di 385.525 byte. Ai fini di tesi l'applicazione non è stata implementata in tutte le sue parti, poiché avrebbe richiesto ulteriore tempo:

mancano, infatti, l'implementazione della parte relativa alla gestione degli annunci residenziali e del confronto tra due annunci.

# Bibliografia

- [1] Ryan Dewsbury, Google Web Toolkit Application, Prentice Hall, 2008
- [2] <http://code.google.com/p/gwt-google-apis>
- [3] Davide Russo - Università degli studi dell'Aquila - Facoltà di Ingegneria, Tesi: "Progetto e sviluppo di <<SETTLE>>: applicazione web per lo scheduling di meetings mediante la gestione di tempo e luoghi", A.A. 2009/2010
- [4] Pagina ufficiale del Toolkit,  
<http://code.google.com/intl/it-IT/webtoolkit/>
- [5] Showcase del Toolkit,  
<http://gwt.google.com/samples/Showcase/Showcase.html>
- [6] Google Code, Tutorial RPC,  
<http://code.google.com/intl/it-IT/webtoolkit/doc/latest/tutorial/RPC.html>
- [7] Ext-GWT,  
<http://code.google.com/p/gwt-ext/>
- [8] Google Code, Tutorial Build Application,  
<http://code.google.com/intl/it-IT/webtoolkit/doc/latest/tutorial/gettingstarted.html>
- [9] Hibernate wikipedia,  
<http://it.wikipedia.org/wiki/Hibernate>

- [10] Oggetti persistenti con Hibernate, MokaByte 101 - Novembre 2005,  
<http://www2.mokabyte.it/>
- [11] Introduzione ad Hibernate, Pasquale Congiustì, 29 Ottobre 2007,  
<http://java.html.it/articoli/leggi/2421/introduzione-ad-hibernate/>
- [12] Ajax wikipedia,  
<http://it.wikipedia.org/wiki/AJAX>
- [13] Guida AJAX, Andrea Giammarchi,  
<http://javascript.html.it/guide/leggi/95/guida-ajax/>
- [14] AJAX Tutorial, W3Schools,  
<http://www.w3schools.com/ajax/default.asp>
- [15] Web 2.0 wikipedia,  
<http://it.wikipedia.org/wiki/Web2.0>
- [16] Cos'è Il Web 2.0: Definizione E Mini-Guida Di Robin Good, Robin Good, 7 ottobre 2005,  
<http://www.masternewmedia.org/it/>
- [17] World Wide Web wikipedia,  
<http://it.wikipedia.org/wiki/WorldWideWeb>
- [18] Web 2.0: una definizione in 10 punti, 19 ago 2006,  
<http://www.dynamick.it/web-20-una-definizione-in-10-punti-534.html>
- [19] Levoluzione del Web: dal 2.0 al 4.0, 8 maggio 2008,  
<http://www.didael.it/sito/blogdida/?p=13>
- [20] Google plugin,  
<http://code.google.com/eclipse>
- [21] What Is Web 2.0, Design Patterns and Business Models for the Next Generation of Software, Tim O'Reilly, 09/30/2005, <http://oreilly.com/web2/archive/what-is-web-20.html>