ALMA MATER STUDIORUM - UNIVERSITÀ DI BOLOGNA

CAMPUS DI CESENA
SCUOLA DI INGEGNERIA E ARCHITETTURA

CORSO DI LAUREA IN INGEGNERIA INFORMATICA

TITOLO DELL'ELABORATO

UNA APPLICAZIONE WEB PER L'ANALISI E LA PRESENTAZIONE DI DATI RELATIVI AL MERCATO IMMOBILIARE ITALIANO

Elaborato in
Fondamenti di Informatica L-A

Relatore
Prof. MIRKO VIROLI

Presentata da
STEFANO FALCONIERI

Sessione I
Anno Accademico 2013/2014

Di cuore,
a papà, mamma, Luca, Lorenzo,
ai nonni,
e a tutti coloro
che hanno sempre
tifato per me.

Indice

Introduzione

Questa tesi è il risultato dell'attività di sviluppo di una applicazione Web per l'analisi e la presentazione di dati relativi al mercato immobiliare italiano, svolta presso l'azienda responsabile del portale immobiliare all'indirizzo *www.affitto.it*.
Lo sviluppo di un'applicazione Web è stata un'esperienza fortemente voluta; l'esigenza di approfondire concetti sempre più presenti nei sistemi informativi odierni è stata motivo di grande entusiasmo e impegno.
Il traguardo poi raggiunto, rappresentato dalla decisione dell'azienda di rendere disponibile il prodotto online in fase beta (fruibile da un'utenza ancora limitata), ha dato un forte senso di compiutezza al lavoro svolto, oltre che a rappresentare per me motivo di grande soddisfazione.

Attività svolta: l'azienda commissiona lo sviluppo di un sistema software che generi uno storico e descriva l'andamento del mercato immobiliare nazionale. L'applicativo Web-based, sarà implementato col supporto di tecnologie Web (PHP,HTML,MySQL,CSS) su portale immobiliare e, a tal scopo, è necessario disporre di funzionalità per l'elaborazione e la successiva presentazione di informazioni correlate agli annunci di vendita/affitto. In particolare dovrà essere realizzata una struttura dati che si sincronizzi

mensilmente con il database acquisendone informazioni essenziali, tra le quali la tipologia dell'immobile, la città ed il quartiere/frazione di appartenenza, i metri quadri, il prezzo. I dati raccolti dovranno essere disponibili agli utenti all'interno del sito tramite grafici interattivi.

Obiettivi della tesi: in questa tesi verrà presentato il processo di sviluppo software che ha portato alla realizzazione del prodotto, ovvero l'insieme ordinato di fasi che coinvolgono le attività, i vincoli e le risorse richieste per produrre uno specifico output che soddisfa una serie di requisiti iniziali. Costruire software significa realizzare un prodotto, sulla base di un progetto, adottando uno specifico processo di sviluppo svolto dal lavoro cooperativo di diverse figure responsabili.
Costruire software con caratteristiche di qualità, non significa solo produrre codice che "funziona", ma anche rispondere a domande essenziali come:

- È stata costruita la "cosa" giusta? (Validazione)
- Il sistema è stato costruito nel modo giusto? (Verifica)
- Il processo logico che ha portato alla produzione del software è documentato?

Altri obiettivi della tesi sono quindi fornire la documentazione adeguata al sistema software realizzato e cercare di rispondere alle domande del "cosa" è stato realizzato, e come lo si è fatto.
Per giustificare l'importanza di tali obiettivi si può fare riferimento al concetto che nell'ingegneria tradizionale (meccanica, edile), il costo del materiale costituisce spesso più del 50% del costo totale di un progetto, mentre nella produzione del software è il costo del lavoro ad essere preponderante: si passa dal 70%, sino ad arrivare al 100%.
L'ingegneria tradizionale ha anche sperimentato che un cambiamento di costo 1 in fase di analisi potrebbe costare 1000 in fase di produzione, per questo si diversificano le fasi di produzione delineando un workflow di cui si è già fatto cenno.

Nella situazione attuale della produzione industriale del software è usuale l'abbattimento di costi di progetto e sviluppo anche limitando le dimensioni del gruppo di lavoro e puntando su uno schema di lavoro del tipo "scrivi, prova e correggi".
In molti casi adeguate fasi di analisi e progettazione hanno luogo, ma questo avviene solo nella mente dei costruttori.
All'aumentare della complessità la mente umana ha bisogno di scomporre il problema in parti di ampiezza limitata, articolando la descrizione in livelli di astrazione diversi: poiché il codice deve esprimere il sistema finale nei minimi dettagli, la maggior parte delle persone sarebbe incapace di leggerlo con profitto (Molesini, et al., 2008)

Risultati. I risultati ottenuti con questa tesi sono:

- Studio del modello della base dati preesistente;
- Analisi, progetto e integrazione dell'applicazione Web sul portale di riferimento;
- Rivisitazione e ottimizzazione del modello del database;
- Gestione ottimizzata degli indici;
- Test e valutazione delle performance: attività che si svolge in contemporanea con l'attività di ottimizzazione del modello e degli indici; si individuano le query "lente" e si interviene sulle stesse, utilizzando tool di supporto forniti dal DBMS.
- Realizzazione ed implementazione del sistema.

La tesi è strutturata in 6 capitoli:

1. Il primo capitolo descrive alcune delle tecnologie di rilievo individuati nel processo di sviluppo dell'applicazione in esame,tra le quali PHP, HTML, CSS, MySQL, portando il lettore a dedurne il processo evolutivo che ha trasformato il Web da document-oriented com'era alla sua nascita, ad application-oriented, com'è allo stato attuale dell'arte. Si concentra poi sul "come" le tecnologie coinvolte

interagiscono e si relazionano tra di loro andando a costituire un'architettura tipica di molte applicazioni Web;

2. Il secondo capitolo tratta della fase di analisi dei requisiti formulati dal committente. Si fa riferimento quindi all'attività di collaborazione col committente e si evince un dominio applicativo, indipendentemente dalle tecnologie realizzative. Si definisce un glossario dei termini, col fine di identificare un significato oggettivo per i termini impiegati, siccome la gran parte degli errori commessi nel processo di sviluppo di un'applicazione Web, deriva da interpretazioni diverse di tali termini, tra committente e analizzatore, o tra gli stessi componenti del team di sviluppo. Si costruisce poi un modello dei casi d'uso con relativi scenari;
3. Il terzo capitolo si concentra sulla fase di analisi del problema, in particolare si cerca di rispondere alle seguenti domande: che ruolo gioca nel contesto di sviluppo di un prodotto software? Qual'è il dominio applicativo del problema? Si presenta quindi l'architettura introducendo una delle pratiche più comuni dell'ingegneria: la scomposizione del problema in sotto-problemi di minore complessità. Il problema è inquadrato da 3 prospettive differenti: struttura, comportamento e interazione;
4. Il quarto capitolo riporta la fase di progettazione, si prova a rispondere alla domanda del "come" è stato risolto il problema. L'intento è quello di mostrare come la fase di progettazione dovrebbe procedere dal generale al particolare, sviluppando sotto-sistemi quanto più è possibile fedeli alle entità attive nella fase di implementazione;
5. Nel quinto capitolo si discute la fase di implementazione, con l'analisi delle soluzioni implementative adottate, ed esempi di codice sorgente. Si studia la complessità del sistema da un punto di vista computazionale-algoritmico piuttosto che da una prospettiva sistemistica-architetturale tipica delle fasi di analisi e di progetto;

6. Il sesto capitolo contiene le conclusioni tratte dall’elaborazione della tesi.

1 Background

1.1 Php

Php è un linguaggio di programmazione che consente di arricchire le pagine Web di codice script che sarà eseguito sul server, permettendo quindi la generazione dinamica del codice HTML.
Il motore di esecuzione di PHP è open source e si presenta tradizionalmente come un modulo da affiancare a un Web server, di solito Apache. Il nome PHP, nel classico spirito open source, è un acronimo ricorsivo che significa PHP: `Hypertext Preprocessor`.
Uno script PHP è una pagina HTML all'interno della quale viene inserito il codice di scripting, per questo motivo viene definito un linguaggio `HTML - Embedded`.
Il codice PHP viene eseguito dal motore presente sul server prima che la pagina Web venga inviata al browser: quest'ultimo riceverà quindi esclusivamente il codice HTML generato dinamicamente dall'esecuzione degli script sul server e non avrà alcun accesso al codice dello script PHP.
La potenzialità di generare codice di markup lato server permette di scrivere applicazioni multicanale che generano il linguaggio di markup in funzione del dispositivo client che ha effettuato la chiamata HTTP.

Per esempio, nel caso in cui si tratti di un classico browser Web, l'applicazione genererà codice HTML, o nel caso sia un dispositivo WAP, l'applicazione potrà generare codice WML.
La multicanalità permette anche, attraverso l'utilizzo di XML per il trasporto dei dati, di far comunicare le applicazioni PHP con Web services esposti da altre applicazioni, magari attraverso il protocollo SOAP.
In questo contesto, PHP si candida come un eccellente alternativa rispetto a linguaggi come Perl e Python, usati più facilmente in ambienti amatoriali, ma anche rispetto a linguaggi più blasonati come Java, o linguaggi server-side di Microsoft come VBScript (nelle `Active Server Pages`) o C# nelle nuove pagine ASPX della piattaforma Microsoft .net. In Figura 1-1 si mostra l'architettura di un'applicazione Web relativa al modello client-server, mentre in figura 1-2 si mostra come si colloca l'interprete Php nel contesto di un'applicazione Web.
Di seguito vengono indicate le caratteristiche che denotano le specificità di PHP:

- È un motore di scripting server side completamente open source;
- È un linguaggio `HTML - embedded`, nel senso che consente di arricchire le pagine HTML con tag proprietari che permettono di inserire, all'interno del codice HTML, gli script PHP che saranno eseguiti sul server;
- Ha una sintassi semplice che assomiglia a quella di altri linguaggi molto utilizzati: C, Java, Perl;
- È utilizzabile su piattaforme diverse (per ora Unix-like e Windows), nel senso che per questi sistemi esiste un'implementazione funzionante del motore di scripting;
- È dotato di un'ampia varietà di librerie (per la gestione di array mono e multidimensionali, librerie grafiche, funzioni matematiche, gestione delle stringhe, connessioni http, gestione completa della posta elettronica, librerie per XML);

- Supporta la connettività a database attraverso componenti standard (`dBase, Informixm, InterBase, mSQL, MySQL, Oracle, PostgreSQL`);
- Può essere installato e configurato come un modulo aggiuntivo del Web Server, ma può anche essere utilizzato come un modulo CGI separato.

Tool principali introdotti con PHP5

Il passaggio da PHP4 a PHP5 ha rappresentato una significativa evoluzione dal punto di vista dello sviluppo di sistemi complessi. Il supporto alla programmazione fornito da PHP4 risultava solido ed adeguato per applicazioni di bassa/media complessità, ma carente di strumenti per l'implementazione di applicazioni più complesse ed in particolare:

- Nell'approccio OOP;
- Nell'interfaccia alle nuove versioni rilasciate da MySQL;
- Nel supporto a XML;

Con la quinta release (http://www.openskill.info) si colmano tali lacune, vengono completamente riscritti i componenti relativi a:

- Object Oriented Programming;
- MySQL: la nuova libreria `Mysqli` fornisce un'interfaccia ottimizzata con connessione SSL al server ed una nuova interfaccia object oriented;
- XML. Il supporto a DOM consente di considerare questa specifica come API, indipendentemente dal linguaggio di programmazione viene introdotta la possibilità di estrarre informazioni da documenti XML.

Altri update effettuati:

- Gestione db totalmente embedded, SQlite offre supporto a `transazioni, subquery, trigger`;
- Iteratori: permettono di iterare su diversi tipi di dato come elenchi di file, risultati di query e anche documenti XML;

- Permette come già avveniva in java o C++ di separare la gestione degli errori dalla logica del programma;
- Gestione di `stream`, `filtri` e `wrapper` come interfaccia di lettura e scrittura su file.

Un esempio di applicazione della libreria Mysqli (http://www.php.net):

```
if (mysqli_query($link, "CREATE TEMPORARY TABLE
    myCity LIKE City") === TRUE) {
    printf("Table myCity successfully created.\n");
}
/* Select queries return a resultset */
if ($result = mysqli_query($link, "SELECT Name FROM
    City LIMIT 10")) {
    printf("Select returned %d rows.\n",
mysqli_num_rows($result));
}
```

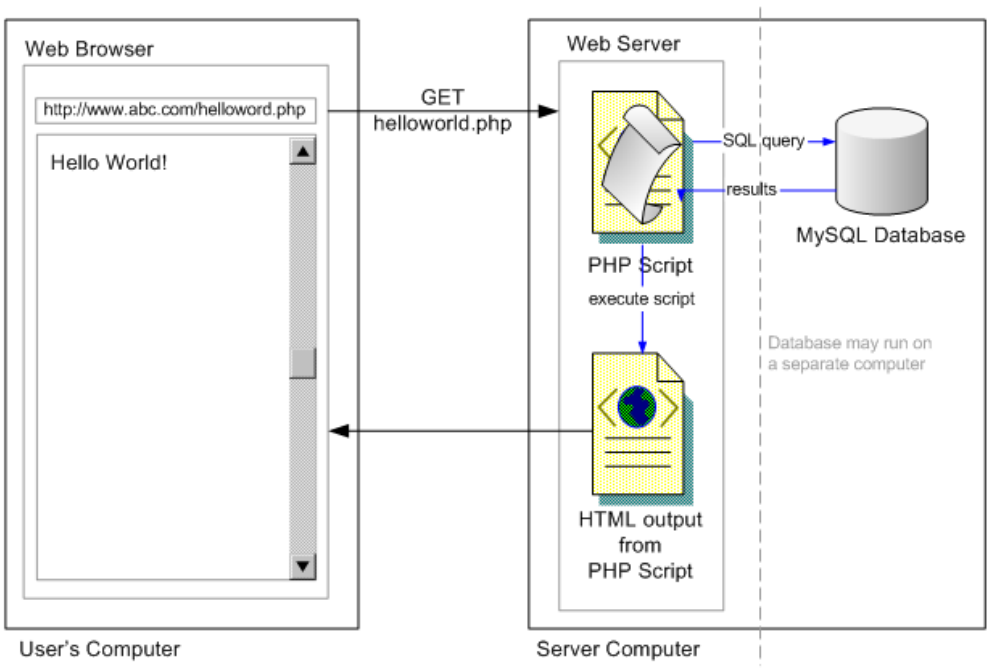

Figura 1-1: Architettura di un'applicazione Web relativa al modello client-server (http://www.hosting.vt.edu)

1.2 MySQL

MySQL è un prodotto software che, insieme ad altri prodotti software quali `Microsoft Access`, `Microsoft SQL Server`, `Oracle Database`, `SyBase`, `DB2`, `INGRES`, `MySQL` e `PostgreSQL` appartiene alla categoria dei DBMS relazionali, o RDBMS.

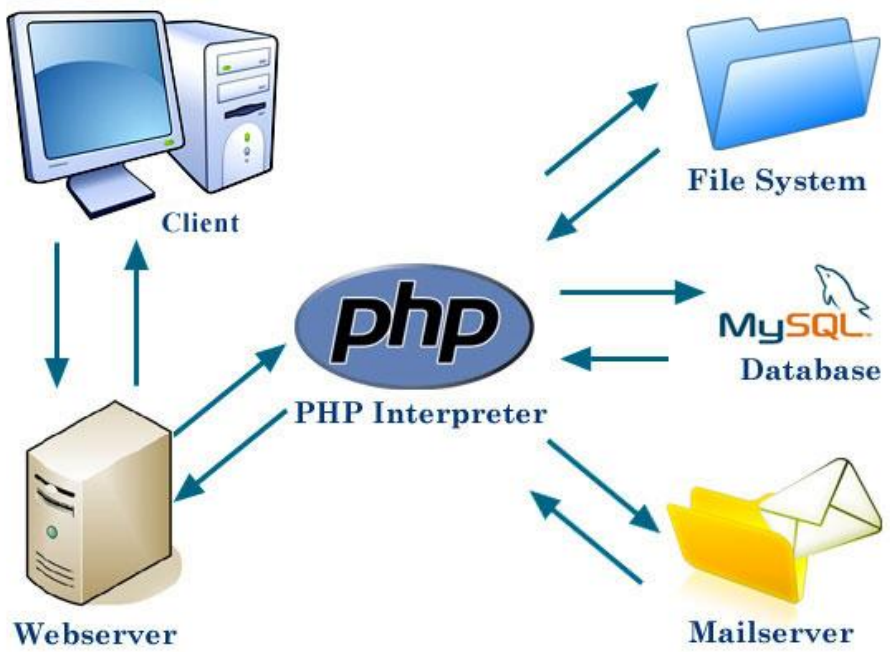

Figura 1-2: L'interprete php nel contesto di un'applicazione Web (https://forum.congnghethongtin.org)

Sarebbe stato proibitivo pensare allo sviluppo dell'applicazione web trattata in questo documento senza il supporto di un RDBMS, che svincola lo sviluppatore dalla logica delle più comuni necessità di interazione col database; offre quindi tutti i servizi necessari all'organizzazione e alla manutenzione della base dati, tra cui i seguenti (Oppel, 2006):

- Spostamento di dati da e verso i file di dati fisici, secondo necessità;

- Gestione di accesso contemporaneo ai dati da parte di più utenti, con misure che impediscono il conflitto tra aggiornamenti simultanei;
- Gestione di transazioni, in modo che ogni modifica al database relativamente alla transazione sia trattata come singola unità di lavoro. In altre parole se la transazione viene completata correttamente, tutte le modifiche apportate al database da tale transazione sono registrate nel database; caso contrario, nel database non viene registrata alcuna modifica. Si noti che alcuni DBMS relazionali non dispongono del supporto alle transazioni;
- Supporto per un linguaggio di query, il sistema di comandi utilizzati da un utente di database per il recupero di dati dal database; SQL è il linguaggio più utilizzato con DBMS relazionali;
- Metodi per il backup del database e il recupero del database da errori;
- Meccanismi di sicurezza per impedire l'accesso e modifiche ai dati non autorizzate.

I principali tool di interazione con il database forniti da MySQL sono i seguenti:

- `mysqld`. Questo è il server MySQL che accoglie le richieste (interrogazioni) dei client. Lo script `mysqld_safe` è il modo più comune di lanciare il server, in quanto lo fa ripartire qualora esso termini in modo anomalo;
- `mysql`. Questo è il client MySQL che permette connettersi al server e eseguire le interrogazioni SQL;
- `mysqladmin`. Serve per amministrare il server MySQL, ad esempio per terminarlo;
- `mysqlshow`. Serve per ottenere informazioni sulla base di dati, sulle tabelle e sulle colonne;

- `mysqldump`. Serve per esportare schemi e tabelle su file di testo. Il client mysql oppure il programma mysqlimport sono in grado di caricare tabelle leggendole dai file esportati;
- `mysqlhotcopy`. Serve per esportare una base di dati o singole tabelle nel formato interno usato da MySQL. Per recuperare la base di dati basta copiare i file esportati nella cartella dei dati di MySQL.

In figura 1-3 è illustrata l'architettura MySQL nel modello client-server.

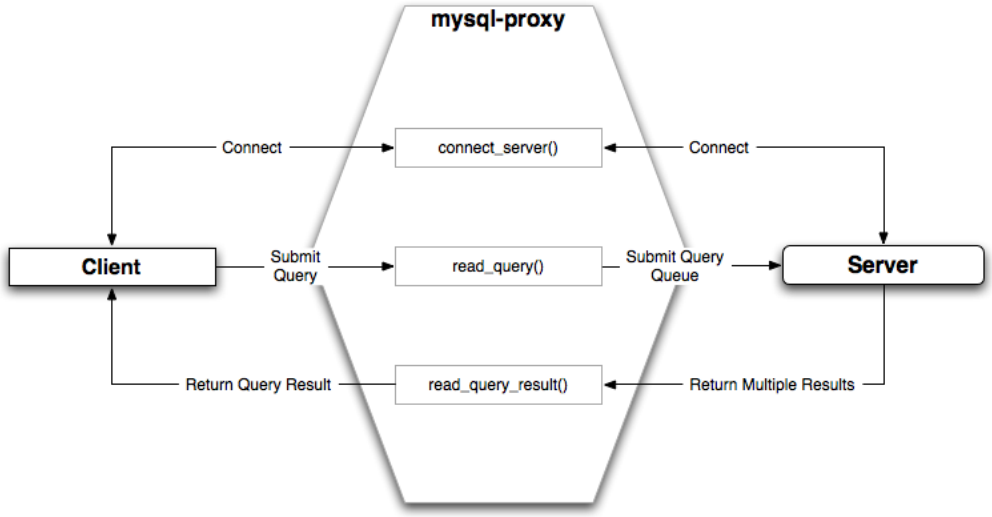

Figura 1-3: Architettura MySQL nel modello client-server (http://dev.mysql.com)

Perché scegliere MySQL

Test effettuati in precedenza hanno sempre dimostrato la competitività di MySQL nella folta rappresentanza di DBMS presenti sul mercato. Di seguito ne sono riportati i vantaggi (Weeling, et al., 2004).

- **Prestazioni:** Al centro della progettazione di MySQL c'è sempre stata la rapidità di esecuzione. Si è provveduto ad aggiungere nuove caratteristiche al prodotto solo quando

questo non comportava un calo di prestazioni. Questo ha determinato aggiornamenti più lenti nel corso della sua storia, ma assicurandone costantemente prestazioni elevate;

- **Prezzo:** MySQL viene distribuito con una doppia licenza: una licenza commerciale (a pagamento), che consente di includere le funzionalità di MySQL nello sviluppo di un proprio software e vendere tale software con licenza commerciale. Una licenza libera (GNU General Public License, GPL), che consente si scaricare liberamente i sorgenti e gli eseguibili, modificare i sorgenti e ridistribuirli a patto che il prodotto creato sia distribuito con la licenza GPL. In ogni caso, MySQL AB mantiene i diritti sui sorgenti e sul marchio MySQL, che quindi non può essere usato come nome nel software distribuito;
- **Stabilità:** Il processo di rilascio di nuove versioni prevede un processo mirato a preservare la stabilità del sistema, questo processo verifica le funzioni e le altre caratteristiche, oltre ai risultati delle operazioni di cui sia stato corretto un bug; gli sviluppatori devono anteporre la correzione degli errori all'introduzione di nuove funzionalità. Il sistema di segnalazione e correzione degli errori di MySQL è pubblico, ed avere più di 4 milioni di utenti nel mondo che partecipano attivamente, costituisce un importante contributo alla solidità del sistema;

In Figura 1-4 si mostrano le principali funzionalità introdotte con le ultime versioni di MySQL, e di seguito ne è riportata una breve descrizione.

Trigger

Un trigger è una funzionalità associata a una tabella che viene eseguita quando si verifica un determinato evento sulla tabella stessa. Per esempio, si può creare un trigger per una tabella che si attiva ogniqualvolta viene inserita una nuova riga. I nuovi comandi

che supportano i trigger includono `CREATE TRIGGER` e `DROP TRIGGER`.

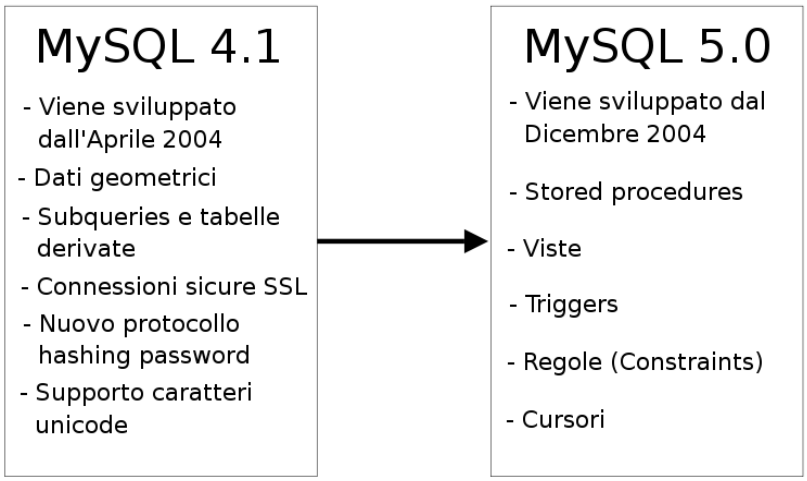

Figura 1-4: Funzionalità aggiuntive introdotte con MySql 5.0 (http://www.miamammausalinux.org)

Stored procedure

Sono una caratteristica che gli utenti MySQL hanno atteso per diverso tempo. Una `stored procedure` equivale ad una funzione scritta interamente in SQL e memorizzata all'interno del database. Le `stored procedure` sono utili per incapsulare un numero di istruzioni SQL che vengono sempre eseguite insieme dai client facendo riferimento a un unico nome logico. Questi i nuovi comandi che supportano le `stored procedure`: `CREATE PROCEDURE`, `ALTER PROCEDURE`, `DROP PROCEDURE`, `CALL`, `BEGIN/END`.

Nuovi storage engine

I più comuni `storage engine` di MySQL sono MyISAM e InnoDB, ma ne sono stati introdotti di nuovi nelle più recenti versioni. `ARCHIVE`, `BLACKHOLE`, `CSV`, `FALCON`, `FEDERATE`, `MERGE`.

Gli eventi

Gli eventi permettono di predisporre l'esecuzione di codice SQL in un determinato istante futuro o secondo un calendario ricorrente.

Considerando come caso concreto `il trigger`, si può aggiungere che quest'ultimo, definibile come una procedura, viene relazionato ad un evento che può verificarsi su una determinata tabella. Allo scatenarsi dell'evento, viene eseguito il codice `T-SQL` relativo al `trigger`. Un esempio pratico di trigger è il seguente:

```
CREATE TRIGGER TR_DEL_Employees
ON Employees
FOR DELETE /* , INSERT, UPDATE più azioni
contemporaneamente */
AS
INSERT CrologiaImpiegati
SELECT EmployeeID, FirstName, LastName, 'Eliminato'
AS Azione
FROM deleted
```

`Create trigger` è l'istruzione che fisicamente crea il trigger `TR_DEL_Employees`, la parola chiave `ON` invece ci dice invece la tabella sulla quale viene ancorato mentre la parola chiave `FOR` indica a quali eventi viene associato.

1.3 Css

I CSS o Cascading Style Sheet sono stati introdotti dal W3 Consortium come complemento del linguaggio HTML. Il W3C infatti ha sempre considerato l'HTML come un linguaggio dedicato alla definizione della struttura del documento, privilegiando il contenuto piuttosto che la presentazione dello stesso. L'introduzione dei CSS ha apportato significative novità nel modo di concepire un documento, diviene evidente infatti la separazione di struttura, contenuto e stile.
La prima specifica dei fogli di stile risale al 1996 (CSS1), mentre la seconda versione (CSS2) è stata rilasciata nel 1998, fino ad arrivare alla definizione della terza versione (CSS3).

Una caratteristica di rilievo che differenzia i CSS3 dal loro predecessore è la modularità (http://www.interact.it). Ogni modulo è dedicato ad un particolare aspetto dei CSS3, tra i quali sintassi, selettori, box model, testo, colori). Questo approccio cambia considerevolmente l'interazione dei browser con questa tecnologia, infatti la logica del browser non deve necessariamente implementare intere specifiche, ma supportare il modulo opportuno, in base alle esigenze. Dal punto di vista software per esempio, un reader non implementerà il modulo video, svincolandosi di tale specifica.
Un esempio di modulo (http://www.tomstardust.com):
Considerato un contenitore e degli elementi al suo interno, questi sono in grado di adattarsi allo spazio vuoto. Considerando come HTML il seguente frammento di codice sorgente:

```
<div class="container">
    <div class="box"></div>
    <div class="box"></div>
</div>
```

Le regole CSS funzionali al far adattare verticalmente i due div interni sarà:

```
.container {
    display: box;
    box-align: stretch;
    box-orient: vertical;
}
.box {
    box-flex: 1;
}
```

Per ciò che concerne la definizione di nuovi moduli CSS3, lo sviluppo, a differenza delle specifiche precedenti, prevede il

passaggio obbligato di alcuni step fondamentali (http://www.w3c.it):

- In prima fase, o anche `Working draft` sono definite le specifiche;
- A conclusione del lavoro viene emessa l'ufficialità mediante un annuncio detto `Last call`;
- L'esito positivo della prima fase permette al modulo di candidarsi come possibile standard, passando allo stato successivo di `Candidate recommendation`;
- Il modulo è sottoposto a giudizio di un comitato ufficiale W3C;
- L'ultima fase nella quale si definisce il modulo come standard, nel caso in cui c'è stata approvazione da parte del comitato; Il modulo diviene quindi *recommendation*.

Gli ultimi aggiornamenti introdotti sono (http://www.w3.org):

- 03-04-2014 CSS WG pubblica due *Working Drafts: CSS Scoping Module Level 1* and *CSS Line Grid Module Level 1*
- 06-05-2014 CSS WG ufficializza una *Last Call* del modulo: *CSS Custom Properties for Cascading Variables Module Level 1*
- 13-05-2014 CSS WG aggiorna due *Working Drafts*: *CSS Generated Content for Paged Media Module and CSS Grid Layout Module Level 1*;
- 03-06-2014 CSS WG pubblica il primo *Working Draft* del modulo *Non-element Selectors Module Level 1*;
- *05-06-2014 CSS WG* pubblica il primo *Working Draft* del modulo *Media Queries Level 4;*

1.4 Architettura

Come, tali tecnologie, si relazionano tra di loro per lo sviluppo di un'applicazione web?

Interfaccia PHP-MySQL
Come già accennato nella descrizione delle novità introdotte con la release 5 di PHP, si possiedono i tool necessari per l'interazione ad alto livello con il server MySQL. Le API implementate forniscono allo sviluppatore tutti gli strumenti necessari per un approccio alla programmazione procedurale o anche orientato agli oggetti, nascondendone la logica legata ad aspetti più a basso livello.
Un esempio di utilizzo della libreria nell'approccio Object oriented (http://www.php.net):

```
$mysqli->real_query("SELECT * FROM friends");

if ($mysqli->field_count) {
    /* this was a select/show or describe query */
    $result = $mysqli->store_result();

    /* process resultset */
    $row = $result->fetch_row();

    /* free resultset */
    $result->close();
}
```

Integrazione delle regole CSS nel codice sorgente HTML
Un foglio di stile CSS può definirsi interno o esterno. Il foglio di stile esterno è separato dal documento HTML ed ha estensione .css. Il foglio di stile interno invece, presenta il suo codice all'interno del file HTML e precisamente nella parte di codice definita dal tag `<style>`, a sua volta dichiarato nell'elemento html `<head>`.

Esistono 2 modi per far riferimento a regole CSS esterne al documento (http://www.html.it):

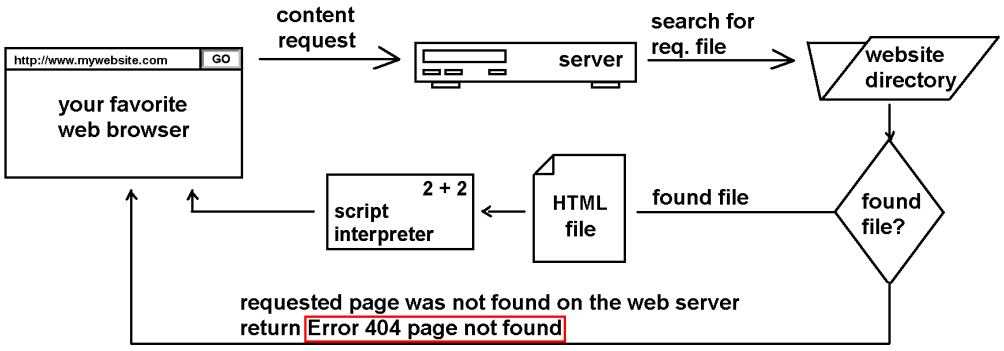

Figura 1-5: Integrazione delle tecnologie HTML e PHP nel modello client-server

Con l'utilizzo del tag `<link>`:

```
<html>
 <head>
  <link href="stile.css" rel="stylesheet"
   type="text/css">
 </head>
 <body>
 [...]
</html>
```

E con l'utilizzo della direttiva `@import` all'interno dell'elemento `<style>`:

```
<html>
 <head>
  <style>
   @import url(stile.css);
  </style>
 </head>
 <body>
```

```
 [...]
</html>
```

Interazione PHP - CSS

PHP è in grado di rendere dinamici non solo i contenuti di una pagina Web, ma anche la presentazione della stessa: in quest'ultimo caso l'interazione deve avvenire tra le tecnologie PHP e CSS (http://www.laboratoriocss.it).

```
<div id="banner"></div>

#banner{
    width: 679px;
    height: 300px;
    background: url(randimg.php) no-repeat 0 0;
}
Elemento di interazione.
header('Location: image'.rand(1,93).'.jpg');
```

La Figura 1-5 e la Figura 1-6 mostrano alcuni esempi di integrazione di diverse tecnologie software nell'architettura di un'applicazione Web

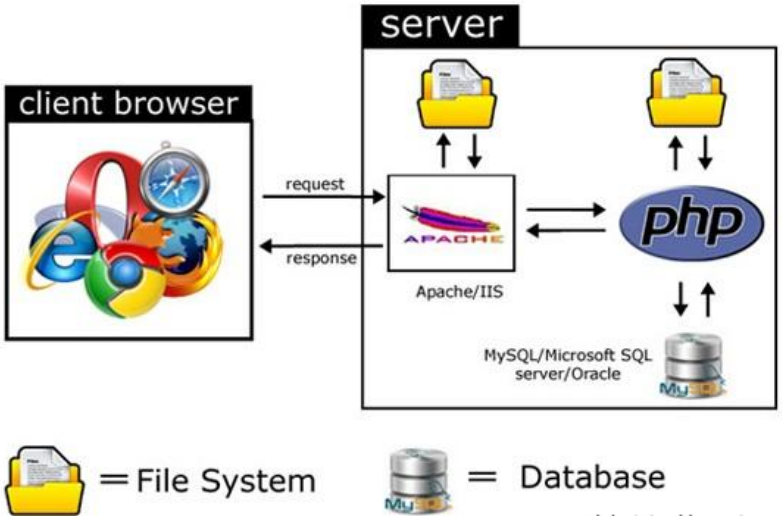

Figura 1-6: Modello di funzionamento e interazione di diverse tecnologie tipico di un'applicazione Web (http://phpdevtutes.blogspot.it)

2 Analisi dei Requisiti

2.1 Introduzione

L'azienda Piano B SRL commissiona lo sviluppo di un sistema software che generi uno storico e descriva l'andamento del mercato immobiliare nazionale. L'applicativo sarà implementato su portale immobiliare "affitto.it". A tal scopo, è necessario disporre di funzionalità per l'elaborazione e la successiva presentazione di informazioni correlate agli annunci di vendita/affitto. In particolare dovrà essere realizzata una struttura dati che si sincronizzi mensilmente con il database di affitto.it acquisendone informazioni essenziali, tra le quali la tipologia dell'immobile, la città ed il quartiere/frazione di appartenenza, i metri quadri, il prezzo. I dati raccolti dovranno essere disponibili agli utenti di affitto.it all'interno del sito tramite grafici interattivi.

2.2 Visione

Un `workflow` sarà seguito come supporto e ottimizzazione del processo di realizzazione del software, seguendo un approccio `top-down`. La caratteristica di tal metodo è l'organizzazione ottimizzata

del lavoro, attraverso un processo “a spirale” con prototipi funzionanti alla fine di ogni iterazione, con l’obiettivo di realizzare un prodotto software quanto più possibile riusabile, modulare, ed estensibile.
Ogni spirale è composta dall’ analisi dei requisiti, dall'analisi del problema, dal progetto, e dall’implementazione. Alla fine di ogni ciclo è prevista la stabilizzazione e la regolazione dell’analisi dei requisiti e di progetto. In questo modo le specifiche del problema sono definite , poi passo dopo passo gli aspetti relativi alla risoluzione del problema si consolidano, focalizzandosi su livelli di astrazione sempre più dettagliati.
Si utilizza il linguaggio UML come riferimento nel team di sviluppo, perchè estraneo a problematiche di ambiguità ed interpretazione tipiche del linguaggio umano e successivamente linguaggi di programmazione web-oriented per l’implementazione.

2.3 Obiettivi

Linearità e accuratezza del modello della base-dati richiesta è considerata una delle priorità nello sviluppo dei requisiti esposti dal committente.
E’ altresì importante, nel criterio di modellazione del database, archiviare informazioni finalizzate ad un utilizzo flessibile ed al contempo immediato. Tali dati dovranno prestarsi ad un utilizzo flessibile ed immediato, senza eccessiva elaborazione.
Il sistema deve disporre di una user interface che permette:

- Di fornire parametri relativamente alle tipologie o posizione dell’immobile richiesto;
- Di visualizzare differenti modalità di andamento del prezzo medio tra cui: prezzo medio, prezzo al metro quadro;

- Di fornire un quadro completo del prezzo medio al metro quadro mensile, per ogni tipologia di immobile;
- La gestione del `form`: i parametri di ricerca devono "ricordare" l'ultima configurazione scelta, oppure nel caso di primo accesso alla pagina, fare riferimento ai parametri di ricerca selezionati in altre sezioni del sito. Ogni `selectbox`, deve poter interagire con l'esterno in modo tale da offrire un range di valori che dipenda dalla ricerca che si stava effettuando in un'altra sezione del sito (per esempio, se il valore selezionato relativo al parametro città è Milano, allora il parametro relativo al quartiere/frazione deve potersi aggiornare ed offrire come range di valori tutti i quartieri/frazioni relativi).

2.4 Analisi dei requisiti

In questa sezione sono descritti in modo più specifico i requisiti software, e cioè il problema che il sistema deve risolvere. I requisiti di sistema saranno dettagliatamente discussi con il committente, e definiti in tempi brevi.
Dall'analisi dei requisiti si può individuare un primo modello del dominio indipendente dalle tecnologie realizzative e, per quanto possibile, anche dalla specifica applicazione in esame.

2.5 Descrizione dei requisiti

Linearità e accuratezza del modello della struttura-dati di archivio richiesta è considerata una delle priorità nello sviluppo dei requisiti esposti dal committente.
È altresì importante, nel criterio di modellazione del database, archiviare informazioni finalizzate ad un utilizzo flessibile ed al

contempo immediato. Il loro utilizzo sarà richiesto per diversi "scope", e senza eccessiva elaborazione.
Il sistema software deve disporre di una user interface che permette:

- All'utente di fornire parametri relativamente alle tipologie o alla posizione dell'immobile richiesto;
- Al sistema di visualizzare differenti modalità di andamento-prezzo;
- Di fornire un quadro completo del prezzo medio al metro quadro mensile, per ogni tipologia di immobile;
- Al sistema di gestire l'input console (il filtro dati selezionato deve rimanere di default all'ultimo valore selezionato dall'utente).

2.6 Glossario

TERMINE	SIGNIFICATO
Archiviare	Registrare informazioni strategiche nel database
utente	È l'entità che interagisce col sistema.
User-interface	È l'interfaccia del sistema, deve permettere al sistema di interagire con l'utente.
gestire	Il sistema deve implementare una serie di meccanismi per rendere agevole l'esperienza di utilizzo da parte dell'utente.
parametri	Sono le informazioni richieste all'utente dal sistema per fornire un determinato output.

TERMINE	SIGNIFICATO
Tabella tipologie	È l'elemento che fornisce informazioni riguardo l'evoluzione dei prezzi, di tutte le tipologie di immobile di una determinata città

2.6 Casi d'uso e scenari

Il modello presentato in Figura 2-1 è un diagramma che esprime un comportamento, offerto o desiderato, sulla base dei suoi risultati osservabili.
Il diagramma dei casi d'uso individua chi o che cosa ha a che fare con il sistema (attore) e che cosa viene fatto (caso d'uso).
Si tratta tipicamente del primo tipo di diagramma ad essere creato in

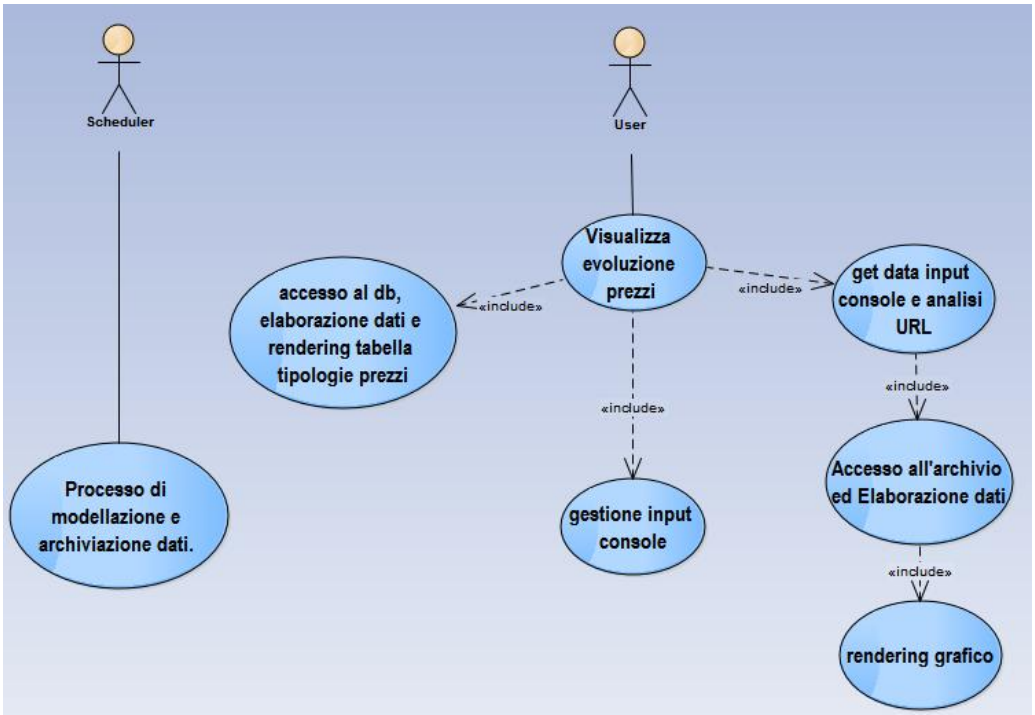

Figura 2-1: Modello dei casi d'uso

un processo o ciclo di sviluppo, nell'ambito dell'analisi dei requisiti e modella i requisiti funzionali di un sistema. Tali requisiti specificano "cosa" deve essere fatto, sono indipendenti dalla tecnologia, dall'architettura, dalla piattaforma, e dal linguaggio di programmazione.
I requisiti non-funzionali invece specificano vincoli aggiuntivi, ad esempio: performance, scalabilità, dimensioni degli eseguibili.
I casi d'uso sono particolarmente utili se la maggior parte dei requisiti sono funzionali, in pratica questo diagramma specifica cosa va fatto, non come va fatto. Spetta poi alle fasi successive capire come realizzare i casi d'uso, la fase dei requisiti invece, li specifica soltanto.
È importante in questa fase scegliere il giusto livello di dettaglio:,infatti una granularità troppo fine rischia di sconfinare nella progettazione, mentre una troppo grossa rischia di generare ambiguità nelle fasi successive. Il diagramma dei casi d'uso dovrebbe risultare comprensibile anche a un non-esperto, in particolare al committente considerando che rappresenta uno strumento per chiarirsi. È importante infine mostrare l'interazione da parte dell'utente con il sistema, in questo caso rappresentati dallo `scheduler` in esecuzione lato server, e l'utente che agisce, lato client.

Si possono individuare in questa fase, due macro-utilità del sistema rappresentate da:

1. Sincronizzazione dell'applicazione con la base dati per la sintesi delle informazioni relative al mercato immobiliare;
2. Fase di ulteriore modellazione e presentazione dei dati richiesti dall'utente;

L'utente che interagisce col `form` per l'inserimento dei dati sul portale www.affitto.it deve avere la possibilità di scegliere "cosa"

visualizzare tramite l'inserimento di parametri di ricerca: `get data input console`.
Il committente ha richiesto una gestione specifica del form ed in particolare di oggetti `selectbox`:

- i valori di default devono essere corrispondenti all'ultimo valore selezionato;
- parametri di selezione (`selectbox` di riferimento) di default che dipendono dalle condizioni iniziali di accesso alla pagina (qual'era la ricerca che si stava effettuando su altre sezioni del portale www.affitto.it?);
- Ogni `selectbox`, deve poter interagire con l'esterno in modo tale da offrire un range di valori che dipenda dalla ricerca che si stava effettuando in un'altra sezione del sito (per esempio, se il valore selezionato relativo al parametro città è Milano, allora il parametro relativo al quartiere/frazione deve potersi aggiornare ed offrire come range di valori tutti i quartieri/frazioni relativi).

La presentazione delle informazioni è organizzata su pagina web secondo 2 modalità: grafico con curva che rappresenta l'andamento dei prezzi nel tempo, in forma specifica e forma tabellare che mostra una panoramica delle medie dei prezzi di tutte le tipologie di immobili, sempre in relazione ai parametri di ricerca selezionati.
La prima modalità, come schematizzato, è gestita dello script per `l'accesso archivio ed elaborazione dati`, che include la computazione relativa al rendering grafico.

Scenari

Field	Description
Name	**V001**

Field	Description
Description	Seleziona ed elabora i dati di interesse dallo storico prezzi e rendering del grafico (che descrive l'andamento del prezzo medio di tutte le tipologie di immobili di un determinato comune e codice tipo annuncio) e della `tabella tipologie`
Actors	User
Preconditions	Primo accesso alla pagina, indirizzo URL secondo le specifiche.
Main scenario	Le opzioni filtro sono impostate di default a "tutte le tipologie", "tutti i quartieri/frazioni" e "prezzo medio". Accesso all'archivio, elaborazione dati e rendering grafico relativo ai filtri.
Alternative scenarios	Se dati non sufficienti visualizza: "dati non sufficienti" e "–" invece del valore corrispondente nella tabella.
Postconditions	Il sistema visualizza in output il risultato .
Name	**V002**
Description	Seleziona ed elabora i dati di interesse dallo storico prezzi e rendering del grafico (che descrive l'andamento del prezzo/mq di tutte le tipologie di immobili di un determinato comune e codice tipo annuncio) e della `tabella tipologie`.
Actors	User
Preconditions	è già stato utilizzato il pulsante "vedi dati".
Main	

Field	Description
scenarios	Le opzioni filtro selezionate sono impostate come segue: “tutte le tipologie”,”tutti i quartieri/frazioni”, “prezzo al mq”. Accesso archivio, elaborazione dati e rendering grafico relativo ai filtri
Alternative scenarios	Se dati non sufficienti visualizza: “dati non sufficienti” e “–“ invece del valore corrispondente nella tabella.
Postconditions	Il sistema visualizza in output il risultato.
Name	**V003**
Description	Seleziona ed elabora i dati di interesse dallo storico prezzi e rendering del grafico (che descrive l’ andamento del prezzo/mq di una determinata tipologia di immobile in un determinato comune) e della `tabella tipologie`.
Actors	User
Preconditions	è già stato utilizzato il pulsante “vedi dati”.
Main scenarios	Le opzioni filtro selezionate sono: “una determinata tipologia”, “tutti i quartieri/frazioni”,”prezzo medio”.Accesso archivio, elaborazione dati e rendering grafico relativo ai filtri
Alternative scenarios	Se dati non sufficienti visualizza: “dati non sufficienti” e “–“ invece del valore corrispondente nella tabella
Postconditions	Il sistema visualizza in output il risultato.

Field	Description
Name	**V004**
Description	Seleziona ed elabora i dati di interesse dallo storico prezzi e rendering del grafico (che descrive l' andamento prezzo/mq di una determinata tipologia di immobile in un determinato comune) e della `tabella tipologie.`
Actors	User
Preconditions	è già stato utilizzato il pulsante "vedi dati".
Main scenarios	Le opzioni filtro selezionate sono: "una determinata tipologia", "tutti i quartieri/frazioni","prezzo al mq". Accesso archivio, elaborazione dati e rendering grafico relativo ai parametri selezionati.
Alternative scenarios	Se dati non sufficienti visualizza: "dati non sufficienti" e "–" invece del valore corrispondente nella tabella
Postconditions	Il sistema visualizza in output il risultato.
Name	**V005**
Description	Seleziona ed elabora i dati di interesse dallo storico prezzi e rendering del grafico (che descrive l'andamento del prezzo/mq di una determinata tipologia di immobile nel quartiere/frazione di un determinato comune) e della `tabella tipologie.`
Actors	User
Preconditions	è già stato utilizzato il pulsante "vedi dati".
Main	Le opzioni filtro selezionate sono: "una determinata

Field	Description
scenarios	tipologia,”un determinato quartiere/frazione”,”prezzo al mq”. Accesso archivio, elaborazione dati e rendering grafico relativo ai filtri.
Alternative scenarios	Se dati non sufficienti visualizza: “dati non sufficienti” e “–“ invece del valore corrispondente nella tabella.
Postconditions	Il sistema visualizza in output il risultato
Name	**V006**
Description	Seleziona ed elabora i dati di interesse dallo storico prezzi e rendering del grafico (che descrive l’andamento del prezzo medio di una determinata tipologia di immobile nel quartiere/frazione di un determinato comune) e della `tabella tipologie`.
Actors	User
Preconditions	è già stato utilizzato il pulsante “vedi dati”.
Main scenarios	Le opzioni filtro selezionate sono: “una determinata tipologia,”un determinato quartiere/frazione”,”prezzo medio”. Accesso archivio, elaborazione dati e rendering grafico relativo ai filtri.
Alternative scenarios	Se dati non sufficienti visualizza: “dati non sufficienti” e “–“ invece del valore corrispondente nella tabella

Field	Description
Postconditions	Il sistema visualizza in output il risultato.
Name	**V007**
Description	Seleziona ed elabora i dati di interesse dallo storico prezzi e rendering del grafico (che descrive l'andamento del prezzo/mq per tutte le tipologie di immobili locati in un determinato quartiere/frazione di un determinato comune) e della `tabella tipologie`.
Actors	User
Preconditions	è già stato utilizzato il pulsante "vedi dati".
Main scenarios	Le opzioni filtro selezionate sono: "tutte le tipologie", "un determinato quartiere/frazione", "prezzo al mq" Accesso archivio, elaborazione dati e rendering grafico relativo ai filtri
Alternative scenarios	Se dati non sufficienti visualizza: "dati non sufficienti" e "–" invece del valore corrispondente nella tabella
Postconditions	Il sistema visualizza in output il risultato.
Name	**V008**
Description	Seleziona ed elabora i dati di interesse dallo storico prezzi e rendering del grafico(che descrive l'andamento del prezzo medio per tutte le tipologie di immobili locati in un determinato quartiere/frazione di un determinato comune) e della `tabella tipologie`.

Field	Description
Actors	User
Preconditions	E’ già stato utilizzato il pulsante “vedi dati”.
Main scenarios	Le opzioni filtro selezionate sono: “tutte le tipologie”, “un determinato quartiere/frazione”, “prezzo medio” Accesso archivio, elaborazione dati e rendering grafico relativo ai filtri.
Alternative scenarios	Se dati non sufficienti visualizza: “dati non sufficienti” e “–“ invece del valore corrispondente nella tabella.
Postconditions	Il sistema visualizza in output il risultato.
Name	**A001**

Description	Processa e modella i dati di interesse in un archivio per un rapido utilizzo. Il sistema si sincronizza una volta al mese con il database di affitto e genera un istantanea del quadro immobiliare del mese in corso, secondo il modello:
	`id_comune`
	`media_mq`
	`media_prezzo`
	`id_tipo_immobile`
	`id_quartiere_frazione`
	`codice_tipo_annuncio`

Field	Description
	`numero_immobili` `mese_anno_corrente`
Actors	Scheduler
Preconditions	Il sistema deve essere configurato correttamente.

Main scenarios	Il sistema si sincronizza con database di affitto e genera un istantanea della situazione immobiliare del mese in corso, secondo il seguente modello:
	`id_comune`
	`media_mq`
	`media_prezzo`
	`id_tipo_immobile`
	`id_quartiere_frazione`
	`codice_tipo_annuncio`
	`numero_immobili`
	`mese_anno_corrente`

Alternative scenarios	**---**
Postconditions	Il sistema permette lo `storage` di dati significativi.
Name	**R001**
Description	Un messaggio “dati non sufficienti” è visualizzato

Field	Description
	dall'utente se le info richieste sono insufficienti o non disponibili. La tabella sostituirà le info mancanti con "—".
Actors	User
Preconditions	---
Main scenarios	Un messaggio "dati non sufficienti" è visualizzato dall'utente se le informazioni richieste sono insufficienti o non disponibili. La tabella sostituirà le info mancanti con "—".
Alternative scenarios	---
Postconditions	Il sistema visualizza in output il risultato.

2.7 Requisiti non funzionali

I requisiti non-funzionali specificano vincoli aggiuntivi, ad esempio: performance, scalabilità, dimensioni degli eseguibili.

3 Analisi del problema

L’analisi del problema comincia con un accurata lettura dei documenti prodotti in fase di analisi dei requisiti e deve produrre una documentazione UML contenente informazioni sul problema con un formalismo non ambiguo. Tale formalismo descriverà il problema attraverso 3 punti di vista: struttura, comportamento ed interazione.
In questa fase ci si colloca ad un livello di astrazione sufficientemente lontano dall’implementazione, viene presentata una prima idea di scomposizione del problema tramite macro componenti.
L’analisi individua i seguenti sotto problemi principali:

- Sincronizzazione mensile degli annunci immobiliari con la banca dati affitto.it;
- Acquisizione parametri di selezione tramite interfaccia, elaborazione dati richiesti;
- Rendering grafico.

3.1 Architettura logica

L'artefatto utilizzato in questa fase dello sviluppo descrive il sistema ed il problema da risolvere visualizzandolo da 3 prospettive differenti e permettendo di rilevare in maniera immediata i concetti principali.

3.2 Struttura

Il diagramma in Figura 3-1 descrive il sistema in fase di analisi del problema dal punto di vista della struttura, è una prima presentazione del dominio applicativo e identifica i macro-sistemi in cui il problema stesso suggerisce di articolare il sistema risolvente.
Ogni entità in gioco, come spesso accade, può essere comune a molteplici casi d'uso, analizzati nella fase precedente. Si possono rilevare già in questa fase i macrosistemi che rappresentano i componenti principali coinvolti nella realizzazione del sistema e la loro interazione.

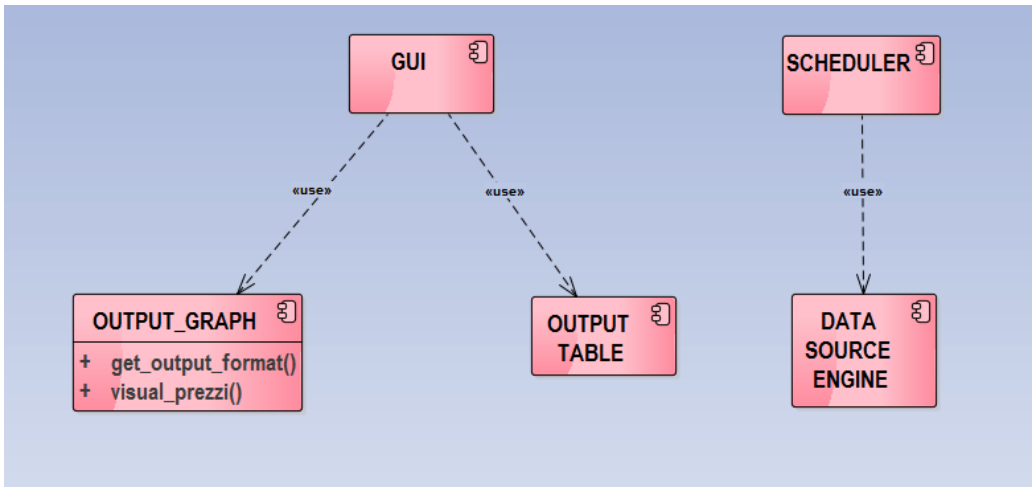

Figura 3-1: Modello della struttura del sistema relativo alla fase di analisi del problema

In particolare, dall'analisi dei requisiti elaborata col committente, si possono inquadrare le seguenti entità:

- `GUI` di presentazione;
- `Output_graph`;
- `Output_table`;
- `Scheduler`;
- `Data_source_engine`.

L'entità `GUI` rappresenta il processo che si occupa di fornire al sistema un'interfaccia per l'interazione con l'utente. Racchiude quindi all'interno non solo una parte prettamente statica ma anche la logica che gestisce il `form` e l'interazione con gli strumenti di supporto `output_graph` e `output_table`, che concettualmente rappresentano i dati richiesti dall'utente, in diverse forme di visualizzazione.
`Scheduler` è un entità di terze parti, rappresenta la logica che attiva mensilmente `l'engine data source`, il processo dedicato allo sviluppo della sorgente dati principale dalla quale attinge l'intera applicazione.

3.3 Comportamento

I diagrammi relativi al comportamento permettono la descrizione del sistema osservato da un punto di vista differente rispetto a quello strutturale sopra-descritto.
Viene sintetizzato il sistema mettendone in evidenza non la struttura, ma le macro-attività coinvolte nel processo di computazione.
Il diagramma degli stati e delle transizioni è quello che meglio soddisfa le esigenze descrittive caratteristiche di questa fase di valutazione; viene definito il comportamento della realtà di interesse

in tutte le sue entità, e si descrivono le leggi che governano l'evoluzione del sistema, relative ad ogni servizio che quest'ultimo eroga.
Uno stato rappresenta una situazione in cui un oggetto ha un insieme di proprietà considerate stabili. Una transizione invece, modella un cambiamento di stato e si colloca banalmente tra uno stato e l'altro.
Il sistema in esame presenta maggiormente una situazione in cui ogni stato, coincide con l'attività svolta ad eccezione dello stato wait, dove invece il sistema non svolge nessuna attività.
L' utilizzo di stati per identificare attività si esprime con la notazione: `do/(Processo)`.

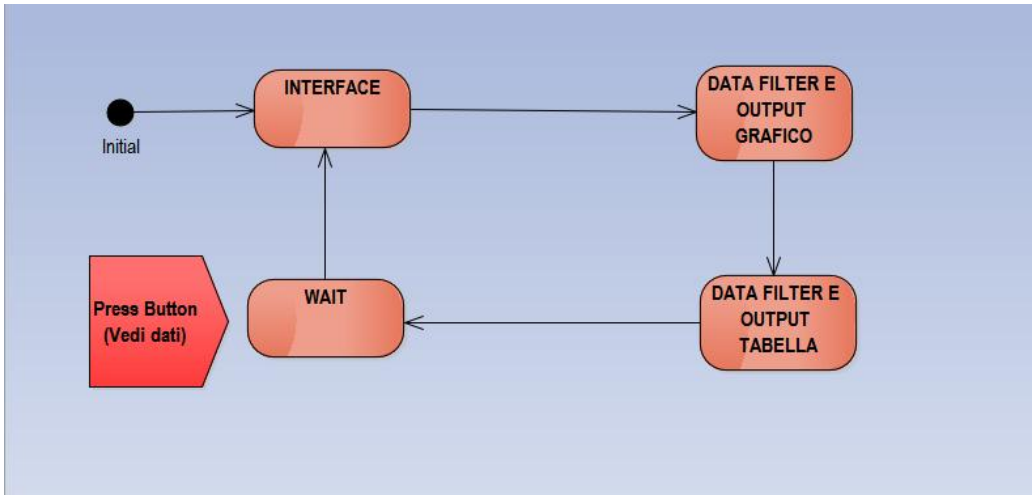

Figura 3-2: Modello generale del comportamento del sistema relativo alla fase di analisi del problema

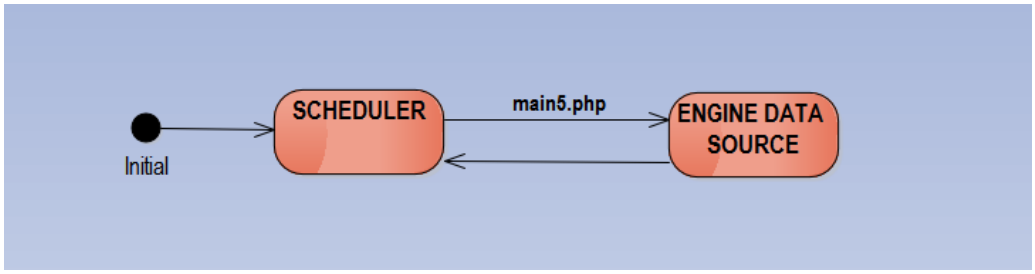

Figura 3-3: Modello del comportamento del sistema relativo alla sincronizzazione mensile dei dati

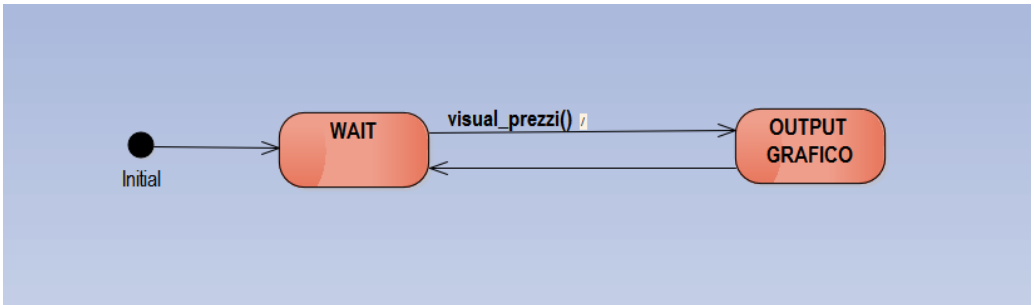

Figura 3-4: Modello del comportamento del sistema relativo all'attività di output grafico

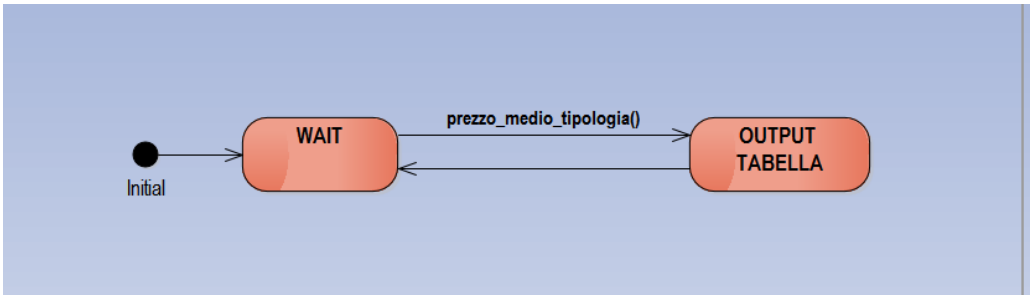

Figura 3-5: Modello del comportamento del sistema relativo all'attività di output tabella

Per ciò che concerne l'analisi sul diagramma in esame, come si mostra in Figura 3-5 le attività principali individuate sono:

- `stato iniziale`: *corrisponde all'accesso sulla pagina raggiungibile tramite indirizzo URL relativo, o link presenti* sul portale web affitto.it. *Rappresenta lo "*start*" dell'intera applicazione;*
- `interface`: questa attività si occupa di fornire al sistema gli strumenti necessari per colloquiare con l'utente. E' inclusa la logica per la gestione del form di ricerca;
- `data filter e output grafico`: l'operazione di data filter, comune ad entrambe le modalità di output rappresenta la logica per il filtro, la selezione e la predisposizione dei dati presente in archivio a partire da parametri di selezione forniti dall'utente;
- `data filter e output tabella`;

- `wait`: tale stato è raggiungibile solo dopo aver fornito risposta alla richiesta avanzata dall'utente. Il sistema si porta in uno stato di attesa, e reagisce solo in occasione di un evento scatenato dall'utente tramite interazione con *form*.

I diagrammi in Figura 3-3, 3-4, 3-5 descrivono il comportamento del sistema relativo ai singoli servizi offerti, specificando la funzione che innesca uno stato/attività.

3.4 Interazione

In Figura 3-6 si mostra il modello di interazione del sistema relativo alla fase di analisi del problema. In questa fase si specificano le interazioni ed il flusso di esecuzione tra gli elementi individuati, rimanendo ad un livello di astrazione funzionale alla comprensione del problema da risolvere.

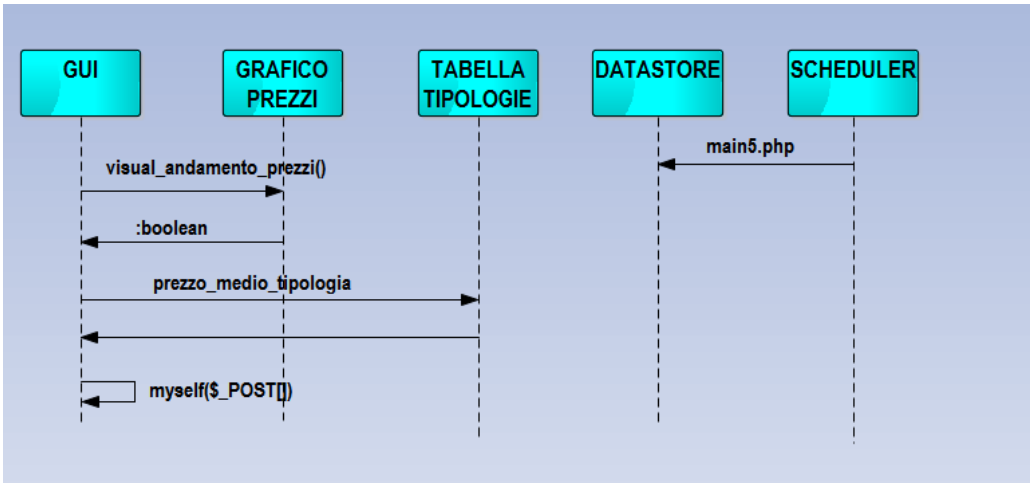

Figura 3-6: Modello di interazione del sistema relativo alla fase di analisi del problema

3.5 Piano di lavoro

Il sistema ha seguito un singolo thread di sviluppo, il lavoro è stato commissionato da Affitto.it al sottoscritto, incaricato alla progettazione ed alla implementazione dell'applicazione Web.
Gli aspetti di collaborazione col committente hanno riguardato l'intero processo di analisi dei requisiti, dove si è reso necessario uno scambio costante di opinioni, finalizzato a definire "cosa" il sistema deve essere in grado di fare, e quali servizi offrire.

4 Progetto

Il progetto è lo sviluppo dettagliato dell'analisi. Mentre l'analisi del problema risponde alla domanda "qual'è il problema?", la fase di progettazione deve rispondere alla domanda: "come il problema può essere risolto?".
In accordo con il piano di lavoro, il problema in questione è stato scomposto in sotto-problemi, più circoscritti, quindi di minore complessità.

4.1 Architettura logica

Cosi come in fase di analisi il sistema è valutato in 3 dimensioni diverse: struttura, comportamento e interazione.

4.2 Comportamento

Lo scopo della fase di progettazione consiste nel raffinamento dell'architettura logica del sistema; deve mirare ad individuare e descrivere una soluzione al problema (what), l'utilizzo di adeguati modelli espressi in UML può risultare fondamentale per il miglioramento della comunicazione e della comprensione reciproca.

Nel modello presentato in Figura 4-1, l'attività interface presentata in fase di analisi è stata sostituita da `job` più specifici: `input analyzer` e `form generator`, mentre l' attività `data filter` e `output grafico` si specializza in `data processor`, `rendering grafico e output`.

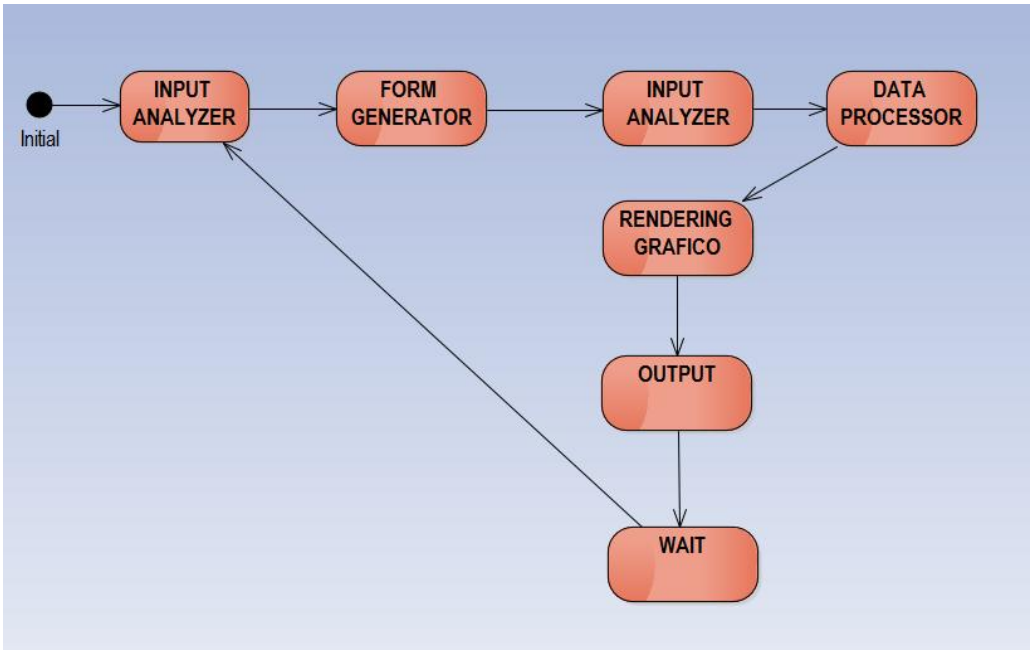

Figura 4-1: Modello generale del comportamento del sistema relativo alla fase di progetto

Lo stato di `wait` attende lo scatenarsi di un evento, in particolare l'interazione da parte dell'utente per far uscire il sistema dalla fase di attesa. `Input analyzer` è di supporto al `form generator`, che dipende dai parametri di selezione settati dall'utente e dallo stato del sistema (prima visita alla pagina). Nel caso invece di preparazione all'attività di `data processor` e `rendering grafico`, `input analyzer` seleziona la modalità di grafico da visualizzare in funzione dei parametri di selezione scelti dall'utente, o dallo stato del sistema.

Input analyzer e form generator

Il diagramma presentato in Figura 4-2 descrive nello specifico i compiti svolti dall' `input analyzer` e `form generator`, che come già detto, sono legate da una dipendenza.

La fase di controllo dell'input che precede il `form generator` è il risultato di una scelta a livello di analisi dei requisiti, discussa col committente.

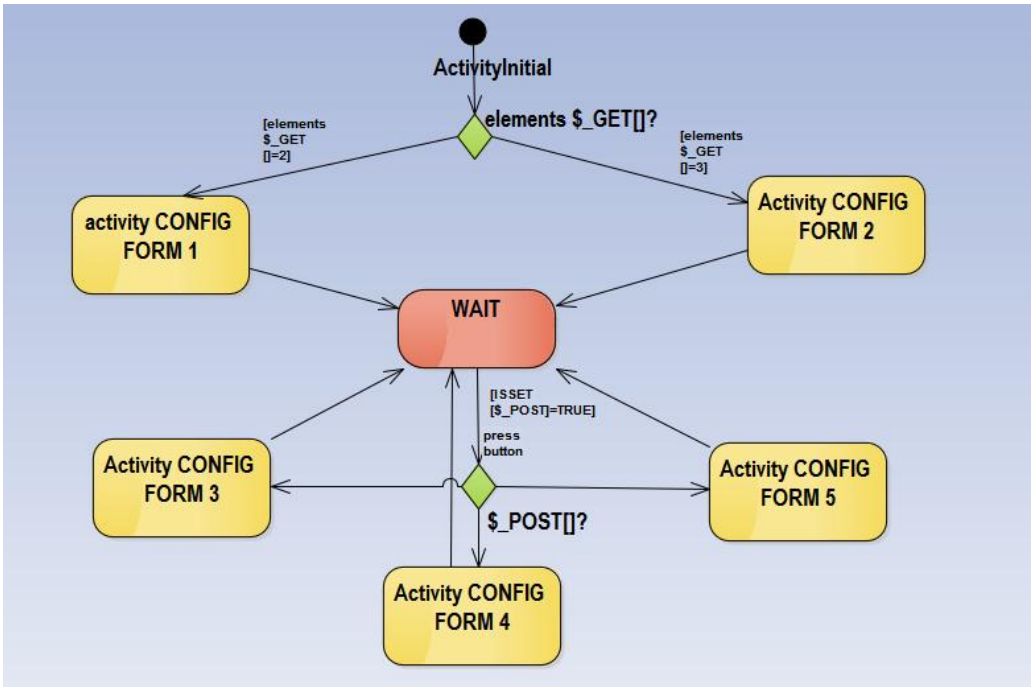

Figura 4-2: Modello del comportamento del sistema relativo ai componenti input analyzer e form generator

Il modello evidenzia la scelta di far riferimento alle variabili universali `$_GET` e `$_POST` per definire la configurazione iniziale del `form`, e quindi dell'intera interfaccia. In pratica si definisce lo stato iniziale del sistema.

Se ne comprende quindi nel dettaglio la logica: le informazioni riguardo l'accesso alla pagina dell'utente che proviene da altri siti, o da altre sezioni del sito affitto.it (o trovandosi sulla pagina in questione re-inserisce l'URL della stessa), sono registrate nella

variabile `$_GET`, dove il numero di elementi dell'array rappresenta il numero di parametri passati tramite URL.
Questo discrimina le possibili configurazioni del `form` o lo stato iniziale del sistema.
L'esecuzione si interrompe ed il sistema è in fase di attesa.
È un `signal` inviato dall'oggetto `form` a riportare il sistema in fase di esecuzione, e precisamente nello stato di `input analyzer`, che questa volta farà riferimento alla variabile universale `$_POST` per definire in quale stato si porterà il sistema. Immaginando il sistema in esame come una macchina a stati finiti, si può notare come gli stati coincidono con l'uscite del sistema e le uscite dipendano solo dagli ingressi e non dagli stati precedenti.

4.3 Interazione

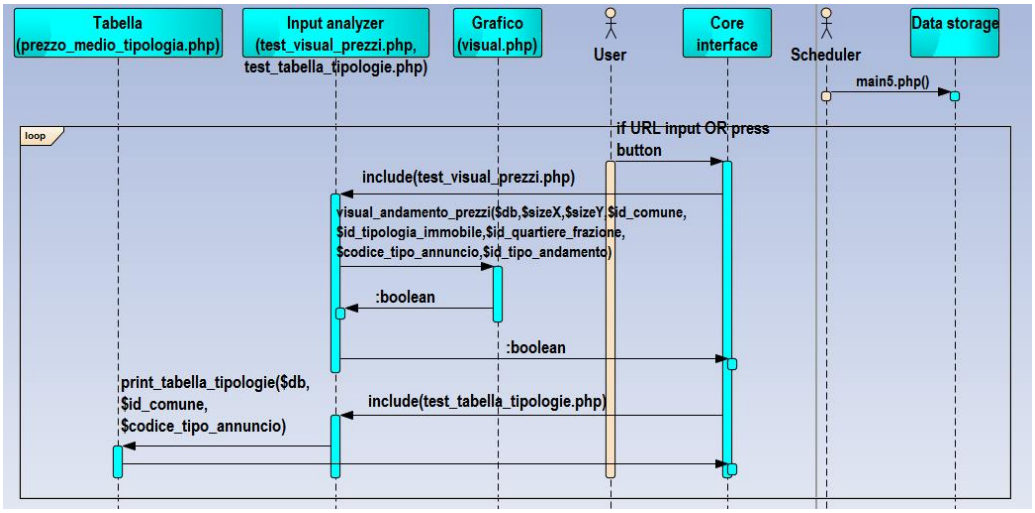

Figura 4-3: Modello di interazione del sistema relativo alla fase di progetto

La sequenza temporale ed il controllo del flusso sono i concetti principali evidenziati dal modello. Lo scopo è mostrare come un certo comportamento viene realizzato dalla collaborazione delle entità in gioco (fino ad ora ci siamo occupati solo di cosa fa il sistema). Può essere usato a vari livelli di astrazione e necessita di:

- Un comportamento da realizzare tratto da un classificatore, come ad esempio un caso d'uso o un'operazione di classe;
- Una serie di elementi che realizzano il comportamento come ad esempio attori o istanze di classe.

Questi elementi provengono da diagrammi creati in precedenza.
Nello specifico, come si evince dal modello in Figura 4-3, lo start del sistema è determinato dall'attore, un input esterno. Il flusso esecutivo passa quindi al `core interface` che effettua una valutazione sul segnale esterno ricevuto e sceglie la configurazione giusta per il `form generator`.
A questo punto il `core interface` attiva l'esecuzione dell'`input analyzer` che sarà poi specifico a seconda del tipo di oggetto richiesto (grafico o tabella), a livello di implementazione, e che precede l'esecuzione degli oggetti grafico e tabella. Il passaggio di flusso da interface all'`input analyzer` non avviene tramite un messaggio o chiamata vera e propria ma con un include, questo per facilitare il passaggio di parametri in fase di implementazione, pur mantenendo modularità del sistema.
L'`input analyzer` quindi utilizza i servizi offerti dalla classe `visual` per l'elaborazione e la formattazione dei dati ed il `rendering grafico`. Il controllo torna infine al `core interface`.

5 Implementazione

5.1 Studio e modellazione del database

Modello affitto_archivio

La struttura "container" di tutte le informazioni significative è stata ideata e concretizzata dopo una fase di valutazione col committente. Il modello dati in questione è il `main source` dell'intera applicazione. È quindi determinante ai fini dell'attendibilità delle informazioni che l'utente richiede, porre particolare attenzione allo studio di questa fase di modellazione.

L'organizzazione dei dati presentati deve permettere un adeguato livello di astrazione, nello specifico si auspica un buon compromesso tra flessibilità di impiego e utilizzo delle informazioni senza un' eccessiva elaborazione. Si punta ad un informazione con un buon grado di generalità -quindi capace di adattarsi a diversi ambiti di utilizzo- ed al contempo sintetica, quindi "di immediato utilizzo".

Oltre a disporre di una buona flessibilità tale organizzazione permette un'analisi veloce da parte del committente, che in tal modo accedere direttamente ai dati significativi e valutare la situazione del mercato immobiliare nello specifico, in particolare di ogni singolo comune, tipologia di immobile, quartiere frazione, numero di

immobili, metri quadri e prezzo. In Figura 5-2 e 5-3 è riportato un esempio concreto di organizzazione delle informazioni nella tabella `affitto_archivio`.

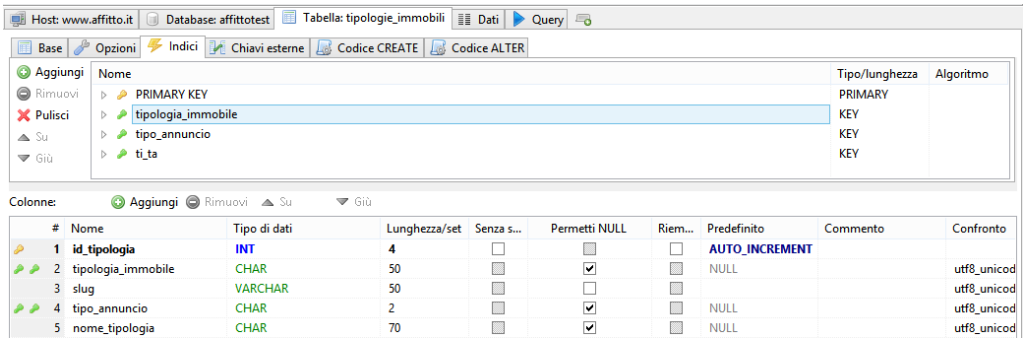

Figura 5-1: Struttura e indici relativi alla tabella *tipologie_immobili*

id_comune	media_mq	media_prezzo	id_tipo_immobile	id_quartiere_frazione	codice_tipo_annuncio	numero_immobili	mese_anno_corrente
93	243	247923	31	0	VI	13	2015-05-15
93	100	400	32	0	OI	1	2015-05-15
93	82	495	57	0	OI	13	2015-05-15
93	130	360000	59	0	VI	1	2015-05-15
93	89	160872	60	0	VI	47	2015-05-15
93	127	200000	61	0	VC	1	2015-05-15
93	30	400	76	0	OC	1	2015-05-15
93	3425	1000000	94	0	VC	2	2015-05-15
93	200	210000	98	0	VC	1	2015-05-15
93	800	258000	103	0	VI	1	2015-05-15
93	484	272500	104	0	VI	2	2015-05-15
93	156	280000	106	0	VI	1	2015-05-15
4075	2039	11383	2	1	OC	12	2015-05-15
4075	870	10751	2	2	OC	17	2015-05-15
4075	350	27000	2	4	OC	1	2015-05-15
4075	1800	32000	2	5	OC	1	2015-05-15
4075	1264	5577	2	6	OC	5	2015-05-15
4075	1180	12500	2	8	OC	1	2015-05-15
4075	510	2500	2	9	OC	1	2015-05-15
4075	429	3430	2	11	OC	18	2015-05-15
4075	700	8000	2	12	OC	1	2015-05-15
4075	1523	10600	2	16	OC	3	2015-05-15
4075	478	3666	2	18	OC	5	2015-05-15
4075	70	800	2	19	OC	1	2015-05-15
4075	567	5019	2	20	OC	7	2015-05-15

Figura 5-2: Modello relativo alla tabella *affitto_archivio*

id_comune	media_mq	media_prezzo	id_tipo_immobile	id_quartiere_frazione	codice_tipo_annuncio	numero_immobili	mese_anno_corrente
4075	115	244143	5	1	VC	7	2015-05-15
4075	112	216900	5	2	VC	10	2015-05-15
4075	150	350000	5	3	VC	1	2015-05-15
4075	110	1085000	5	4	VC	2	2015-05-15
4075	130	204333	5	6	VC	15	2015-05-15
4075	108	398333	5	8	VC	9	2015-05-15
4075	205	269545	5	9	VC	11	2015-05-15
4075	75	157143	5	11	VC	7	2015-05-15
4075	76	326000	5	12	VC	10	2015-05-15
4075	90	130000	5	13	VC	1	2015-05-15
4075	234	1536667	5	14	VC	6	2015-05-15
4075	67	272000	5	15	VC	10	2015-05-15
4075	121	191625	5	16	VC	8	2015-05-15
4075	129	318000	5	17	VC	5	2015-05-15
4075	61	408000	5	18	VC	5	2015-05-15
4075	107	320000	5	19	VC	6	2015-05-15
4075	101	199444	5	20	VC	9	2015-05-15
4075	16	28375	7	1	VI	8	2015-05-15
4075	23	32556	7	2	VI	16	2015-05-15
4075	16	39917	7	3	VI	6	2015-05-15
4075	15	28500	7	4	VI	4	2015-05-15
4075	13	37000	7	5	VI	2	2015-05-15
4075	19	40353	7	6	VI	17	2015-05-15
4075	18	43714	7	8	VI	7	2015-05-15
4075	20	39140	7	9	VI	10	2015-05-15
4075	18	26333	7	10	VI	3	2015-05-15

Figura 5-3: Altro esempio di informazioni organizzate nella tabella *affitto_archivio*

Rivisitazione modello del database

Rivisitazione del modello del database ed in particolare creazione della tabella `tipologie_immobili` per:

1. Processare informazioni sulla base dati utilizzando un indice piuttosto che una stringa. Questo lavoro rientra nel processo di ottimizzazione (o nell'applicazione delle "best practices") del modello relazionale.

2. Omettere dati non necessari: nonostante tutte le informazioni necessarie fossero presenti in `opzioni_valori`, le tipologie degli immobili costituiscono concettualmente un'entità, evitando ulteriore elaborazione per una tabella che contiene informazioni troppo eterogenee.

Uno sviluppo del modello di questo tipo può essere dovuto al non aver immaginato il sistema in proiezione a specifiche funzionalità e servizi, come per esempio quelle attualmente in fase di sviluppo.

Aggiunta del campo nome_tipologia

Fare riferimento a opzioni valori per il campo o replicarlo nella nuova entità creata?

Tutte le valutazioni che si fanno in questo caso dipendono dalla progettazione: in questo caso vale il paradigma della coperta corta, o si guadagna in termini di tempo di sviluppo, o in termini di spazio fisico su disco impiegato; considerando il fatto che la dimensione di *tipologia_immobile* non è eccessiva, è preferibile non coinvolgere più tabelle, in tal modo la complessità delle query diminuisce, a favore di velocità di esecuzione e tempi di sviluppo.

Un eventuale funzionalità da sviluppare in futuro che vede il collegamento di più tabelle tra le quali anche *annunci* o *annunci2* con un milione di record circa, potrebbe gravare non poco sui tempi di computazione del sistema introducendo tempi di latenza; siccome non è semplice prevedere come un sistema evolve nel tempo, allora la scelta è stata determinata da tale motivazione, sicuramente di rilievo.

Valutazione dell'inserimento del campo codice_tipo_annuncio in affitto_archivio

Anche in questo caso la scelta fatta potrebbe introdurre ridondanza. L'alternativa è il collegamento con `tipologie_immobili`, per estrarre il tipo di annuncio. Anche in questo caso il confronto tra i pro, quali la velocità di esecuzione in funzione delle strategie implementative, ed i contro, come la dimensione su disco impiegata, non è significativamente sbilanciato.

Come già discusso per il caso precedente, un eventuale funzionalità da sviluppare in futuro che vede il collegamento di più tabelle tra le quali anche `annunci` o `annunci2` con un milione di record circa, potrebbe gravare non poco sui tempi di computazione del sistema introducendo tempi di latenza. Questo è sufficiente a motivare la scelta in esame.

In generale, il come interagire in futuro con tali dati è un fattore che potrebbe influenzarne decisioni in merito alla struttura del modello.

Dato che il sistema deve comunque avere un margine di espandibilità, il più possibile esteso, il sistema deve mantenere una struttura con specificità con elevata, in particolare è preferibile che il modello non dipenda da scelte future, che in questo caso hanno un bassissimo indice di previsione.

5.2 Ottimizzazione modello del database

Per comprendere l'importanza degli indici è rilevante capirne il funzionamento (Du Bois, 2004). Per farlo, si considera una tabella che ne sia priva. Una tabella senza indici è banalmente una collezione disordinata di righe. Per trovare quale valore corrisponde ad una determinata società, per esempio, è necessario esaminare ogni riga della tabella per vedere se corrisponde al valore desiderato. Questo implica il controllo di tutta la tabella, operazione che si rivela non solo lenta, ma soprattutto inefficiente nel caso in cui la tabella sia di grosse dimensioni e contenga solo poche registrazioni che soddisfino i criteri di ricerca.
L'aggiunta dell'indice sulla colonna di interesse cambia considerevolmente l'approccio della ricerca da parte del motore SQL.
L'indice contiene un valore per ogni riga della tabella presa in esame, ma queste risultano ordinate secondo i valori della colonna selezionata.
A questo punto, invece di scorrere tutta la tabella, riga dopo riga, alla ricerca delle registrazioni che corrispondono al valore ricercato, lo strumento di supporto MySQL si può avvalere dell'indice.
Se si volesse cercare, come caso di studio, tutte le righe corrispondenti all'azienda numero 13, la ricerca, sfruttando l'indice sul campo di interesse, trova tre righe corrispondenti, quindi nella riga successiva incontra la registrazione che contiene il valore 14, un numero più alto di quello che era stato indicato nei criteri di ricerca.

È comprensibile adesso il motivo per cui la ricerca può terminare, ed il fattore di efficienza in termini di prestazioni ottenuto.
Un altro fattore di efficienza è dovuto agli algoritmi di posizionamento, decisamente meno banali rispetto al metodo di ricerca lineare.
I dettagli secondo i quali MySQL indicizza le tabelle variano a seconda dei diversi tipi di tabella. Nelle tabelle MyISAM o ISAM, per esempio, le righe di dati di una tabella sono tenute in un file di dati, mentre i valori degli indici sono conservati in un file indice. Se esiste più di un indice per tabella, vengono raccolti tutti in un unico file di indice: ogni indice nel file di indice consiste in un array ordinato di valori chiave che vengono utilizzate per accedere velocemente al file di dati. Per contro, i gestori delle tabelle BDB e InnoDB non usano lo stesso criterio per separare le righe di dati e valori, sebbene entrambi gestiscano gli indici come insieme di valori ordinati: il gestore BDB usa un singolo file per la tabella per registrare sia i file che i valori degli indici, mentre InnoDB usa un solo spazio tabella nel quale conserva sia i dati che gli indici di tutte le tabelle gestite.

Le precedenti considerazioni descrivono i benefici derivanti dall'uso di un indice nel contesto di interrogazioni singole.
Tuttavia gli indici hanno un forte impatto prestazionale in occasione di query che coinvolgono più tabelle. Per esempio, in una query su singola tabella il numero di valori da esaminare per ogni colonna corrisponde al numero di record della tabella, mentre per una query su più tabelle il numero delle possibili combinazioni cresce a dismisura (anche di ordini di grandezza) poiché è il prodotto delle n righe di ciascuna tabella.
Si immagina di avere tre tabelle non indicizzate `t1`, `t2`, `t3` ognuna contenente rispettivamente una colonna `c1`, `c2`, `c3` formate da mille righe ciascuna che contengono i numeri da uno a cento.

Una query che trovi tutte le combinazioni fra le righe della tabella in cui i valori risultano uguali sarà del tipo:

```
SELECT t1.c1, t2.c2, t3.c3
FROM t1, t2, t3
WHERE t1.c1=t2.c2 AND t1.c1=t3.c3
```
(collegamento di più tabelle in forma esplicita)

Il risultato è di mille righe ognuna contenente tre valori uguali. Se si elabora la query senza aver configurato un indice, è impossibile dedurre quale colonna contenga quali valori, e di conseguenza bisognerebbe procedere per tentativi, provando tutte le combinazioni per trovare quelle che soddisfano la clausola `WHERE`: il numero delle combinazioni sarebbero in questo caso 1000 x 1000 x 1000 = 1000000000.

Svantaggi nell'utilizzo degli indici

L'ottimizzazione delle query con il supporto degli indici deve comunque essere eseguita con prudenza, ne derivano infatti svantaggi dall'uso non oculato degli stessi.

Un indice occupa spazio su disco e banalmente più indici occupano uno spazio maggiore, questo può contribuire a far raggiungere più velocemente il limite di dimensioni per la tabella:

- Indicizzare pesantemente una tabella ISAM o MyISAM può portare il file di indice a raggiungere la sua dimensione limite prima del file di dati.
- Aggiungere indici a tabelle BDB, che registrano dati e indici insieme nello stesso file, può portare la tabella alla sua dimensione massima su file molto più velocemente.
- Tutte le tabelle InnoDB condividono uno spazio tabella comune e aggiungendo degli indici c'è il rischio di esaurire lo

spazio tabella a disposizione di gran lunga più velocemente. Tuttavia si potrebbe ovviare a tal problema espandendo lo spazio tabella a disposizione aggiungendo componenti hardware.

Ogni indice aggiunto rallenta le operazioni di scrittura, ed è anche possibile che MySQL non riesca a scegliere l'indice migliore qualora ce ne fossero troppi, introducendo ulteriore latenza.
Prendere in considerazione quali tipi di confronto si effettueranno su un campo, agevola l'utilizzo ottimizzato di indici. In generale gli indici sono utilizzati per le operazioni `<, <=, =, >=, > e BETWEEN` ma sono utilizzati anche con `LIKE` quando il modello di ricerca ha un prefisso a caratteri.
Se invece si richiama una colonna solo per altri tipi di operazioni tipo `STRCMP()`, non si otterrà alcun beneficio dall'utilizzo degli indici. Come caso particolare, nelle tabelle `HEAP` gli indici sono in `hash` e utilizzati solo per le uguaglianze. Ma ciò non è stato influente nell'analisi svolta sulla base dati di Affitto.it.
Un' altra fase coinvolta nel processo di ottimizzazione consiste nella rilevazione delle query "lente".Tale fase ha un impatto significativo sulla prestazione del sistema.
`Mysqldumpslow` permette di identificare le query con un grado di efficienza ridotto, potenziali candidate all'indicizzazione, o segnale di studio non ottimale delle stesse. Si deve necessariamente tener conto in fase di test che tali misurazioni sono effettuate in tempo reale e che quindi nel registro delle query "lente" appariranno più query del previsto se il server è in una situazione di overload. Lo studio delle query progettate per l'applicazione web in esame sul server affitto.it è avvenuto in condizioni di alti carichi di lavoro da parte del server, questo per verificarne il funzionamento "nel caso peggiore".
Lo sviluppo della procedura di sincronizzazione e creazione di un istantanea è stato ideato seguendo un approccio che delega quanto più possibile le call function all' interprete di php, svuotando il core

SQL di tali responsabilità. Infatti, nonostante uno strato software che permette un elaborazione più complessa di stringhe per la conversione ed il confronto, SQL rimane un linguaggio altamente performante per operazioni su database e raramente utilizzato per altri scopi, nonostante ne fornisse un minimo supporto per questioni di completezza. Tale pratica, potrebbe inoltre rappresentare una significativa limitazione nella fase di configurazione degli indici e l'esempio successivo è un caso concreto di tale affermazione. In particolare si mostra come le *call function* impiegate nel costrutto `WHERE` ostacolano l'utilizzo dell'indice da parte dell'SQL processor, rendendolo di fatto inutilizzato.

```
WHERE (TO_DAYS(date_col) – TO_DAYS(CURDATE()) <
intervallo
```

È evidente che non si può assegnare un indice dato che la colonna deve essere interrogata per tutte le righe in modo da poterne computare il valore di `TO_DAYS(date_Col)`, e successivamente valutarne il confronto.
Di seguito è presentata una parte di codice sorgente dell'applicazione in esame in merito agli aspetti sopra discussi:
Si vuole mettere in evidenza come la parte di elaborazione sia stata interamente delegata al compilatore php lasciando al motore SQL il solo compito di produrre un set di record, già filtrato, ma in attesa di un ulteriore fase di selezione.
L'implementazione sfrutta API php le quali forniscono un interfaccia DBMS flessibile e sufficientemente astratta. La soluzione studiata coinvolge la libreria `mysqli` ed in particolare le funzioni:

- `Mysqli_fetch_array(mysqli_result resultset)`: Fornito come parametro di ingresso il set di record risultato della query, la funzione restituisce un array di stringhe che corrisponde alla riga letta o `NULL` se è già stata letta l'ultima riga disponibile. Si mantiene la relazione tra un nome del

campo della tabella e la chiave non numerica che identifica un campo dell'array.

- `Explode, preg_replace, echo, stristr` sono gli strumenti utilizzati per manipolare i dati. Il supporto fornito dalle `API` di php tramite tale libreria ha reso possibile e decisamente più performante tale implementazione, facendo ricadere la scelta su un approccio non orientato agli strumenti per la manipolazione di stringhe fornite da SQL.

```
while($array = mysqli_fetch_array($res2)) {

    if(prezzoAnnuncio_isValid($array)){//controllo prezzo valido.Saranno scartati gli annunci con prezzi non appartenenti al range di valori ammissibili secondo la tipologia di immobile
      reset($nome_quartiere_frazione);//ripristino degli indici presenti sull'array $nome_quartiere_frazione
      $nome_quartiere_frazione=explode("/",
                               preg_replace('/\s+/',
                               '',
                               $array['nome_quartiere_
                               frazione'])); //elimino gli spazi e trasformo in array di n elementi la stringa(presente nel campo "nome_quartiere_frazione" tabella "quartieri_frazioni") che contiene n quartieri separati da /


$zona_no_space=preg_replace('/\s+/','',$array['zona']
);//elimino gli spazi dalla stringa contenuta nel
campo "zona" tabella "descrizione_annunci".



while($current_nome_quartiere_frazione=
      each($nome_quartiere_frazione)){ //leggo uno per
volta tutti gli elementi di //$nome_quartiere_frazione
```

```
        if(stristr($zona_no_space,
           $current_nome_quartiere_frazione['value'])
           !=FALSE){//se il quartiere estratto è contenuto nella
        //descrizione della zona inserisci il record
            $id_quartiere_frazione=$array['id_
                                  quartiere_frazione'];
            $array['prezzo']=$array['prezzo'];
            $a = mysqli_query($con, "INSERT INTO
                 affitto_temp(id_comune, mq, prezzo,
                 id_tipo_immobile,
                 id_quartiere_frazione,
                 codice_tipo_annuncio,
                 mese_anno_corrente)
                 VALUES ('$array[id_comune]',
                 '$array[mq]',
                 '$array[prezzo]',
                 '$array[id_tipologia]',
                 '$id_quartiere_frazione',
                 '$array[codice_tipo_annuncio]',
                 '$mese_anno_corrente')");
            break;
    }
}
```

5.3 Test e valutazione delle performance

Si è fatto riferimento alla funzione `BENCHMARK` per testare l'utilizzo degli indici.
Tale funzione permette di sfruttare la console di MySQL per misurare quanto tempo impiega una query per essere eseguita. La sintassi è la seguente (Butcher, 2003):

```
BENCHMARK(conteggio, espressione)
```

In essa, espressione è una query SQL e conteggio è il numero di volte le quali dovrebbe essere eseguita.
Un esempio di impiego di tale funzione è:

`SELECT BENCHMARK(1, "[query]")`. Bisogna però tener conto del fatto che `BENCHMARK` misura il tempo trascorso e non il tempo macchina effettivo impiegato dal microprocessore. Per tal motivo queste misurazioni sono state effettuate su un ambiente con la medesima configurazione per le varie misurazioni e con pochi processi in esecuzione.

Impiego di risorse

E' facile ipotizzare che con il termine "prestazione" ci si riferisca alla velocità di elaborazione. Anche se questo è vero in molti casi, è necessario tenere conto del fatto che l'impiego dello spazio su disco o nella memoria, in alcuni casi è una questione prioritaria, tale da rientrare nel concetto di "prestazione".

Per ciò che concerne i tempi di esecuzione relativi al sistema in esame, la query già presentata come esempio nella discussione precedente è quella che maggiormente richiede impiego di risorse; visionando la base dati di affitto.it questo è di facile comprensione se si considera il coinvolgimento nella relazione di tabelle con quasi un milione di record, come la tabella annunci per esempio.

Nel caso specifico l'utilizzo di `JOIN`- tra le operazioni più esigenti in termini di quantità di memoria - il numero di tabelle collegate (4) e la dimensione delle stesse giustifica la gran quantità di risorse di cui necessita.

Lo sviluppo della query senza le fasi di ottimizzazione sopra descritte ha prodotto come risultato una interrogazione decisamente onerosa per le configurazioni di hardware disponibili; 40 minuti di elaborazione e 2 GB di memoria richiesta sul server di affitto.it, mentre il blocco dello script e l'uscita dall'applicazione si sono verificati su un comune computer con server in locale, la quale configurazione non permette di dedicare una tale quantità di risorse ad un singolo processo.

5.4 Sviluppo back-end

Presentazione del problema

Si vogliono memorizzare dati principali riguardo annunci di immobili nel mercato immobiliare italiano.
Nello specifico ogni record, sintetizzando dati relativi agli annunci, mostra il numero degli immobili di una determinata tipologia che sono situati nel quartiere/frazione di un determinato comune.
L'applicativo deve sincronizzarsi mensilmente col server di Affitto.it e creare un' istantanea del quadro immobiliare.
Prima di passare alla descrizione di come risolvere il problema a livello di implementazione -quindi più nel dettaglio rispetto alle valutazioni architetturali e organizzative viste in precedenza- è necessario presentare un vincolo di implementazione, probabilmente conseguenza di scelte non ottimali in fase di progettazione del sistema già esistente.
Tali carenze dell'applicazione portano spesso lo sviluppatore a studiare soluzioni implementative con gradi di complessità più elevati, oltre ad un maggiore dispendio di risorse in termini di tempo.

Come già accennato il problema nasce da un errata gestione della base dati:

- Se id_quartiere_frazione è determinato e diverso da 0 significa che sicuramente esiste un quartiere/frazione per quell'immobile;
- Se id_quartiere_frazione = 0 significa che:
  1. se il comune non compare nella tabella `quartieri_frazioni` allora l'immobile considerato non può essere collocato in un quartiere/frazione,

poiché non esiste suddivisione in quartieri/frazione per il comune dove risiede l'immobile considerato;

2. se il comune dell'immobile considerato compare nella tabella `quartieri_frazioni` allora l'inserimento dei dati su quell'immobile è stato incongruente e l'informazione sul quartiere/frazione di residenza dell'immobile può e deve essere cercato con altri metodi, siccome quest'ultimi costituiscono una parte numericamente significativa dell'intera base dati influenzandone considerevolmente l'attendibilità dei dati presentati all'utente.

Soluzione del problema

Fase 1

Di seguito si presentano i modelli delle tabelle coinvolte nella realizzazione del sistema. In Figura 5-4, 5-5, 5-6, 5-7 si mostrano rispettivamente i modelli di `Annunci`, `tipologie_immobili`, `descrizione annunci` e `quartieri_frazioni`.

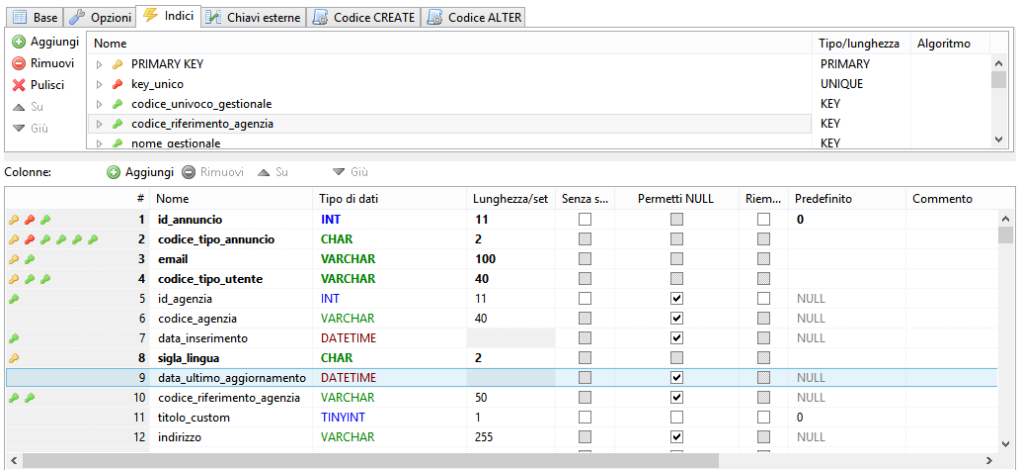

Figura 5-4: Modello e indici relativi alla tabella *annunci*

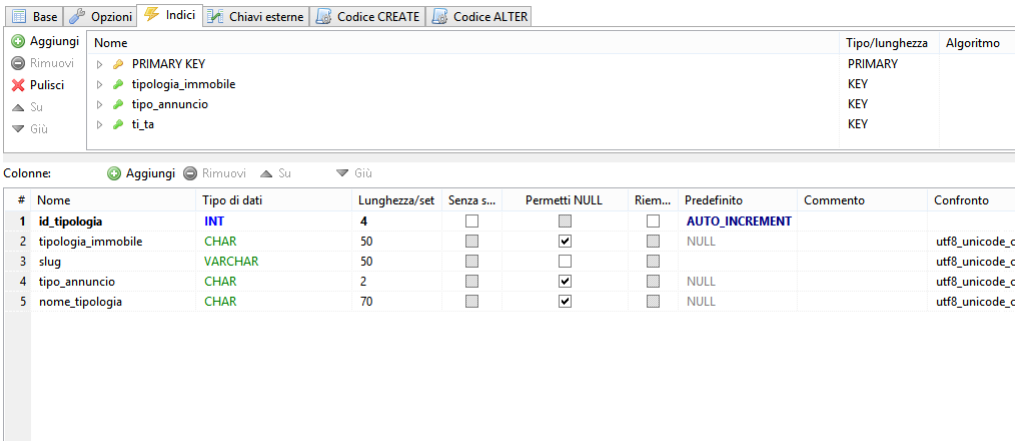

Figura 5-5: Modello e indici relativi alla tabella tipologie_*immobili*

L'attività di sviluppo nella fase 1 prevede:

1. query per la selezione del prezzo medio, mq ed altri parametri significativi degli annunci con comuni che non compaiono nella tabella `quartieri_frazioni`. I comuni in questione non possiedono quartieri/frazioni, quindi il relativo `id_quartiere_frazione` memorizzato in `affitto_archivio` sarà 0. Il campo `id_tipologia` riassume le informazioni `tipo_immobile` e `codice_tipo_annuncio`. entrambi campi presenti in `annunci`.
2. `Insert into` del risultato prodotto dalla query precedente nella tabella temporanea `affitto_temp` con normalizzazione; memorizzo l'`id` della tipologia e non più il nome della tipologia come in annunci (perchè non un identificativo già in annunci? Non è stato possibile modificare la struttura di annunci da direttive aziendali). In quel caso la soluzione ottimale sarebbe stata un identificativo per la tipologia in `annunci` senza il campo `nome_tipologia` (`codice_tipo_annuncio` deve per forza essere presente poiché componente della chiave primaria di `annunci`).

```
//prelevo data corrente
$mese_anno_corrente = date("Y-m-d");


//query selezione prezzo mq etc.. degli annunci con comuni che
//NON COMPAIONO nella tab "quartieri_frazioni" (i comuni in
//questione non possiedono quartieri/frazioni).Il loro
//id_quartiere_frazione sarà 0
//La tipologia è formata dalle info "codice_tipo_annuncio" e
//"tipo_immobile", entrambi campi presenti in "annunci"
$res=mysqli_query($con, "SELECT A.id_comune, A.mq,
     A.prezzo, T.id_tipologia, A.codice_tipo_annuncio
     FROM annunci as A
     JOIN tipologie_immobili as T ON
     A.tipo_immobile=T.tipologia_immobile AND
     A.codice_tipo_annuncio=T.tipo_annuncio
     WHERE A.id_comune NOT IN (SELECT id_comune FROM
     quartieri_frazioni)  AND A.id_comune <> 0 AND
     A.tipo_immobile <> 'NULL' AND A.tipo_immobile <>
    '0' AND A.mq <> 'NULL' AND A.mq <> 0 AND A.prezzo
     <> 'NULL' AND A.prezzo <> 0 AND A.attivo=1 AND
     A.confermato=1 AND (A.codice_tipo_annuncio =
    'VI' OR A.codice_tipo_annuncio = 'VC' OR
     A.codice_tipo_annuncio = 'OI' OR
     A.codice_tipo_annuncio = 'OC ')");


//insert into della query precedente nella tabella temporanea
//affitto_temp con id_tipologia invece del
//nome tipologia (id_tipologia lo estraggo collegando le 2 tabelle
nella query precedente)
while($array = mysqli_fetch_array($res)) {
  if(prezzoAnnuncio_isValid($array)){    //controllo prezzo
     //valido.saranno scartati gli annunci con prezzi non
     //appartenenti al //range di valori ammissibili secondo la
     //tipologia di immobile
     $array['prezzo']=$array['prezzo'];
```

```
        $a = mysqli_query($con, "INSERT INTO
             affitto_temp(id_comune, mq, prezzo,
             id_tipo_immobile, id_quartiere_frazione,
             codice_tipo_annuncio, mese_anno_corrente)
             VALUES
             ('$array[id_comune]', '$array[mq]',
                  '$array[prezzo]',
                  '$array[id_tipologia]', '0',
                  '$array[codice_tipo_annuncio]',
                  '$mese_anno_corrente')");//insert into
        con id_quartiere_frazione 0(il comune
        selezionato non compare nella tabella
        "quartieri_frazioni" quindi non possiede
        quartieri/frazioni)
    }
}
```

Fase 2

In questa fase si coinvolgono le medesime strutture dati già presentate nello step precedente

1. SELEZIONE
   Query per la selezione degli annunci con comuni che compaiono in `quartieri_frazioni` (quindi che hanno registrati i quartieri e frazioni) e con identificativo del quartiere/frazione diverso da 0.

2. INSERIMENTO IN AFFITTO_TEMP
   Il set di record selezionato nella query precedente viene filtrato dai record di annunci che non sono attendibili secondo le specifiche sui prezzi formalizzate dal committente e inserito nella tabella temporanea di affitto. In questa fase i record presenti in `affitto_temp` hanno ancora un carattere esplicito, o meglio, i record selezionati non hanno subito alcuna modifica se non la selezione dei campi utili.

```
//query per la selezione degli annunci con comuni CHE
//COMPAIONO in quartieri_frazioni" (quindi che possiedono
//quartieri/frazioni) E con un determinato campo
//id_quartiere_frazione

$res1=mysqli_query($con, "SELECT A.id_comune, A.mq,
      A.prezzo, T.id_tipologia,
      A.id_quartiere_frazione, A.codice_tipo_annuncio
      FROM annunci as A
      JOIN tipologie_immobili as T ON
      A.tipo_immobile=T.tipologia_immobile AND
      A.codice_tipo_annuncio=T.tipo_annuncio
      WHERE A.id_quartiere_frazione <> 0
      AND id_quartiere_frazione <> 'NULL'
      AND A.id_comune <> 0
      AND A.tipo_immobile <> 'NULL'
      AND A.tipo_immobile <> '0'
      AND A.mq <> 'NULL' AND
      A.mq <> 0 AND A.prezzo <> 'NULL'
      AND A.prezzo <> 0 AND A.attivo=1
      AND A.confermato=1
      AND (A.codice_tipo_annuncio = 'VI'
      OR A.codice_tipo_annuncio = 'VC'
      OR A.codice_tipo_annuncio = 'OI'
      OR A.codice_tipo_annuncio = 'OC ')");


while($array = mysqli_fetch_array($res1)) {
    if(prezzoAnnuncio_isValid($array)){
      $a = mysqli_query($con, "INSERT INTO
           affitto_temp(id_comune, mq, prezzo,
           id_tipo_immobile, id_quartiere_frazione,
           codice_tipo_annuncio, mese_anno_corrente)
           VALUES ('$array[id_comune]', '$array[mq]',
          '$array[prezzo]', '$array[id_tipologia]',
          '$array[id_quartiere_frazione]',
          '$array[codice_tipo_annuncio]',
          '$mese_anno_corrente')");
```

```
    }
}
```

Fase 3

Le attività relative allo sviluppo nella fase 3 sono:

1. SELEZIONE

Query per la selezione degli annunci con comuni che compaiono in `quartieri_frazioni` (quindi che hanno registrati i quarteri/frazioni) ma il relativo campo `id_quartiere_frazione` non è specificato in `annunci`. Selezione quindi di quelli il cui nome del quartiere/frazione compare nella descrizione dell'immobile e precisamente nel campo `zona` in `descrizione_annunci`.

In `descrizione_annunci` sono presenti più record che identificano lo stesso annuncio, per distinguere lingue diverse. Si tratta quindi di una relazione 1aN tra `annunci` e `descrizione_annunci`. Un modello relazionale del database in questione avrebbe sicuramente fornito un supporto nella lettura di questi casi, riducendo i tempi di sviluppo. Nella query seleziono quindi solo quelli in lingua italiana per creare la corrispondenza univoca tra record di `annunci` e `descrizione_annunci`. In questa fase l'operazione principale è la ricerca del quartiere/frazione nella descrizione dell'annuncio che l'utente registra.

A rallentare i tempi di sviluppo contribuisce il fatto di avere più quartieri/frazioni identificati da un unico `id`, separati da carattere speciale.

2 INSERIMENTO.

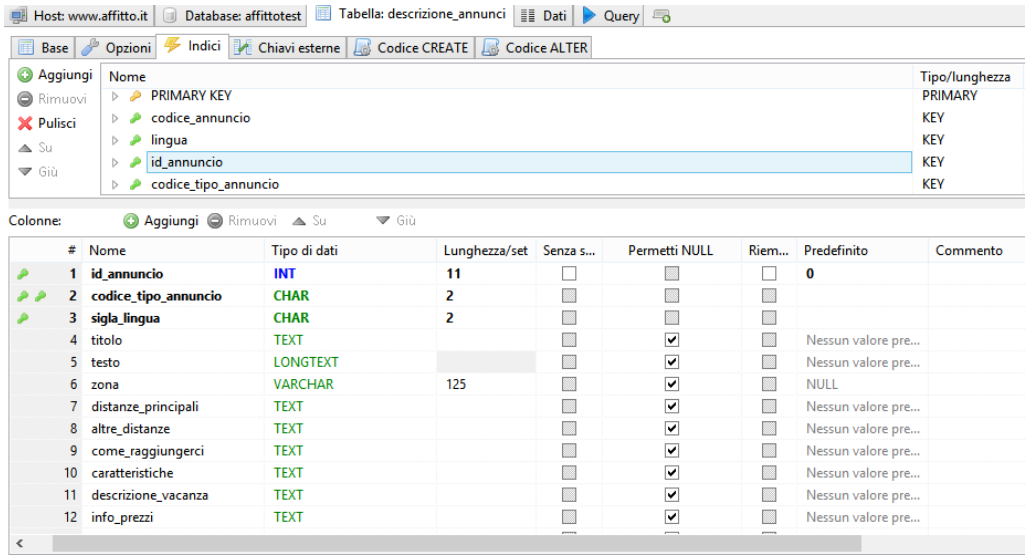

Figura 5-6: Modello e indici relativi alla tabella descrizione_annunci

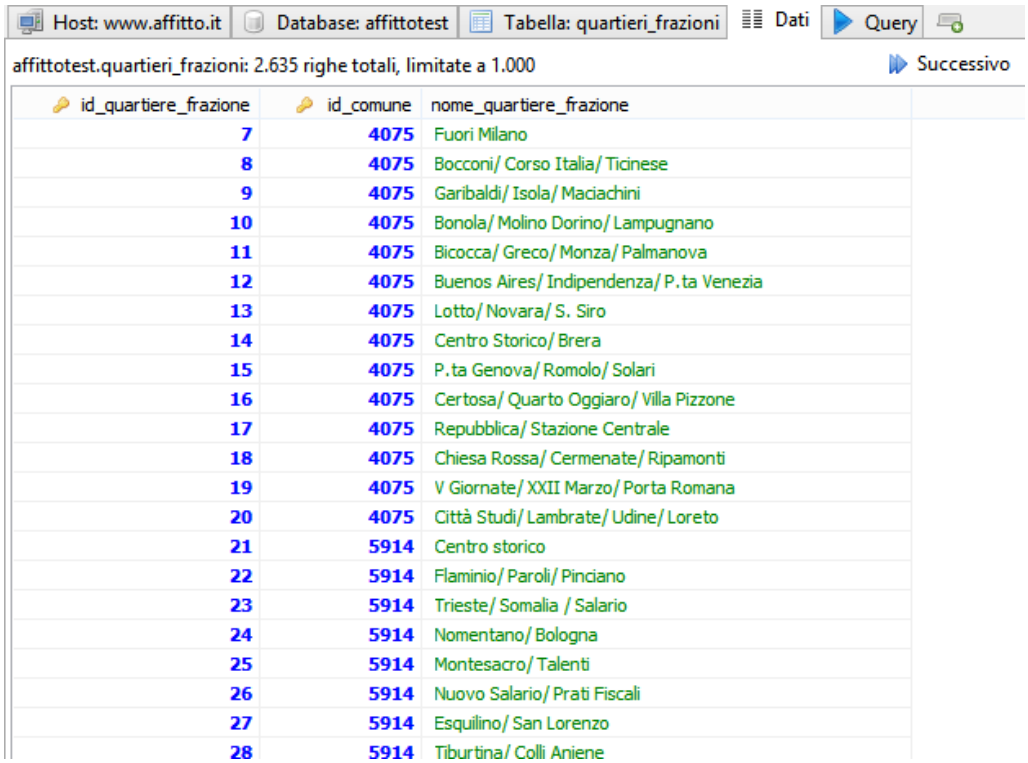

Figura 5-7: Esempio delle informazioni registrate nell'archivio quartieri_frazioni

Quindi:

//Query per la selezione degli annunci con comuni CHE
//COMPAIONO in "quartieri_frazioni" (quindi che possiedono

//quartieri/frazioni)ma il corrispondente campo
//"id_quartiere_frazione" non è specificato in "annunci".Seleziono
//quindi quelli il cui nome del quartiere_frazione compare
// nella descrizione dell'immobile, precisamente nel campo "zona" in
//"descrizione annunci".

//in "descrizione_annunci" sono presenti piu' record che identificano lo stesso annuncio ma con una lingua diversa, nella query
// specifico lingua italiana per creare una corrispondenza univoca tra i record di "annunci" e "descrizione_annunci" che saranno collegati

```
$res2=mysqli_query($con, "SELECT A.id_comune, A.mq,
      A.prezzo, T.id_tipologia,
      Q.id_quartiere_frazione,
      Q.nome_quartiere_frazione, D.zona,
      A.codice_tipo_annuncio
      FROM annunci as A
      JOIN quartieri_frazioni as Q ON
      A.id_comune=Q.id_comune
      JOIN descrizione_annunci as D ON
      A.id_annuncio=D.id_annuncio AND
      A.codice_tipo_annuncio=D.codice_tipo_annuncio
      JOIN tipologie_immobili as T ON
      A.tipo_immobile=T.tipologia_immobile AND
      A.codice_tipo_annuncio=T.tipo_annuncio
      WHERE A.id_quartiere_frazione=0 AND
      D.sigla_lingua='it' AND A.id_comune <> 0 AND
      A.tipo_immobile <> 'NULL' AND A.tipo_immobile
      <> '0' AND A.mq <> 'NULL' AND A.mq <> 0 AND
      A.prezzo <> 'NULL' AND A.prezzo <> 0 AND
      A.attivo=1 AND A.confermato=1 AND
     (A.codice_tipo_annuncio = 'VI' OR
      A.codice_tipo_annuncio = 'VC' OR
      A.codice_tipo_annuncio = 'OI' OR
      A.codice_tipo_annuncio = 'OC')");


while($array = mysqli_fetch_array($res2)) {
```

```
if(prezzoAnnuncio_isValid($array)){//controllo
prezzo valido.saranno scartati gli annunci con prezzi non
appartenenti al range di valori ammissibili secondo la
tipologia di immobile
    reset($nome_quartiere_frazione);//ripristino
    degli indici      presenti sull'array
    $nome_quartiere_frazione
    $nome_quartiere_frazione=
                          explode("/" ,
                          preg_replace('/\s+/',
                          '',$array['nome_
                quartiere_frazione']));//elimino
    gli spazi e trasformo in array di n elementi la
    stringa (presente nel campo "nome_quartiere_frazione"
    tabella "quartieri_frazioni") che contiene n quartieri
    separati da /

    $zona_no_space=preg_replace('/\s+/', '',
    $array['zona']);//elimino gli spazi dalla stringa
    contenuta nel campo "zona" tab "descrizione_annunci"

    while($current_nome_quartiere_frazione=each
    ($nome_quartiere_frazione)){//leggo uno per
    volta tutti gli elementi di $nome_quartiere_frazione
        if(stristr($zona_no_space,
           $current_nome_quartiere_frazione
           ['value'])!=FALSE){//se il quartiere
           estratto è contenuto nella descrizione della
           zona inserisci il record
             $id_quartiere_frazione=$array['id
             _quartiere_frazione'];
             $a = mysqli_query($con, "INSERT
             INTO affitto_temp(id_comune, mq,
             prezzo, id_tipo_immobile,
             id_quartiere_frazione,
```

```
                    codice_tipo_annuncio,
                    mese_anno_corrente)
                     VALUES

                    ('$array[id_comune]',
                    '$array[mq]', '$array[prezzo]',
                    '$array[id_tipologia]',
                    '$id_quartiere_frazione',
                    '$array[codice_tipo_annuncio]',
                    '$mese_anno_corrente')");
                break;
                }
            }
    }
}
```

Fase 4

L'ultima fase relativa alla risoluzione del problema di sincronizzazione col database di affitto e creazione di un'istantanea che rappresenti la sintesi del mercato immobiliare italiano mese per mese è lo step che porta alla forma definitiva della tabella `affitto_archivio`, la struttura dati utilizzata a tale scopo.
Concettualmente, le macro operazioni individuate nelle principali 4 fasi descritte possono essere rappresentate, seppur con un livello di dettaglio molto basso, da:

- Fase di selezione e prima organizzazione dei dati (per esempio la registrazione dati nella tabella, con l'introduzione di identificativi prima mancanti: `id_quartiere_frazione`, `id_tipologia_immobile`).
  In questa fase, i dati sono registrati nella tabella temporanea, che nello step successivo verrà utilizzata come contenitore dove i dati vengono elaborati e modellati, fino a giungere

nella forma definitiva ed essere portati in blocco nella tabella `affitto_archivio`;.

- Fase di Raggruppamento
  E' la fase accennata nello step precedente; l'operazione di rilievo che i dati subiscono in questa fase è il raggruppamento. A questo punto, sono definite le informazioni principali del quadro immobiliare relativo al mese in corso.
  Il set di record ottenuto viene agganciato in blocco ai dati già presenti in `affitto_archivio` che quindi, corrispondono alle sincronizzazioni avvenute nei mesi precedenti.

```
//ultimo step: group by degli annunci per comune, tipologia e
quartiere/frazione con relativo conteggio e media del prezzo e dei
mq.
//insert into affitto_archivio

$res3=mysqli_query($con, "INSERT INTO
      affitto_archivio (id_comune, media_mq,
      media_prezzo, id_tipo_immobile,
      id_quartiere_frazione, codice_tipo_annuncio,
      numero_immobili, mese_anno_corrente)
      SELECT AT.id_comune, AVG(AT.mq) as media_mq,
      AVG(AT.prezzo) as media_prezzo,
      AT.id_tipo_immobile, AT.id_quartiere_frazione,
      AT.codice_tipo_annuncio, COUNT(AT.id_comune) as
      numero_immobili, mese_anno_corrente
      FROM affitto_temp as AT
      GROUP BY AT.id_comune, AT.id_tipo_immobile,
      AT.id_quartiere_frazione");
```

Funzioni di supporto

In tutte le fasi di registrazione, relative ai primi 3 step descritti in precedenza è pianificato un ulteriore controllo che precede la convalida, e quindi la registrazione di tutti i record di `annunci`.
Tale controllo ha come scopo l'esclusione dai dati presentati di tutti gli immobili con valore economico o metratura non verosimili.
Nell'ottica della fase di calcolo delle medie per ogni classificazione di gruppo di immobili un valore distante dal range di valori ammissibili per il campo `prezzo`, può alterare considerevolmente il risultato finale presentato all'utente.
Anche in questo caso si evidenzia come le scelte implementative adottate hanno delegato tale operazione all'interprete di php e non alla gestione del DBMS. Questo, come già accennato, per motivi legati non solo alle performance, ma anche ai tempi di sviluppo alla leggibilità e all'organizzazione del codice sorgente.

```
function prezzoAnnuncio_isValid($array){
  //controllo prezzo valido.saranno scartati gli annunci con prezzi
  //non appartenenti al range di valori ammissibili secondo la
  //tipologia di immobile
  //paramentro di ingresso:array corrispondente al record-annuncio
  //considerato
  //ritorna true se il prezzo è ammissibile, false nel caso contrario
```

```
    if(($array['id_tipologia']==9 AND
      ($array['prezzo']>5 AND
       $array['prezzo']<10000)) OR
      ($array['id_tipologia']==7 AND
      ($array['prezzo']>1000 AND
       $array['prezzo']<200000)) OR
      ($array['id_tipologia']==79 AND
      ($array['prezzo']>50 AND
```

```
  $array['prezzo']<500000)) OR
  ($array['id_tipologia']==101 AND
  ($array['prezzo']>1000 AND
  $array['prezzo']<50000000)) OR
  (($array['id_tipologia']==105 OR
  $array['id_tipologia']==100) AND
  ($array['prezzo']>5000 AND
  $array['prezzo']<50000000)) OR
  ($array['id_tipologia']==74 AND
  ($array['prezzo']>50 AND
  $array['prezzo']<500000)) OR
  ($array['id_tipologia']==96 AND
  ($array['prezzo']>2000 AND
   $array['prezzo']<50000000)) OR
  ($array['id_tipologia']==42 AND
  ($array['prezzo']>300 AND
  $array['prezzo']<300000)) OR
  ($array['id_tipologia']==75 AND
  ($array['prezzo']>500 AND
  $array['prezzo']<5000000)) OR
  ($array['id_tipologia']==97 AND
  ($array['prezzo']>5000 AND
  $array['prezzo']<10000000))) {
     return true;
}

if ($array['id_tipologia']!=9 AND
   $array['id_tipologia']!=7 AND
   $array['id_tipologia']!=79 AND
   $array['id_tipologia']!=101 AND
   $array['id_tipologia']!=105 AND
   $array['id_tipologia']!=74 AND
   $array['id_tipologia']!=96 AND
   $array['id_tipologia']!=42 AND
$array['id_tipologia']!=75 AND
$array['id_tipologia']!=97) {
    if (($array['codice_tipo_annuncio']=='OI'
       OR $array['codice_tipo_annuncio']=='OC')
       AND $array['prezzo']>100 AND
       $array['prezzo']<50000){
```

```
                return true;
              }
            if (($array['codice_tipo_annuncio']=='VI'
               OR
               $array['codice_tipo_annuncio']=='VC')
               AND $array['prezzo']>10000 AND
               $array['prezzo']<100000000)    {
                  return true;
             }
     }
    return false;
}
```

Un altro controllo comune ai 3 step discussi in precedenza, questa volta attivo già con l'esecuzione della query, consiste nell'esclusione di tutti i record risultati da errori di inserimento dati, per esempio in fase di registrazione di un annuncio da parte dell'utente.
Si nota quindi, come le scelte relative allo sviluppo di tale sistema sono state fatte considerando la configurazione e le scelte progettuali della piattaforma pre-esistente.
Nel caso specifico, monitorare e vincolare l'inserimento di dati da parte dell'utente "a monte", uniformando tali dati all'idea di gestione ottimizzata del sistema, avrebbe di sicuro permesso uno sviluppo più agevole e decisamente più performante.
Tali errori di sviluppo si ripercuotono a cascata, e correggere eventuali bug nelle fasi più avanzate del sistema, risulta decisamente più oneroso dal punto di vista computazionale. E' evidente come lo sviluppo del sistema in esame possa aver risentito delle scelte giuste o sbagliate fatte in precedenza, relative alla piattaforma preesistente.

```
A.id_comune <> 0 AND
A.tipo_immobile <> 'NULL' AND
A.tipo_immobile <> '0'
```

```
AND A.mq <> 'NULL' AND
A.mq <> 0 AND
A.prezzo <> 'NULL' AND
A.prezzo <> 0
```

In Figura 5-8, 5-9 e 5-10 si riportano nuovamente le strutture e l'organizzazione delle informazioni risultanti dal processo di sviluppo; il risultato ottenuto coincide con gli obiettivi prefissati.

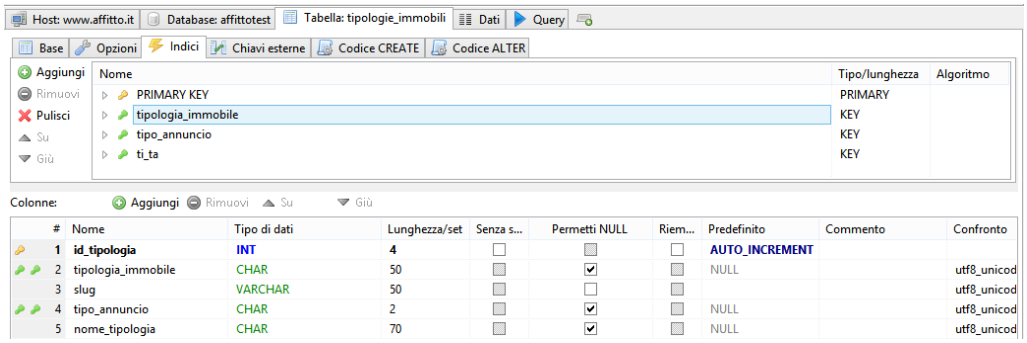

Figura 5-8: Risultato finale, organizzazione delle informazioni nell'archivio *tipologie_immobili*

id_comune	media_mq	media_prezzo	id_tipo_immobile	id_quartiere_frazione	codice_tipo_annuncio	numero_immobili	mese_anno_corrente
93	243	247923	31	0	VI	13	2015-05-15
93	100	400	32	0	OI	1	2015-05-15
93	82	495	57	0	OI	13	2015-05-15
93	130	360000	59	0	VI	1	2015-05-15
93	89	160872	60	0	VI	47	2015-05-15
93	127	200000	61	0	VC	1	2015-05-15
93	30	400	76	0	OC	1	2015-05-15
93	3425	1000000	94	0	VC	2	2015-05-15
93	200	210000	98	0	VC	1	2015-05-15
93	800	258000	103	0	VI	1	2015-05-15
93	484	272500	104	0	VI	2	2015-05-15
93	156	280000	106	0	VI	1	2015-05-15
4075	2039	11383	2	1	OC	12	2015-05-15
4075	870	10751	2	2	OC	17	2015-05-15
4075	350	27000	2	4	OC	1	2015-05-15
4075	1800	32000	2	5	OC	1	2015-05-15
4075	1264	5577	2	6	OC	5	2015-05-15
4075	1180	12500	2	8	OC	1	2015-05-15
4075	510	2500	2	9	OC	1	2015-05-15
4075	429	3430	2	11	OC	18	2015-05-15
4075	700	8000	2	12	OC	1	2015-05-15
4075	1523	10600	2	16	OC	3	2015-05-15
4075	478	3666	2	18	OC	5	2015-05-15
4075	70	800	2	19	OC	1	2015-05-15
4075	567	5019	2	20	OC	7	2015-05-15

Figura 5-9: Risultato finale, esempio di organizzazione dei dati nella tabella *affitto_archivio*

id_comune	media_mq	media_prezzo	id_tipo_immobile	id_quartiere_frazione	codice_tipo_annuncio	numero_immobili	mese_anno_corrente
4075	115	244143	5	1	VC	7	2015-05-15
4075	112	216900	5	2	VC	10	2015-05-15
4075	150	350000	5	3	VC	1	2015-05-15
4075	110	1085000	5	4	VC	2	2015-05-15
4075	130	204333	5	6	VC	15	2015-05-15
4075	108	398333	5	8	VC	9	2015-05-15
4075	205	269545	5	9	VC	11	2015-05-15
4075	75	157143	5	11	VC	7	2015-05-15
4075	76	326000	5	12	VC	10	2015-05-15
4075	90	130000	5	13	VC	1	2015-05-15
4075	234	1536667	5	14	VC	6	2015-05-15
4075	67	272000	5	15	VC	10	2015-05-15
4075	121	191625	5	16	VC	8	2015-05-15
4075	129	318000	5	17	VC	5	2015-05-15
4075	61	408000	5	18	VC	5	2015-05-15
4075	107	320000	5	19	VC	6	2015-05-15
4075	101	199444	5	20	VC	9	2015-05-15
4075	16	28375	7	1	VI	8	2015-05-15
4075	23	32556	7	2	VI	16	2015-05-15
4075	16	39917	7	3	VI	6	2015-05-15
4075	15	28500	7	4	VI	4	2015-05-15
4075	13	37000	7	5	VI	2	2015-05-15
4075	19	40353	7	6	VI	17	2015-05-15
4075	18	43714	7	8	VI	7	2015-05-15
4075	20	39140	7	9	VI	10	2015-05-15
4075	18	26333	7	10	VI	3	2015-05-15

Figura 5-10: Risultato finale, esempio di organizzazione dei dati nella tabella *affitto_archivio*

5.5 Sviluppo back-end: altri script lato server

Di seguito l'elenco degli altri script implementati, di sicuro non meno importanti ma dei quali ci si soffermerà a presentarne solo una breve descrizione:

```
Test_visual_prezzi.php
Visual.php
Test_tabella tipologie.php
Prezzo_medio_tipologia.php
Regolamento2.php
```

Test_visual_prezzi.php,test_tabella_tipologie.php

Nascono come script di test degli oggetti che contengono la logica relativa agli oggetti grafico andamento prezzi e tabella tipologie, si possono quindi vedere come blocchi intermediari che si pongono tra l'interfaccia utente ed i rispettivi componenti del sistema. Nello specifico, tale oggetto comunica con i parametri di selezione, elabora le informazioni acquisite e richiama, fornendo tali

informazioni, i servizi offerti dagli oggetti `grafico andamento prezzi` e `tabella tipologie`, implementati rispettivamente da `visual.php` e `prezzo_medio_tipologia.php`.

Visual.php, prezzo_medio_tipologia.php
Script che implementa funzioni statiche per la presentazione dei dati sotto forma di curva che rappresenta l'andamento temporale dei prezzi relativi al mercato immobiliare italiano.
I servizi offerti sono:

- `visual_andamento_prezzi`
- `visual_prezzimq`
- `visual_prezzi`
- `visual_prezzimq_tipologia_imm`
- `visual_prezzi_tipologia_imm`
- `visual_prezzimq_quartiere_frazione`
- `visual_prezzi_quartiere_frazione`
- `visual_prezzimq_comune_frazione`
- `visual_prezzi_comune_frazione`
- `print_graph`
- `get_output_format`

Le funzioni statiche dalla 2 alla 9 sono funzioni di supporto all'esecuzione di `visual_andamento_prezzi`. Tali funzioni gestiscono l'interazione con il database per estrarre i dati richiesti in visual_andamento_prezzi e renderli utilizzabili per la funzione `print_graph`.
`Print_graph` genera il grafico a partire dal pacchetto dati ricevuto e lo rende disponibile alla gestione lato `front-end`, per essere visualizzato.
`Get_ouput_format` è utilizzata da tutte le funzioni dalla 2 alla 9 ed in particolare nella fase di formattazione dei dati. Nello specifico converte il formato data americano con il quale è rappresentata la

data di creazione di un' istantanea, al formato europeo, e gestisce tale formato, apportandone modifiche in relazione all'ambito di utilizzo.

5.6 Sviluppo front-end

Regolamento2.php, style.css

Di seguito si riportano le funzionalità implementate:

- i valori di default devono essere corrispondenti all'ultimo valore selezionato;
- parametri di selezione(`selectbox`) di default che dipendono dalle condizioni iniziali di accesso alla pagina (qual'era la ricerca che si stava effettuando su altre sezioni del portale www.affitto.it?);
- Ogni `selectbox`, deve poter interagire con l'esterno in modo tale da offrire un range di valori che dipenda dalla ricerca che si stava effettuando in un'altra sezione del sito (per esempio, se il valore selezionato relativo al parametro `città` è Milano, allora il parametro `quartiere/frazione` deve potersi aggiornare ed offrire come range di valori tutti i quartieri/frazioni relativi).

6 Conclusioni

A conclusione del percorso che portato all'integrazione Web sul portale immobiliare www.affitto.it, è opportuno trarne delle considerazioni.
I goal raggiunti, coincidono con quelli attesi relativamente alle specifiche, ma non relativamente alle risorse temporali, infatti, ulteriori valutazioni fatte sul modello del database hanno riportato alcune criticità che hanno condizionato significativamente la fase di implementazione, allungandone i tempi di sviluppo.
Come sarebbe cambiato allora il sistema, disponendo di un periodo di tempo più esteso? Tra le varie ipotesi si considera l'integrazione di JAVASCRIPT per incrementare l'indice di gradimento lato utente, puntando a migliorarne le caratteristiche di interazione e interfacciamento coi servizi offerti dal sistema. In particolare, si pianifica la rivisitazione del modello grafico e tabellare per una maggiore dinamicità degli stessi.
Sempre ipotizzando di disporre di una maggiore quantità di risorse temporali, avrei preso in considerazione l'integrazione di AJAX e JSON, rispettivamente una tecnica e un formato per lo scambio di dati in un'architettura client-server, finalizzati ad ottimizzare l'esperienza di utilizzo dei servizi forniti dal portale.

Bibliografia

Butcher, Antony. *MySQL 4.* 2003.

Du Bois, Paul. *MySQL.* 2004.

http://dev.mysql.com. [Online] [Riportato: 2014 Luglio 2.]

http://phpdevtutes.blogspot.it. [Online] [Riportato: 2 Luglio 2014.]

http://www.hosting.vt.edu. [Online] [Riportato: 2 Luglio 2014.]

http://www.html.it. [Online] [Riportato: 2 Luglio 2014.]

http://www.interact.it. [Online] [Riportato: 1 Luglio 2014.]

http://www.laboratoriocss.it. [Online] [Riportato: 2 Luglio 2014.]

http://www.miamammausalinux.org. [Online] [Riportato: 1 Luglio 2014.]

http://www.openskill.info. [Online] [Riportato: 1 Luglio 2014.]

http://www.php.net. [Online] [Riportato: 2 Luglio 2014.]

http://www.tomstardust.com. [Online] [Riportato: 1 Luglio 2014.]

http://www.w3.org. [Online] [Riportato: 1 Luglio 2014.]

http://www.w3c.it. [Online] [Riportato: 1 Luglio 2014.]

https://forum.congnghethongtin.org. [Online] [Riportato: 2 Luglio 2014.]

Molesini, Ambra e Natali, Antonio. *La costruzione dei sistemi software: dai modelli al codice.* 2008.

Oppel, Andrew. *SQL Tecniche e soluzioni.* 2006.

Reese, George. *MySQL.* 2008.

Weeling, Luke e Thomson, Laura. *MySQL Tutorial.* 2004.

Indice delle figure