Politecnico di Milano
Facoltà di Ingegneria
Corso di Laurea Specialistica in Ingegneria Informatica

CoolTraveling

Scalabilità di un'idea ed evoluzione dei servizi

Relatore: Prof.re Stefano Bruna

Raffaele BISOGNO matr. 740036
Marco CAVONE matr. 724791

Anno Accademico 2011/2012

Indice

Abstract

In questo elaborato verrà descritta la nascita del progetto CoolTraveling, dalla sua concezione sino alla sua traduzione in idea imprenditoriale. CoolTraveling è un nuovo modo di concepire la pianificazione dei viaggi via mobile: il primo servizio che permette in pochi semplici click di confrontare soluzioni di viaggio pianificate con mezzi di trasporto differenti assistendo i clienti con informazioni di valore come il meteo, le accommodation e i ritardi in realtime sino al perfezionamento dell'acquisto. Nel documento sarà dato ampio spazio alla progettazione di un'architettura a servizi scalabile nel cloud e le modalità di integrazione dei servizi di Viaggiatreno, Bad&Breakfast e Meteo. A corollario dell'elaborato saranno presentate le storyboard, storie d'uso dell'applicazione.

Introduzione

Negli ultimi anni si è assistito ad una rivoluzione del mercato del mobile che ha vissuto una vera e propria metamorfosi alimentata sia dalla sempre maggiore diffusione di telefonini di ultima generazione sia dall'ascesa dei nuovi tablet che consentono una fruizione completa, semplice e gratificante in termini di contenuti ad alto valore aggiunto oltre ad una totale interattività mai raggiunta prima.

La possibilità di essere "always on" (constantemente connessi) e "anywhere" (ovunque noi siamo) sono le vere "killer application" dei mobile phone che rappresentano per definizione il concetto di prossimità tra le persone, i prodotti, i servizi e le aziende.

Oggi, quando si parla di trend di mercato e del web, si pensa subito all'acronimo SoLoMo, ovvero:

- **Social**
- **Local**
- **Mobile**

È sicuramente importante capire il valore di avere una presenza anche nei social media, che sia al tempo stesso sia geolocalizzata e fruibile da dispositivi mobile e che dia valore a ognuno dei contesti di fruizione dei contenuti da parte di un utente (vedi ad esempio un'offerta riservata agli iscritti ad una pagina Facebook aziendale e una geolocalizzata riservata all'utente Foursquare e alla sua "presenza" nel punto vendita sul territorio).

Sin dall'inizio la nostra attenzione si è rivolta all'utilizzo del mobile come compagno di viaggio, sempre presente a "consigliarci" dovunque noi ci trovassimo. Le statistiche, infatti, non fanno altro che darci ragione. Una tale tecnologia aiuta soprattutto i turisti che hanno raggiunto la loro destinazione: le potenzialità in questo ambito sono enormi e lasciano ampi spazi di manovra.

Ma la domanda importante da porsi è quanto il mobile ha cambiato il modo in cui noi viaggiamo. Circa 1/3 degli utenti social media hanno usato una applicazione mobile per cercare ottimi prezzi per voli/hotel e il 15% hanno scaricato un'app specifica per un determinato viaggio, secondo la survey realizzata da Lab42[1].

L'immagine in figura 1 sintetizza con i numeri quanto detto precedentemente.

Nel nostro piccolo abbiamo cercato di interpretare parte di questa esigenza attraverso un'applicazione mobile che proponesse un modo semplice e immediato per ricercare B&B, per visualizzare le condizioni meteorologiche e soprattutto per confrontare diverse soluzioni di viaggio con mezzi di trasporto differenti.

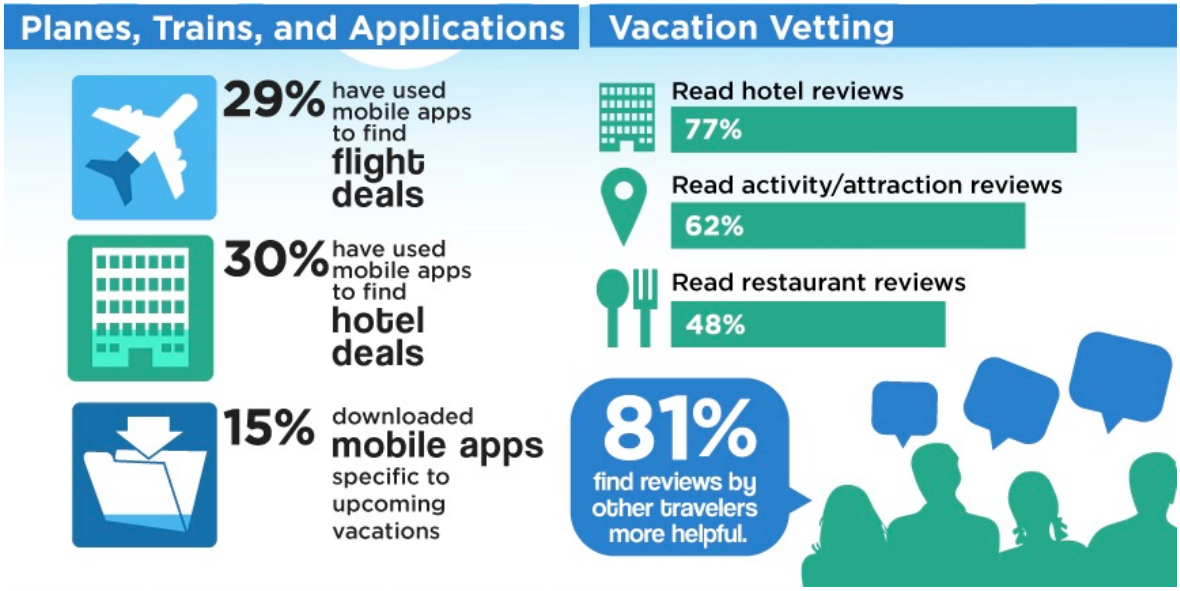


Figura 1: Comportamento degli utenti mobile in vacanza

Capitolo 1

CoolTraveling - Il Progetto

CoolTraveling ti permette di organizzare i tuoi viaggi in modo rapido semplice ed efficiente direttamente dal tuo cellulare. Con CoolTraveling puoi:

- Cercare la stazione o l'aeroporto più vicini alla città in cui ci si trova o nella quale ci si deve recare in tutta Italia
- Cercare quali treni utilizzare per raggiungere la meta desiderata
- Cercare quali aerei utilizzare per raggiungere la meta desiderata
- Confrontare tra loro i viaggi in treno con quelli in aereo in base a:
    - durata del viaggio
    - costo del viaggio
    - primo a partire (data / ora di partenza)
- Verificare le condizioni meteo sia della località di partenza che della località di arrivo
- Scoprire B&B e accommodation vantaggiose in ogni città italiana

Con CoolTraveling tutto questo è possibile. Un unico programma in grado di determinare in modo veloce ed intelligente la stazione o l'aeroporto a noi più vicino, proponendo un rapido e puntuale confronto tra le soluzioni di viaggio (treno/aereo) identificate. Permettendo inoltre la consultazione del meteo e delle accommodation è

in grado di svolgere in un solo istante il lavoro di una persona con almeno 6 pagine internet aperte, tramutando la ricerca dei viaggi con il cellulare un esperienza semplice ed efficace.

Figura 1.1: CoolTraveling: AS IS - TO BE

1.1 Obiettivo

CoolTraveling esprime una esigenza che in primo luogo era manifestata dagli stessi sviluppatori dell'idea: il suo scopo è quello di aiutare gli utenti nella pianificazione di un viaggio dovunque essi si trovino, fornendo un supporto altamente pervasivo. Nella sua completezza, il progetto è costituito da una corposa parte dati, che può essere divisa in due macro-sezioni:

- *dati utente*: necessari per fornire informazioni personalizzate e per avere un alto grado di profilazione utente; quindi il sistema dispone di un processo di autenticazione.

- *dati di ricerca*: dati delle ultime ricerche effettuate dagli utenti, necessari per ottimizzazioni su tempi di risposta e livelli di saturazione del server.

Tali dati (integrati opportunamente con quelli ricavati da fonti esterne eterogenee), consentono alla parte Server di inviare celermente ad una parte Client (mobile app), tramite protocolli standard di comunicazione, una risposta il più possibile idonea e attinente alle esigenze degli utenti.

1.2 Le 5 W

Cos'è CoolTraveling (WHAT) Applicazione All-In-One per dispositivi mobili che permette di ottenere, in modo pratico e veloce, una vasta gamma di informazioni per la pianificazione/gestione di un viaggio (treni, aerei, accommodation, meteo. Vedi immagine 1.2)

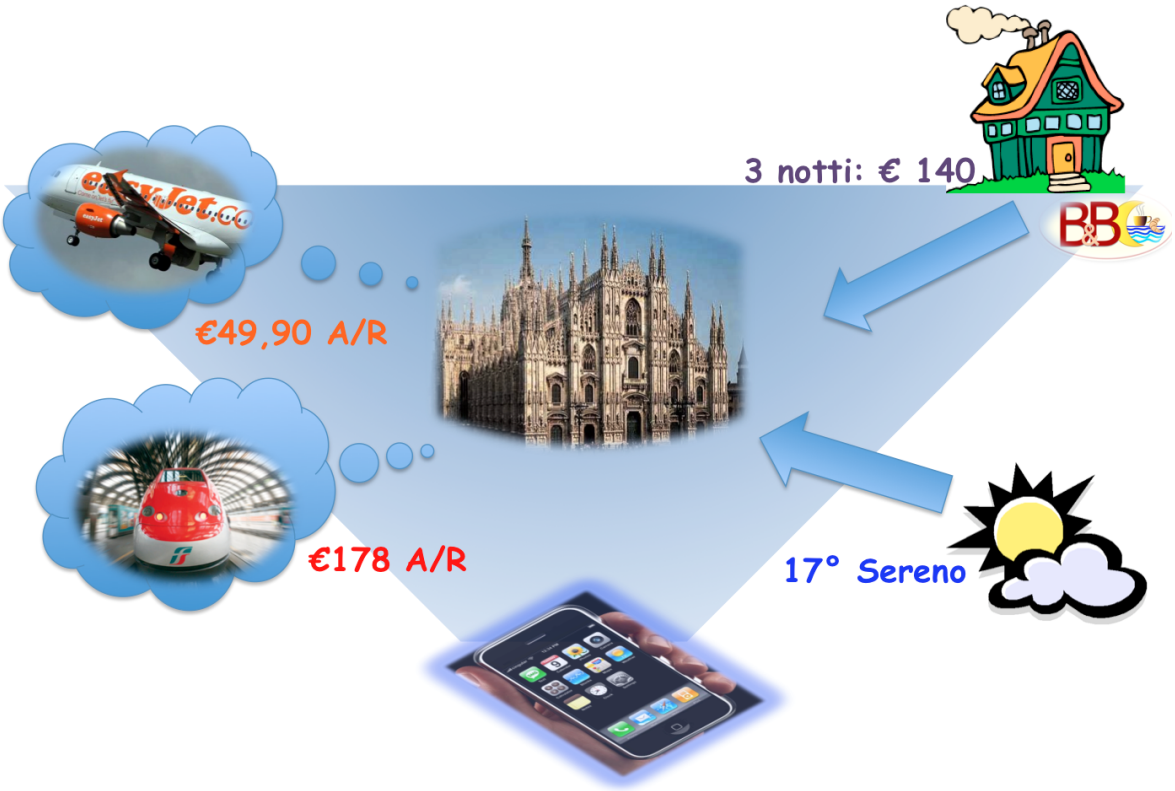


Figura 1.2: Esempio di utilizzo

A chi è rivolto CoolTraveling (WHO) A tutti gli utenti che possiedono uno smartphone e che hanno necessità di viaggiare costantemente.

Qual è lo scopo di CoolTraveling (WHY) Aggregare i dati estratti da più servizi al fine di fornire un'unica interfaccia, userfriendly, agli utenti che hanno la

necessità di viaggiare, evitando così l'utilizzo del browser sui molteplici mobile site (Trenitalia.com, Viaggiatreno.it, Expedia.com, Volagratis.com, bed-and-breakfast.it, AccuWeather.com). L'idea punta ad integrare un numero di servizi tale da soddisfare ogni necessità: gestire in modo autonomo la ricerca di un semplice viaggio in treno o la pianificazione di un week-end in una splendida città d'arte.

Dove opera CoolTraveling (WHERE) Il core di CoolTraveling (il server) è installato su un hosting cloud collegato ad internet. Gli utenti possono accedere ai diversi servizi offerti dal server tramite il loro mobile site o il widget[1] Vodafone 360;

Quando opera CoolTraveling (WHEN) CoolTraveling opera ogniqualvolta l'utente abbia la necessità di ricercare un treno piuttosto che un aereo, reperire informazioni relative a ritardi dei mezzi di trasporto, al meteo ed all'accommodation.

1.3 Com'è nato CoolTraveling

Un pò per passione, un pò per gioco ma soprattutto per la nostra voglia di cimentarci in nuove esperienze, nell'inverno del 2009, abbiamo ideato qualcosa che al mercato italiano ancora mancava. Ci era stato chiesto, al fine di poter partecipare a un concorso sponsorizzato da Vodafone, di presentare un'idea innovativa da implementare sulla *nascente* piattaforma Vodafone360.

L'anno accademico era appena iniziato e già ci trovavamo con un progetto extra sulle spalle! In pochi giorni delineammo le linee guida necessarie a rendere la nostra applicazione vincente. La scelta, forse non casualmente, prese forma un uggioso giorno di novembre nella caotica stazione di Cadorna, dove ci trovavamo a transitare al ritorno dalla sede del Politecnico di Milano Bovisa. In quella cornice di tipico caos milanese, aggravata a causa dell'ennesimo sciopero dei mezzi di trasporto, ci sembrò di grande utilità poter portare la tecnologia il più possibile vicino alle persone in viaggio. Tendenzialmente quando si pianifica una vacanza, piuttosto che un'uscita di lavoro si deve valutare il mezzo di trasporto con il quale muoversi, trovare una sistemazione e infine verificare le condizioni atmosferiche, per evitare brutte sorprese una volta giunti

[1] è un'applicazione mobile sviluppata per la nuova piattaforma Vodafone 360

Figura 1.3: Vodafone 360

a destinazione. Il nostro obiettivo, la nostra idea, risiede proprio in questo: **unificare i servizi legati alla mobilità rendendoli fruibili dalle persone anche nei contesti più impensabili, grazie alla pervasività che il mobile ha attualmente.**

1.3.1 La vittoria del WHYMCA

Nelle settimane successive, che ci separavano dal termine ultimo per la consegna dei lavori, decidemmo di integrare nella nostra applicazione due differenti mezzi di trasporto: treni e aerei, potendo così fornire ai nostri clienti un confronto puntuale sulle soluzioni di viaggio, trovando quella che più si addice alle loro esigenze. A seguire furono annesse anche le accommodation per tutte le città italiane e le previsioni meteorologiche. Con la vittoria del **WHYMCA**, oltre a tante soddisfazioni sono arrivati anche quattro biglietti per il **Mobile World Congress** di Barcelona offerti dallo sponsor Vodafone. Un'opportunità unica che ha consentito a quattro studenti di partecipare alla più importante conferenza mondiale sul mobile. Grazie a questa esperienza siamo tornati in Italia con un rinnovato entusiasmo, forti delle nuove idee e intuizioni scaturite dalla convention di Barcelona. Visti i tempi ristretti, meno di un mese, la prima release sviluppata era poco più di un mock-up che una reale implementazione

della nostra idea. Dopo una breve pausa, motivata dalla sessione d'esami in corso, abbiamo ripreso attivamente lo sviluppo del nostro neonato widget CoolTraveling perfezionando ulteriormente le ricerche dei viaggi. Come prima cosa abbiamo introdotto nuovi algoritmi, in grado di ottimizzare le richieste diminuendo così i tempi di attesa degli utenti, consentendo inoltre di trovare le stazioni e gli aeroporti più vicini ad ogni comune italiano.

1.3.2 La scalata verso la Vodafone App Start Competition

Alla premiazione del concorso **WHYMCA** tutti ci suggerirono di partecipare ad una nuova competizione italiana la **Vodafone App Star Competiton**, dove avremmo potuto confrontarci, non più con soli studenti, ma con tutto il panorama italiano degli sviluppatori per Vodafone360. Una nuova sfida, difficile ma avvincente. Per l'occasione abbiamo migliorato ulteriormente i servizi da noi offerti garantendo la possibilità di ricercare voli tramite le compagnie aeree di tutto il mondo, con il relativo supporto meteo. L'obiettivo dell'App Star Competition, indetta da Vodafone, è promuovere la sua nuovissima piattaforma Vodafone360, mediante la creazione di nuovi widget da parte degli sviluppatori di tutta Europa.
La competizione è stata divisa in due fasi: una prima nazionale, conclusasi nell'Aprile del 2010, dove i migliori tre widget per ogni nazione sono stati scelti da una commissione tecnica Vodafone e una europea alla quale sono stati ammessi solamente i widget classificatisi al primo posto nelle rispettive competizioni nazionali. In quest'ultima fase, iniziata nel mese di maggio, a far da giuria sono stati gli stessi utenti che hanno potuto esprimere il proprio voto, incoronando così la miglior applicazione tra le otto finaliste delle nazioni europee in gara.
CoolTraveling ha vinto il primo posto nella competizione nazionale aggiudicandosi 25000 euro di premio e l'onore di rappresentare l'Italia nella Vodafone App Star Competion con CoolTraveling. Questa volta la posta in gioco è stata ancora più alta, al primo classificato andranno, infatti, 75000 euro e al secondo 25000. A differenza della prima competition, regolamentata da giuria tecnica, in questo caso è stato il volere del pubblico europeo ad incoronare i tre migliori widget. Dopo un lunghissimo mese passato interamente a promuovere il nostro prodotto il 25 Maggio 2010 è arrivato il responso ufficiale da Vodafone UK: **Secondi Classificati**. CoolTraveling, l'unica App italiana in gara per l'App Star Competition, grazie al volere del pubblico, si è aggiudicata il

secondo posto e il premio complessivo di 50000 euro messo in palio da Vodafone.

1.4)

Figura 1.4: App Star Competition

1.3.3 Il comunicato ufficiale Vodafone

http://mobilewidgetdev.wordpress.com/2010/05/25/vodafone-app-star-overall-winners-announced/

1.4 Gli eventi

- Intervista su Neapolis, RAI3
- Intervista radio mercoledì 26 maggio su radio24 - NovaLab24
- Evento fAutori presso lo Spazio Teatro 89 a Milano

1.5 Dall'idea al progetto

Una delle cose spesso più difficili è tramutare un'ottima intuizione in un progetto funzionante. Quando abbiamo ideato CoolTraveling ci siamo resi conto sin da subito che gli attori in gioco sarebbero stati molteplici e l'architettura da progettare tutt'altro che banale.
Sostanzialmente il progetto può essere definito in letteratura come un aggregatore o mashup di servizi, ovvero un prodotto in grado di interrogare simultaneamente più interfacce e fornire il risultato in modo strutturato. Perché tutto questo potesse avvenire è stato necessario sin da subito scovare i vari servizi legati al turismo/mobilità dodati di interfacce pubbliche. Dopo una prima analisi il responso non è stato dei migliori: anche i servizi pubblici, che per legge dovrebbero aver sempre a disposizione interfacce libere per la consultazione dei dati disponibili sul web, non forniscono a chiunque gli accessi ai loro servizi. Se in un primo momento lo sconforto l'ha fatta da padrone, da buoni ingegneri in erba, abbiamo subito trovato una soluzione alternativa al problema: il crawling delle pagine web dei servizi, mentre in parallelo continuavamo la richiesta di servizi da interrogare per attingere alle informazioni. Dopo poche poche settimane grazie a qualche raccomandazione siamo riusciti ad ottenere l'accesso ai servizi di VolaGratis con i quali abbiamo subito coperto l'intera offerta aerei. Purtroppo VolaGratis risolveva solo uno dei nostri servizi e quindi abbiamo dovuto continuare la ricerca mentre in parallelo nasceva CoolTraveling con i servizi aerei di VolaGratis.

Figura 1.5: VolaGratis

Come tutti sappiamo il panorama italiano dei treni è sostanzialmente un monopolio di Trenitalia. Non riuscendo a creare un contatto con loro per poter avere accesso ai loro servizi siamo stati costretti a cimentarci nella strada più complessa ovvero la simulazione del comportamento degli utenti sulle pagine di Trenitalia con relativo crawling dei dati analizzati. Nonostante fossimo consci della complessità dell'obiettivo e della poca stabilità che avrebbe potuto dare ai servizio la ricerca dei dati effettuata

in questo modo in neanche una settimana siamo riusciti a simulare ed interpretare la maggior parte delle pagine di Trenitalia sino a recuperare il prezzo del biglietto e la disponibilità dei posti sul treno. Il processo è stato tutto tranne che semplice in quanto siamo stati costretti ad imparare ed assimilare le modalità di generazione e disposizione dei contenuti delle pagine web nel sito Trenitalia per poter così simulare in prima battuta le richieste e successivamente estrarne i dati.

Figura 1.6: Trenitalia

Successivamente con un po' di esperienza in più siamo riusciti a recuperare da Trenitalia anche le informazioni in realtime sui ritardi dei treni, potendo quindi fornire ai nostri utenti un servizio sempre migliore ed allineato con la realtà. Per integrare questo tipo di informazioni non è bastato analizzare il sito Trenitalia in quanto è il sevizio ViaggiaTreno ad occuparsi di questa tipologia di dati. Dopo aver capito il funzionamento e l'approccio adottato da Trenitalia per la gestione dei dati relativi alle tempistiche sulle varie tratte abbiamo prontamente integrato i ritardi di tutti i treni richiesti dagli utenti all'interno di CoolTraveling.

Figura 1.7: Viaggiatreno

CoolTraveling non è nato solamente con l'idea di unire i servizi di treni e aerei in Italia ma di semplificare tutto il processo decisionale delle persone che si stanno per mettere o sono già in viaggio. A nostro modo di vedere una delle più grandi difficoltà quando ci si mette in viaggio, oltre alla scelta del mezzo, è la scelta della stazione/areoporto più vicino, soprattutto quando non siamo vicini a casa. Forti di questa convinzione abbiamo progettato e sviluppato un algoritmo in grado di mappare logicamente tutti i comuni d'Italia e trovare per ciascuno di essi sia la stazione che

l'aeroporto più vicino. In questo modo i nostri utenti potranno inserire nei campi di ricerca il comune in cui si trovano e automaticamente il sistema si preoccuperà di recuperare per loro le informazioni delle stazioni e degli aeroporti più vicini,s dati con i quali verranno fatte le interrogazioni ai vari servizi.
Dopo aver correttamente configurato i servizi di VolaGratis e Trenitalia per essere fruibili da CoolTraveling e aver standardizzato le informazioni ricevute ci siamo focalizzati sui servizi accessori come le accommodation e il meteo, per avere un'offerta sempre più puntale e completa. Anche in questo caso, come nei precedenti, la ricerca di servizi pubblici non è stata facilissima ma almeno in un caso, il meteo, ci siamo riusciti.
Il servizio per il meteo prescelto è stato Accuweather, in quanto, come anticipato precedentemente, fornisce un WSDL pubblico gratuito per scoprire le previsioni meteo world wide. L'integrazione è stata abbastanza semplice e in meno di una giornata di lavoro ogni richiesta effettuata dal servizio poteva esser corredata dal meteo della località di partenza e quella di arrivo. Successivamente abbiamo ritenuto opportuno sfruttare al massimo le potenzialità del servizio iniziando a richiedere le previsioni del tempo non solo per il giorno esatto del viaggio ma anche per un intervallo di tempo di un paio di giorni, consentendo così ai nostri utenti di poter pianificare con più consapevolezza i propri viaggi.

Figura 1.8: Accueweather

Dopo aver completato l'integrazione di servizi legati agli Aerei, Treni e infine Meteo CoolTraveling inziava decisamente a prender forma e si stava trasformando sempre più in un servizio vero e proprio. Complesso ma allo stesso tempo ricco in tutte le sue sfaccettature. Come ben ricorderete dall'introduzione iniziale, oltre ai servizi appena citati mancano all'appello le accommodation ovvero il mondo degli Hotel, Case Vacanza, Appartamenti, Bed & Breakfast, ecc...
Come nostro solito ci siamo tuffati a capofitto in questo nuovo mondo iniziando a

scremare la concorrenza ed iniziare quindi la ricerca del fornitore del servizio ideale. Dopo un po' di ricerche ci è stato facile intuire che il mondo delle accommodation è tanto complesso, se non di più, di quello dei treni e degli aerei. In questo mondo infatti le variabili in gioco sono molte e le configurazioni a cui gli utenti possono essere interessati molteplici. Inoltre in questo settore è quasi impensabile trovare interfacce pubbliche a cui richiedere dati. Visto il nostro intento di fornire un servizio semplice ed immediato per le persone in mobilità e la difficoltà di ottenere servizi ad uso gratuito abbiamo dovuto trovare una soluzione alternativa. Per prima cosa ci siamo focalizzati sul mercato italiano, per ridurre la complessità e aumentare il livello di servizio, quindi con la stessa logica è stata privilegiata un'unica categoria di quelle in gioco. La categoria prescelta è stata Bed & Breakfast poiché sono economici, si trovano sia in città che in provincia e difficilmente sono già adeguatamente sponsorizzati sul web. Dopo aver effettuato la scelta ed aver fatto un rapido screening del mercato italiano abbiamo deciso di procedere in modo analogo a quanto fatto in precedenza per i comuni, ovvero mappando su database tutti i Bed & Breakfast italiani. Il giorno stesso abbiamo identificato il nostro *fornitore* dei dati e in qualche giorno di lavoro siamo riusciti a replicare in casa nostra il loro database di Bed & Breakfast completo di indirizzo e numero di telefono di ognuno di essi. A posteriori, analizzando tutti i dati ricevuti, abbiamo fatto la piacevole scoperta che il sito web utilizzato per il recupero dei dati offriva oltre a Bed & Breakfast anche case vacanze e altre tipologie simili di alloggio. Il fornitore identificato in prima battuta è stato il sito web `bed-and-breakfast.it`.

Figura 1.9: Bed and Breakfast.it

Con l'inserimento delle accommodation in CoolTraveling è stata completata una delle parti fondamentali per il successo del prodotto ovvero l'identificazione e relativa integrazione dei servizi esterni per il recupero delle informazioni da mostrare agli utenti. Una parte fondamentale per ogni progetto basato sul delivery di informazioni di servizi terzi. Completata questa fase era chiaro a tutti che la strada per la messa online di CoolTraveling fosse davvero spianata. Nelle settimane successive abbiamo completato la progettazione del prodotto focalizzandoci sui modelli dei dati e la loro mappatura logica. Tutto sembrava andare per il meglio, ma a due giorni dalla consegna del progetto per il concorso Vodafone tutto si complicò. Senza alcun preavviso i servizi di VolaGratis iniziarono a non rispondere più in modo corretto e fu subito chiaro che ciò che ci era stato così facilmente concesso in modo altrettanto facile stava svanendo nel nulla. I tentativi di contattare VolaGratis furono inutili e purtroppo CoolTraveling avrebbe perso tutta la sua essenza se fosse compromessa la possibilità di confrontare le soluzioni di viaggio di treni e aerei ed essendo VolaGratis il nostro unico fornitore dei servizi aerei il rischio era molto elevato. Con soli due giorni a disposizione e tanta voglia di partecipare al concorso abbiamo preso la decisione più rischiosa ovvero integrare nottetempo un'altro servizio per gli aerei. Non essendoci tempo per molte ricerche o valutazioni abbiamo scelto Expedia come servizio dal quale estrarre i dati. Ovviamente non essendo riusciti ad ottenere alcun accesso diretto ai loro servizi siamo stati costretti, come con Trenitalia, a simulare il comportamento degli utenti e fare il crawling a mano delle pagine web per estrarre i dati.

Figura 1.10: Expedia

La notte stessa in cui si chiudeva la finestra di validità per la presentazione del concorso sono state completate tutte le integrazioni e CoolTraveling fu messo online. Per la messa online del progetto ci siamo in prima battuta appoggiati ad un servizio in

cloud italiano: Seeweb. La ricerca del fornitore non è stata semplice in quanto la nostra architettura, seppur rudimentale, richiedeva il supporto per il deploy di applicazioni Rails, un dettaglio non trascurabile se si pensa al lontano 2010 e allo strapotere di Java a livello enterprise. Sino al giorno della pubblicazione ufficiale di CoolTraveling tutta la parte server del progetto è rimasta in hosting sui nostri computer o su qualche server casalingo configurato. L'avere accesso ad un diverso servizio di hosting nel cloud è stato il primo contatto di CoolTraveling verso il mondo esterno e la nostra prima esperienza diretta della messa live di un progetto.
Una parte fondamentale di tutto il progetto alla quale abbiamo sempre dedicato molta attenzione è lo sviluppo del client mobile: fulcro del concorso indetto da Vodafone. Come già anticipato precedentemente infatti Vodafone ha indetto l'App Start Competition per sponsorizzare il lancio della piattaforma proprietaria Vodafone360. Sin dalla prima presentazione di CoolTraveling per il WHYMCA abbiamo dedicato grande effort nella progettazione della user experience adeguata per il target del prodotto e condotto numerosi test di usabilty al fine di realizzare un App utile ma allo stesso tempo facile ed intuitiva nelle sue numerose sfaccettature. Se in prima battuta, durante il WHYMCA, tutti questi concetti erano rimasti sulla carta, con l'App Star abbiamo tramutato ogni singolo permisero in righe di codice AJAX realizzando il client mobile, definito widget per la piattaforma Vodafone360.
I giorni successivi alla pubblicazione dell'App sono stati giorni di grande fermento ed ansia per l'attesa di un feedback sulla nostra candidature da parte di Vodafone. Feedback è che puntualmente arrivato, contentente alcune critiche che il team di review di Vodafone aveva mosso alla nostra App. Tra le tante segnalazioni ricevute sul fronte tecnico ci è anche stato posta una sottile critica di concetto ovvero l'impossibilità di utilizzare l'App per ricerche in Europa ma solamente per l'Italia. Lì per lì il accusammo il colpo senza sapere che proprio grazie a questo spunto saremmo poi riusciti a trasformare l'App dal giorno alla notte. Inizialmente il progetto era partito per una platea italiana, quindi non avevamo assolutamente preso in considerazione questo capovolgimento di fronte internazionale. Fortunatamente però il fornitore scelto per i voli, Expedia, è un player di livello internazionale. Quindi dopo aver mappato le principali città europee associando per ciascuna l'aeroporto ad essa più vicino avevamo già assolto alle richieste di correzione mosse dalla commissione dell'App Star. Lo spunto che ci era stato fornito ci aveva fatto scoprire la grande scalabilità del prodotto che avevamo progettato e con un'altro piccolo sforzo CoolTraveling si trasformò completa-

mente aprendo le porte al turismo mondiale. La stessa procedura adottata su Expedia è stata replicata in maniera del tutto analoga per Accuweather, abilitando quindi le previsioni meteo per tutto il mondo. Purtroppo, per ovvi motivi, con Trenitalia e Bed & Breakfast.it non è stato possibile adottare il medesimo stratagemma. CoolTraveling era quindi trasformato da un'App interamente dedicata ai viaggi all'interno dell'Italia ad un'App in grado di valorizzare il turismo italiano nel mondo.
Dopo aver inviato l'ultima versione dell'App per l'approvazione da parte della commissione dell'App Star Competition non abbiamo più ricevuto alcuna comunicazione e magicamente nel nostro database sono iniziati a comparire utenti. Con nostro grande stupore la maggior parte degli utenti si registrava correttamente ed iniziava la ricerca di voli, treni e altri servizi con sempre più voracità. Ci fu subito chiaro che di questo passo l'hosting su Seeweb non sarebbe stato sufficiente e iniziò quindi la ricerca di un valido sostituto. Nelle settimane successive, ovvero durante la votazione del concorso il flusso di utenti continuava ad aumentare e decidemmo quindi di effettuare la migrazione da Seeweb al cloud di Amazon. Il passaggio non è stato indolore ma con qualche sacrificio e nottata extra siamo riusciti ad effettuare la migrazione con un disservizio di qualche ora dovuto principalmente alla propagazione dei nuovi indirizzi sui DNS. La migrazione è stata rigorosamente portata avanti durante la notte e il mattino seguente tutto funzionava come sempre senza il minimo problema.
Verso la fine del concorso, guardando il volume di utenti generati senza investire neanche un centesimo in advertising iniziò a farsi sempre più concreta l'idea di trasformare questo progetto in un business. Vedremo però che passare da un progetto ad un business non è per nulla paragonabile al concretizzare un'idea in progetto. Le dinamiche in gioco sono molteplici e non sempre un bel progetto è nato per far business.

1.6 Dal progetto al business

L'idea di tramutare il progetto in business era un chiodo fisso e i successi nelle due competition Italiana ed Europea ci convinsero sempre più che la strada che stavamo perseguendo avrebbe sicuramente portato al successo. La risonanza dei successi fu ottima e sponsorizzare il prodotto non era poi così difficile.
Per poter spiccare il volo, come ogni progetto che ha un suo perché, anche CoolTraveling avrebbe però avuto bisogno di un proprio business model e delle certezze sulle quali basare il proprio futuro. Non avendo internamente delle grandi competenze lato

business abbiamo in prima battuta chiesto supporto ai docenti del Dipartimento di Ingegneria Gestionale del Politecnico di Milano. Grazie ai loro suggerimenti e consigli siamo riusciti a preparare il cosiddetto *Elevator Pitch*, da presentare agli investitori e sondare l'attrattività che CoolTraveling potesse generare. Dopo aver studiato il mercato e prodotto il documento abbiamo iniziato a bussare alla porta dei Venture Capitalist italiani. L'esperienza maturata con Dpixel è stata abbastanza netta, non abbiamo mai ricevuto ne' risposta ne alcun feedback in merito al nostro progetto. Senza perderci d'animo abbiamo continuato a lavorare cercando di trovare l'anello mancante che potesse trasformare CoolTraveling da un progetto vincente a un business.

1.11)

Figura 1.11: Dpixel

Analizzando però nel dettaglio ogni aspetto ci fu chiaro che il nostro prodotto era difficilmente autosostenibile allo stadio attuale e le uniche entrate sarebbero potute arrivare dalla pubblicità. Compresa la difficoltà dell'incanalare un giro d'affari dentro CoolTraveling abbiamo iniziato a pensare a sue future evoluzioni o come poter rendere CoolTraveling un servizio per un futuro prodotto. Il passaggio fu pressoché immediato: quale miglior servizio per chi pianifica viaggi di un comparatore di soluzioni di viaggio? Sicuri di aver intrapreso la strada giusta nell'Ottobre del 2010 abbiamo presentato il progetto CoolTraveling revised agli occhi della commissione del concorso New York City Next Idea[2]. Purtroppo anche in questo caso i feedback ricevuti dalla commissione sono stati in linea con i feedback ricevuti sul campo ovvero di un prodotto interessante, innovativo ma non in grado di autosostenersi. Un'ottimo progetto ma non un'idea da calare in una startup com'era nostro intento fare. Sicuramente a dettare le sorti su come si è evoluto il prosieguo di CoolTraveling ha giocato un grande ruolo la nostra

[2]New York Next Idea è un concorso per imprenditori dove vengono premiate le migliori idee imprenditoriali da realizzare direttamente sul fertile terreno della grande mela. Ogni anno i vincitori del concorso vengono invitati a trascorrere un periodo a NYC per sviluppare la loro idea di business.

inesperienza nel mondo del lavoro e nella percezione di come rendere o meno un'idea profittevole. Infatti per un progetto che non è decollato: CoolTraveling, qualcuno nel mondo ci è riuscito. Stiamo parlando di Kayak[3], un'App mobile interamente incentrata sulla pianificazione dei viaggi. Kayak all'onor del vero offre molti più servizi di quelli da noi ideati e come possiamo vedere dal livello di popolarità raggiunto è percepito dai propri utenti come un prodotto utile ed interessante.

Figura 1.12: Kayak

Siamo quindi sempre più convinti che se non oggi ma in futuro anche CoolTraveling riuscità ad affermarsi tra la concorrenza, basterà solo trovare le partnership giuste con i fornitori dei servizi e finanziamenti da parte delle regioni interessate a convogliare turisti nelle proprie località. Del resto prima di noi qualcuno di un po' più *grosso* e strutturato aveva già pensato di rivoluzionare il settore dei viaggi con *iTravel*[9].

1.13)

1.7 Introduzione capitoli successivi

Nei capitoli successivi vedremo come ogni singolo aspetto sopra enunciato celi alla sue spalle un grande la lavoro di ideazione, progettazione, sviluppo e team working che difficilmente può essere descritto in un semplice documento.

- **Capitolo 1: Il progetto**
  All'interno del primo capitolo è descritta la storia del progetto, dalla sua nascita sino al lancio sul mercato. Particolare attenzione è stata dedicata alle vicende che hanno portato CoolTraveling e il suo team a trionfare sul panorama mobile europeo.

[3]Kayak è un servizio attivo su scala mondiale che permette di ricercare e prenotare: voli, hotel, viaggi e auto.

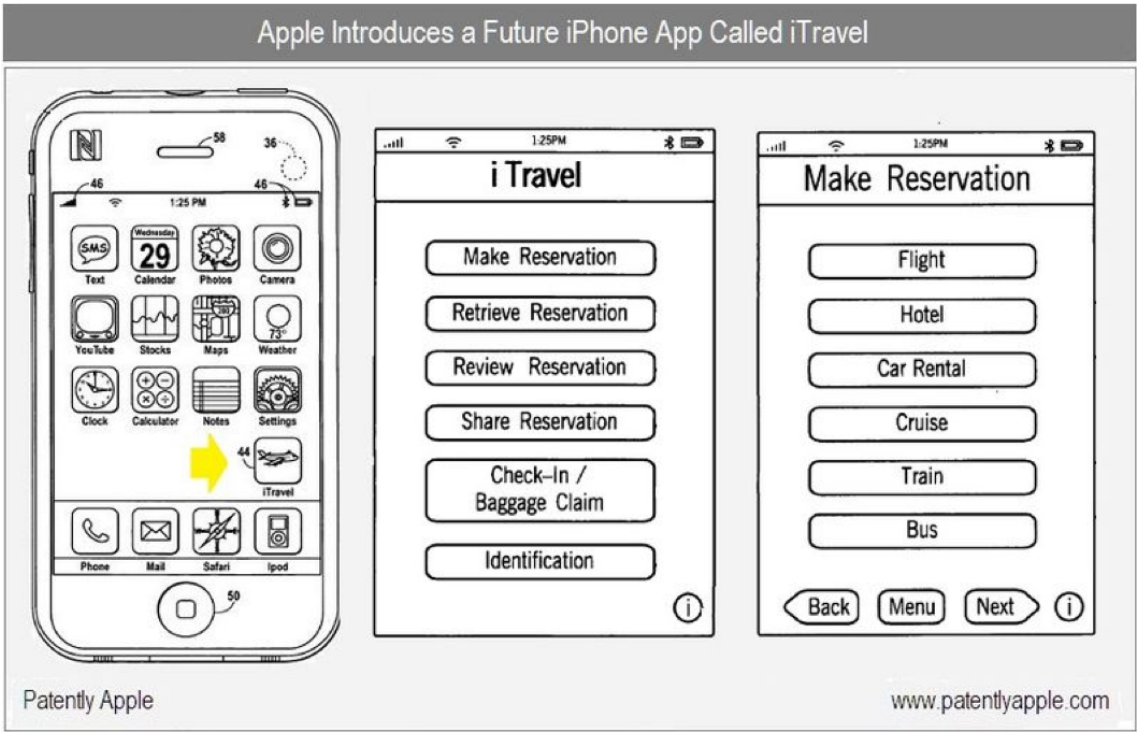


Figura 1.13: Apple - iTravel

- **Capitolo 2: Analisi dei requisiti**
  Il secondo capitolo descrive il lavoro svolto durante la fase di elicitazione, ovvero l'analisi dei requisiti del progetto. In questo capitolo sono quindi descritti tutti gli studi effettuati al fine di ottenere una mobile application ricca di funzionalità e allo stesso tempo intuitiva e facile da usare.

- **Capitolo 3: Architettura Software Back-end**
  Il terzo capitolo descrive il cuore di CoolTravelig, l'architettura sulla quale tutto il progetto si basa. Sono descritte nel dettaglio si le scelte intraprese che la struttura stessa delle singole parti architetturali partendo dal noto modello MVC (Model View Controller).

- **Capitolo 4: Cloud Computing**
  Il settimo capitolo è interamente dedicato agli studi effettuati sul cloud computing e alla nostra esperienza del porting di CoolTraveling sul cloud. Sono descritti i servizi utilizzati e le scelte architetturali intraprese per dare elasticità e scalabiltà al progetto in tutte le sue componenti.

- **Capitolo 5: Servizi: Viaggiatreno, Meteo, Accomodation**
  Nel quinto capitolo vengono descritti i nuovi servizi introdotti in CoolTraveling: Viaggiatreno, Meteo, Accommodation. Ampio spazio è dedicato all'introduzione della cache per l'ottimizzazione dei servizi e la scalabilità delle richieste.

- **Capitolo 6: L'applicazione mobile**
  Nel sesto capitolo è descritta l'applicazione mobile realizzata per l'interrogazione dei servizi esposti dal backend di CoolTraveling. Ampio spazio è dedicato alla stesura dei Mockup e al design dell'applicazione.

- **Capitolo 7: Storyboard**
  Il settimo capitolo è relativo alle due storyboard che descrivono con esempi reali come CoolTraveling sia in grado di introdurre un'innovazione significativa nel panorama mobile della pianificazione viaggi. A supporto delle storyboard sono presenti diagrammi di interazione (sequenza, collaborazione) e comportamento (stato, attività).

- **Capitolo 8: Conclusioni e sviluppi fururi**
  L'ottavo capitolo è dedicato alle conclusioni e agli sviluppi futuri del progetto CoolTraveling.

Capitolo 2

Analisi dei requisiti

2.1 Attori e requisiti funzionali

Dato che il software CoolTraveling è stato sviluppato per raccogliere un elevato bacino di utenza, l'unico attore presente per l'analisi è qualsiasi possessore di uno smartphone.

Le funzionalità richieste per l'**utente** sono:

- possibilità di loggarsi ed usufruire dei servizi messi a disposizione dall'applicazione
- impostare i propri settings
    - eseguire la login automatica
    - impostare la lingua preferita (italiano, inglese)
    - numero di risultati da visualizzare per ogni schermata
    - ordinare le soluzioni di viaggio in base a tempo, durata e prezzo
    - reset account/modifica password
- inserire qualsiasi comune italiano nel campo di ricerca
- inserire città straniere munite di aeroporto nel campo di ricerca
- definire data di andata/ritorno
- possibilità di decidere quali mezzi di trasporto includere nella ricerca
- visualizzazione e confronto di soluzioni di viaggio di diversi mezzi di trasporto

- visualizzazione del dettaglio di una soluzione
- visualizzazione dei dettagli relativi alle accommodations
- visualizzazione delle previsioni meteorologiche

2.2 Requisiti non funzionali

2.2.1 Usabilità

Sebbene sia rivolto ad utenti che presentano background culturali abbastanza eterogenei, dato che l'applicazione può essere utilizzata, ad esempio, sia da un uomo di business in viaggio d'affari o da uno studente che pianifica una vacanza estiva, il sistema non presenta un elevato grado di complessità. Gli utenti che interagiscono con CoolTraveling, devono essere in grado di soddisfare le proprie esigenze non appena diventa chiaro il dominio a cui si rivolge l'applicazione(in sostanza: "*che cosa fa?*").

2.2.2 Affidabilità

Il sistema è basato su un hosting cloud il quale garantisce la persistenza dei dati dell'utente; sono previsti back-up periodici, attualmente di cadenza giornaliera. Questo argomento verrà ulteriormente approfondito nel capitolo relativo al Cloud Computing

2.2.3 Prestazioni

Il sistema prevede un hardware performante. È in grado di interagire contemporaneamente con centinaia di utenti mantenendo contenuti i tempi di risposta.

2.2.4 Interfaccia

L'utente interagisce con il sistema mediante un'applicazione mobile dedicata che scambia dati con il server mediante interfaccia WSDL[5] o Rest[6].

2.2.5 Gestione

La gestione del sistema viene effettuata da uno o più soggetti con account da Amministratore.

2.2.6 Accessibilità

L'applicazione ha avuto come punto di riferimento l'utente, in modo da creare un sistema chiaro ed intuitivo: il messaggio trasmesso è coerente in ogni view.

2.2.7 Sicurezza

Il sistema è dotato di un meccanismo di autenticazione. Lato server, la sicurezza è garantita dalla presenza di firewall hardware e dal monitoraggio costante delle attività, in modo da evitare qualsiasi possibilità di attacco. Lato widget, non sono necessari alti livelli di sicurezza in quanto le informazioni scambiate non sono sensibili.

2.3 Dominio

Per dominio si intende il mondo reale in cui il sistema andrà a inserirsi e a interfacciarsi.
Fanno parte del dominio le entità di interesse, gli stakeholders, le relazioni e gli influssi che il sistema ha su queste ultime.

2.3.1 Entità

Dall'analisi del problema sono emerse le seguenti entità-chiave:

- Users;
- Sessions
- Requests
- Routes
- Trips
- Flight_Details
- Trains_Details

2.3.2 Stakeholders

Le principali persone o enti che hanno una determinata aspettativa dal sistema sono:

- l'utente finale (il viaggiatore) che ha come obiettivo quello di velocizzare il più possibile la pianificazione di un viaggio.
- l'agenzia viaggio che ha come obiettivo avere il confronto tra tipologie di mezzi di trasporto differenti.

Le principali persone o enti che potranno avere una determinata aspettativa dal sistema sono:

- gestore di un albergo o hotel che ha come obiettivo quello di pubblicizzare la sua attività
- gestore di un mezzo di trasporto che ha come obiettivo quella di promuovere il suo mezzo di trasporto

2.3.3 Relazioni

Nell'ambito del dominio dell'applicazione sono state individuate diverse relazioni tra entità, di cui riportiamo le più significative:

- Un utente può avere più sessioni ad esso associate (ma soltanto l'ultima sarà valida per effettuare ricerche). Una sessione viene generata quando l'utente effettua la login.
- Un utente può effettuare più richieste (request), una richiesta è associata solo ad un utente.
- Ad ogni richiesta sono associate più soluzioni di viaggio (route).
- Ad ogni route è associato almeno un trip e il numero dipende dal tipo di mezzi di trasporto scelti
- ad ogni trip è associato un dettaglio treno o un dettaglio aereo a seconda del mezzo di trasporto relativo al trip.

2.4 Casi d'Uso

Nei prossimi paragrafi sono rappresentati i diagrammi dei casi d'uso[1], i quali descrivono informalmente il comportamento sistema per quanto riguarda l'interazione con l'utente.

Di seguito riportiamo il diagramma dei casi d'uso complessivo delle funzionalità del sistema (Figura 7.18). Oltre ai casi d'uso relativi al sistema proposto ne verrà illustrato uno aggiuntivo (Download CoolTraveling) utile per eseguire il download e l'installazione di CoolTraveling sul proprio device.

2.4.1 Download CoolTraveling

Titolo Download CoolTraveling

Descrizione

Attori Partecipanti L'Utente

Condizioni d'Ingresso
- possesso della piattaforma Vodafone 360
- possesso di una connessione ad internet

Flusso di Eventi
1. l'utente accede allo store della piattaforma di riferimento;
2. l'utente ricerca l'applicativo immettendo parole chiavi (come: viaggiare, trasporti, ecc.);
3. l'utente seleziona CoolTraveling;
4. l'utente scarica e installa CoolTraveling sul suo smartphone;

Condizioni d'Uscita
- l'applicativo CoolTraveling è installato sul proprio smartphone

Eccezioni
- per problemi di rete è possibile che il downoad si interrompa e debba essere riavviato

[1] Lo "Use Case" è una descrizione di un comportamento particolare di un sistema dal punto di vista dell'utente. Con questo diagramma è possibile ottenere una chiara definizione dei requisiti del sistema sempre dal punto di vista dell'utente, evitando così di ottenere una difficoltosa e poco chiara interpretazione del problema.

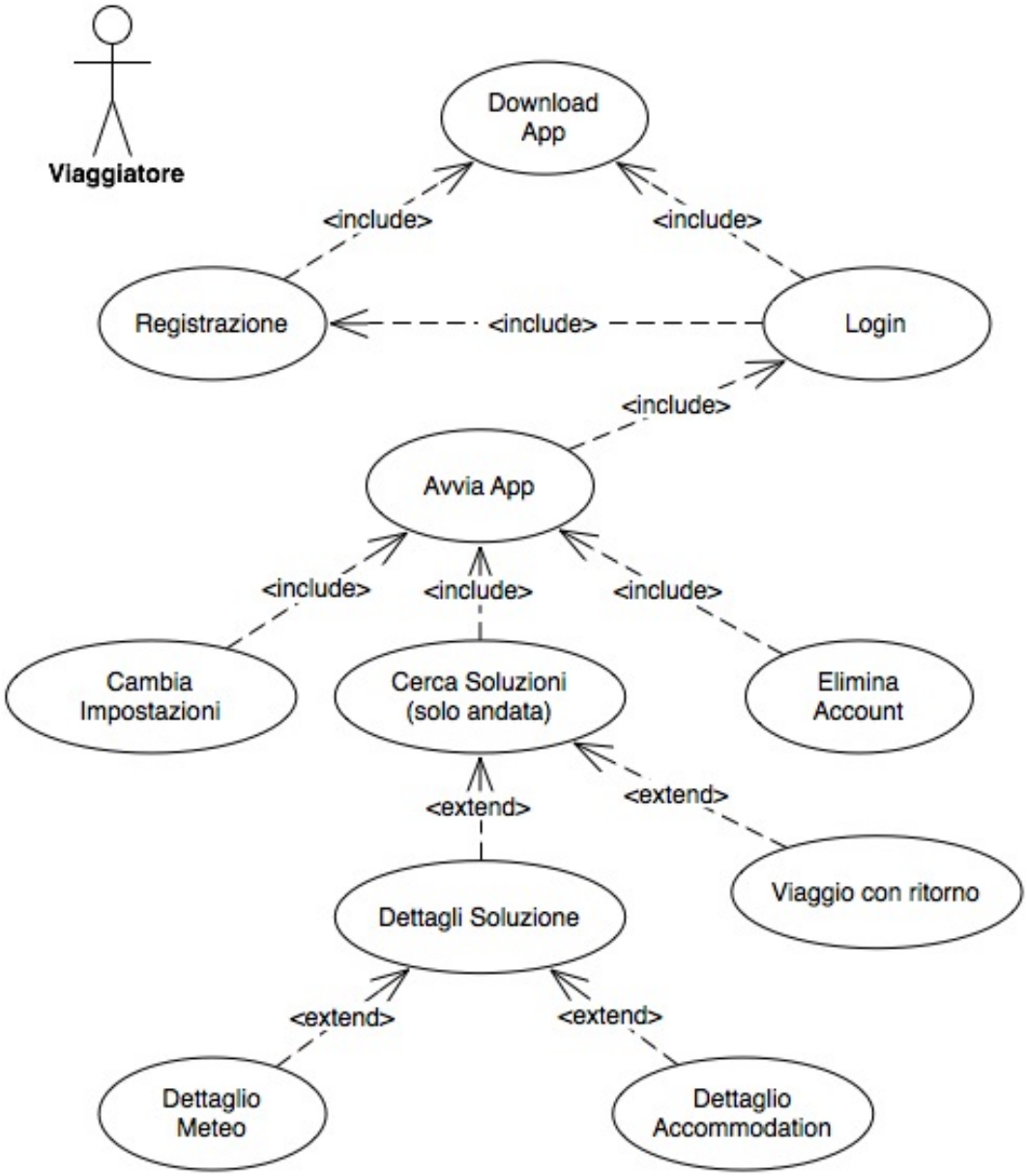


Figura 2.1: Diagramma: Caso d'Uso - complessivo

2.4.2 Registrazione

Titolo Registrazione

Descrizione Per poter utilizzare CoolTraveling è necessaria una registrazione che si conclude con la conferma dell'account mail.

Attori Partecipanti L'Utente

Condizioni d'Ingresso

- Utente non registrato
- CoolTraveling non è stato mai avviato o non è presente nessun account salvato sul device

Flusso di Eventi

1. L'Utente avvia l'applicativo;
2. CoolTraveling risponde visualizzando la form iniziale di login
3. L'utente clicca sul pulsante Registrati
4. CoolTraveling visualizza la form di inserimento dei dati necessari per la registrazione
5. L'Utente inserisce i dati necessari per la registrazione e accetta le condizioni di utilizzo
6. CoolTraveling notifica all'Utente che per completare la registrazione è necessario verificare la mail inviata per confermare l'account
7. l'Utente visualizza la mail precedentemente inviata e clicca sul link presente all'interno della mail
8. CoolTraveling notifica all'utente il completamento della registrazione (l'esito della richiesta)

Condizioni d'Uscita

- registrazione effettuata
- mail abilitata per usare i servizi CoolTraveling

Eccezioni

- se al punto 2 CoolTraveling visualizza la form per la ricerca di un viaggio bisognerà eseguire il caso d'uso elimina account
- se la mail è già presente nel sistema, non sarà possibile completare la registrazione, in quanto l'utente è già registrato
- se la mail non esiste non sarà possibile validare la correttezza della stessa;

2.4.3 Login

Titolo Login

Descrizione Prima di utilizzare CoolTraveling è necessario autenticarsi tramite una login.

Attori Partecipanti L'Utente

Condizioni d'Ingresso

- CoolTraveling non è stato mai avviato o non c'è nessun account salvato sul device
- L'Utente deve essere in possesso di un account CoolTraveling

Flusso di Eventi

1. L'Utente avvia CoolTraveling
2. CoolTraveling visualizza la form iniziale di login
3. L'utente inserisce i dati necessari per la login (username e password)
4. CoolTraveling controlla che l'username e la password forniti siano corretti
5. CoolTraveling effettua l'autenticazione dell'Utente e invia un messaggio di conferma.
6. CoolTraveling salva sul device le informazioni relative alla login automatica
7. L'Utente visualizza la conferma dell'avvenuto login

Condizioni d'Uscita

- L'Utente è autenticato in CoolTraveling

Particolari Esigenze

- username deve essere una mail (esempio formato: name@domain.com)
- la password deve essere almeno di 5 caratteri

Eccezioni

- se al punto 4 username e/o password non saranno corretti CoolTraveling tornerà al punto 2 e la login non verrà effettuata
- se username usato corrisponde ad un utente che non ha completato la procedura di registrazione, CoolTraveling gli notificherà i passi da effettuare

2.4.4 Avvia CoolTraveling

Titolo Avvia CoolTraveling

Descrizione L'Utente avvia CoolTraveling

Attori Partecipanti L'Utente

Condizioni d'Ingresso

– non ci sono condizioni d'ingresso –

Flusso di Eventi

1. l'utente avvia CoolTraveling
2. CoolTraveling risponde visualizzando la form di inserimento dati per la ricerca

Condizioni d'Uscita

- CoolTraveling è pronto per effettuare ricerche

Particolari Esigenze

– non ci sono particolari esigenze –

Eccezioni

- se al punto 2 CoolTraveling risponde visualizzando la form di login è necessario:

- procedere con il caso d'uso login qualora l'Utente possieda un account valido
- procedere con il caso d'uso registrazione se l'Utente non è registrato nel sistema

2.4.5 Elimina Account salvato nel device

Titolo Elimina Account salvato nel device

Descrizione Effettuando una login viene salvato sul device le informazioni necessarie per eseguire la login in modo automatico all'avvio successivo. Con questo caso d'uso verrà illustrata la procedura per eliminare/cambiare account.

Attori Partecipanti L'Utente

Condizioni d'Ingresso

- L'Utente deve aver effettuato almeno una login
- L'Utente ha avviato CoolTraveling

Flusso di Eventi

1. CoolTraveling visualizza la form di inserimento dati per la ricerca
2. L'Utente clicca sul tasto relativo alle configurazioni
3. CoolTraveling visualizza le configurazioni
4. L'Utente clicca sul tasto Account
5. CoolTraveling visualizza la view di configurazioni dell'Account
6. L'Utente clicca sul tasto Elimina Account
7. CoolTraveling risponde eliminando i dati relativi alla login automatica e visualizzando la form iniziale di login

Condizioni d'Uscita

- l'utente non risulta loggato
- sul suo smartphone non sono presenti informazioni necessarie per la login automatica

Particolari Esigenze

– non ci sono particolari esigenze –

Eccezioni

- se username e/o password non sono corrette la login non verrà effettuata
- se username usato corrisponde ad un utente che non ha completato la procedura di registrazione, CoolTraveling gli notificherà i passi da effettuare

2.4.6 Cambia Impostazioni

Titolo Cambia Impostazioni

Descrizione Su ogni device sono salvate delle impostazioni (associate ad un particolare utente), in questo caso d'uso possiamo vedere come è possibile modificarle;

Attori Partecipanti L'Utente

Condizioni d'Ingresso

- l'utente deve aver fatto almeno una login sul device

Flusso di Eventi

1. si proceda secondo il caso d'uso Avvia CoolTraveling
2. CoolTraveling visualizza la form di inserimento dati per la ricerca
3. l'utente clicca sul tasto relativo ai settings dell'applicazione
4. CoolTraveling visualizza la view realiva ai settings
5. l'utente modifica in maniera opportuna le impostazioni
6. CoolTraveling salva automaticamente le impostazioni

Condizioni d'Uscita

- aggiornate le impostazioni di CoolTraveling

Particolari Esigenze

– non ci sono particolari esigenze –

Eccezioni

– non ci sono eccezioni –

2.4.7 Viaggio solo andata

Titolo Viaggio solo andata

Descrizione Viaggiatore esegue una ricerca (solo andata) scegliendo, successivamente, un dettaglio viaggio in particolare;

Attori Partecipanti L'Utente

Condizioni d'Ingresso

- login effettuata correttamente

Flusso di Eventi

1. si proceda secondo il caso d'uso "Avvia CoolTraveling"
2. l'utente inserisce la città di partenza
3. CoolTraveling procede alla validazione della città di partenza inserita
4. CoolTraveling notifica all'utente l'esito
5. l'utente inserisce la città di destinazione
6. CoolTraveling procede alla validazione della città di destinazione inserita
7. CoolTraveling notifica all'utente l'esito
8. l'utente inserisce la data di andata
9. CoolTraveling verifica la validità/coerenza della data inserita
10. l'utente clicca su "Cerca"
11. CoolTraveling elabora la richiesta di ricerca
12. CoolTraveling notifica all'utente l'esito della richiesta visualizzando la view delle soluzioni
13. l'utente visualizza l'elenco delle soluzioni
14. l'utente clicca sulla soluzione preferita

15. CoolTraveling elabora il dettaglio della soluzione scelta
16. CoolTraveling visualizza il dettaglio richiesto sul device
17. l'utente prende visione del dettaglio scelto

Condizioni d'Uscita

- l'utente ha scelto una soluzione di viaggio e ne visualizza i dettagli

Particolari Esigenze

- La validazione della città si comporta come un warning per l'utente. Città non validate verranno trattate nella ricerca come città straniere.
- se una delle città non è stata verificata il check relativo non sarà verde

Eccezioni

– non ci sono eccezioni –

2.4.8 Viaggio completo

Titolo Viaggio completo

Descrizione Il viaggiatore può eseguire una ricerca completa (andata - ritorno) visualizza le soluzioni, prima l'andata e successivamente il ritorno (scegliendone una per tipologia).

Attori Partecipanti L'Utente

Condizioni d'Ingresso

- applicativo avviato
- appena effettuata la login oppure è presente un account CoolTraveling salvato sullo smartphone

Flusso di Eventi

1. si proceda secondo il caso d'uso Avvia CoolTraveling
2. l'utente inserisce la città di partenza
3. CoolTraveling procede alla validazione della città di partenza inserita

4. CoolTraveling notifica all'utente l'esito
5. l'utente inserisce la città di destinazione
6. CoolTraveling procede alla validazione della città di destinazione inserita
7. CoolTraveling notifica all'utente l'esito
8. l'utente inserisce la data di andata
9. CoolTraveling verifica la validità/coerenza della data inserita
10. l'utente inserisce la data di ritorno
11. CoolTraveling verifica la validità/coerenza della data inserita
12. l'utente clicca su Cerca
13. CoolTraveling elabora la richiesta di ricerca
14. CoolTraveling notifica all'utente l'esito della richiesta visualizzando le soluzioni di andata
15. l'utente visualizza l'elenco delle soluzioni di andata
16. l'utente clicca sulla soluzione di andata preferita
17. l'utente clicca sul tasto ritorno
18. CoolTraveling mostra la view delle soluzioni di ritorno
19. l'utente visualizza l'elenco delle soluzioni di ritorno
20. l'utente clicca sulla soluzione di ritorno preferita
21. CoolTraveling elabora il dettaglio delle soluzioni A/R scelte
22. CoolTraveling notifica all'utente la possibilità di visualizzare il dettaglio completo
23. l'utente procede con la visualizzazione del dettaglio completo

Condizioni d'Uscita

- l'utente ha scelto una soluzione di viaggio completo (andata e ritorno)
- l'utente prende visione della view relativa al dettaglio completo

Particolari Esigenze

- La validazione della città si comporta come un warning per l'utente. Città non validate verranno trattate nella ricerca come città straniere.

- se una delle città non è stata verificata il check relativo non sarà verde

Eccezioni

– non ci sono eccezioni –

2.4.9 Dettaglio accommodation

Titolo Dettaglio accommodation

Descrizione Visualizzando una soluzione di viaggio è possibile approfondire il dettaglio visualizzando l'accommodation della città interessata.

Attori Partecipanti L'Utente

Condizioni d'Ingresso

- l'utente ha scelto almeno un dettaglio di soluzione

Flusso di Eventi

1. si procede secondo il caso d'uso viaggio solo andata o viaggio completo
2. l'utente clicca sulla i di informazioni in corrispondenza della città desiderata
3. l'utente seleziona l'immagine corrispondente all'elenco delle accommodation
4. CoolTraveling elabora la richiesta e inoltra l'esito
5. L'Utente visualizza l'elenco delle accomodation
6. l'utente clicca sull'accommodation preferita
7. CoolTraveling elabora la richiesta e inoltra l'esito
8. L'Utente visualizza il dettaglio dell'accomodation scelta precedentemente

Condizioni d'Uscita

- visualizzazione delle informazioni necessarie relative al B&B/hotel prescelto (prezzo, numero telefonico, via ...)

Particolari Esigenze

- se la città non è italiana non sarà visualizzata l'icona dell'accommodation (quindi non sarà possibile visualizzare le accommodation)

Eccezioni

– non ci sono eccezioni –

2.4.10 Dettaglio meteo

Titolo Dettaglio meteo

Descrizione Visualizzando una soluzione di viaggio è possibile approfondire il dettaglio visualizzando le previsioni della città interessata.

Attori Partecipanti L'Utente

Condizioni d'Ingresso

- è stato scelto almeno un dettaglio soluzione

Flusso di Eventi

1. si procede secondo il caso d'uso viaggio solo andata o viaggio completo
2. l'utente clicca sulla i di informazioni in corrispondenza alla città desiderata
3. l'utente seleziona l'immagine corrispondente al meteo
4. L'Utente visualizza l'elenco dei giorni con le previsioni meteorologiche

Condizioni d'Uscita

- visualizzazione delle informazioni relative alle condizioni meteorologiche

Particolari Esigenze

- non verranno visualizzate le previsioni meteorologiche se la data del viaggio risulta superiore a 8 giorni dalla data odierna

Eccezioni

– non ci sono eccezioni –

Capitolo 3

Architettura Software Back-end

In questo capitolo verranno trattate tutte le tecnologie sulle quali si basa il prodotto CoolTraveling.
Il backed a supporto del progetto, secondo il nostro approccio, doveva garantire immediata scalabilità, velocità di sviluppo e allo stesso tempo adottare una struttura semplice e funzionale per gli scopi che avevamo prefissato. Il capitolo seguente serve al lettore per comprendere appieno l'infrastruttura realizzata e le scelte che hanno portato alla sua realizzazione.

3.1 Service Oriented Architecture

Nell'ingegneria del software, una Service-Oriented Architecture (SOA) è un set di principi e tecnologie per progettare e sviluppare dei servizi interoperabili. Questi servizi sono delle funzionalità di business ben definite, costruiti come componenti software che possono essere riutilizzati per differenti scopi.

Non c'è un'unica definizione di SOA che è stata accettata all'unanimità da tutti quanti. In realtà esistono diverse definizioni, alcune delle quali si focalizzano al Business, quindi cosa le SOA possono fare per migliorare e ottimizzare quei processi, altre invece si dedicano agli aspetti prettamente tecnici. Vale sicuramente la pena citarne qualcuna:

- *W3C:* È un set di componenti che possono essere invocate, le cui interfacce e descrizioni possono essere pubblicate e trovate.

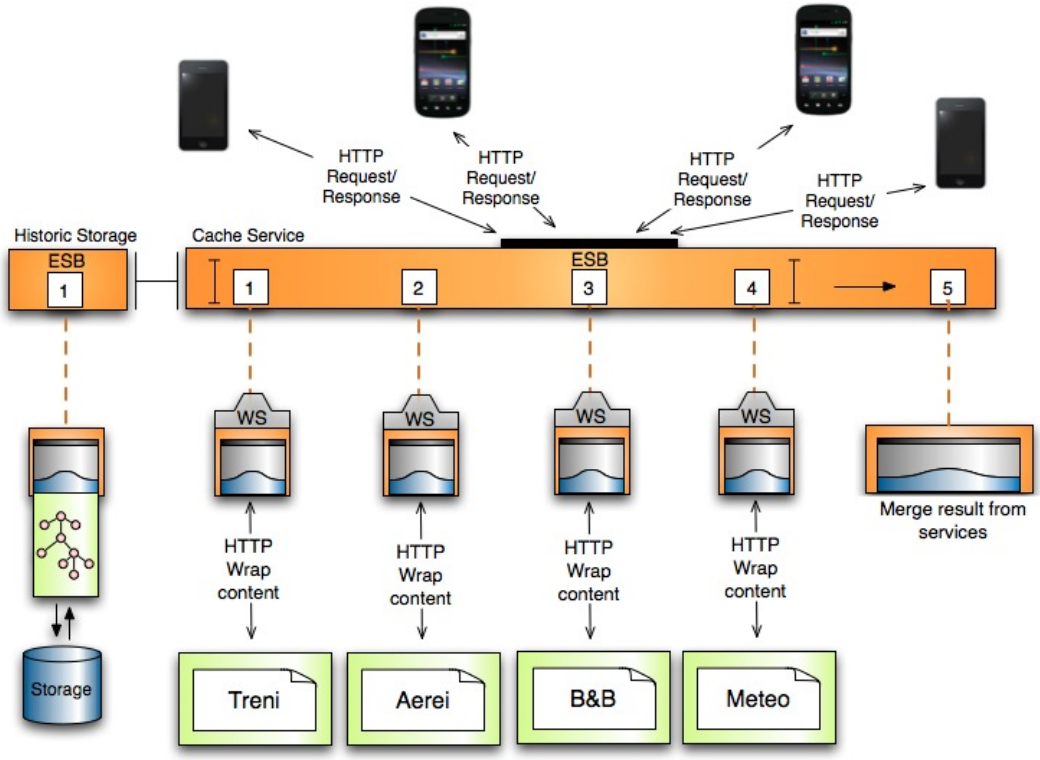


Figura 3.1: Architettura CoolTraveling

- *IBM:* È un'architettura IT scalabile per linkare delle risorse in base alle necessità. Queste risorse sono allineate alle linee di business in modo da poterne soddisfare i bisogni. **Prospettiva di business:** set di servizi che il business vuole esporre ai propri clienti e partner o ad altre parti dell'organizzazione. **Prospettiva architetturale**: Set di principi architetturali, pattern e criteri, che hanno alla base la modularità, il loose coupling, il riuso del codice, la composizione e la singola implementazione. **Prospettiva di implementazione**: un modello di programmazione completa di standard, strumenti e tecnologie come i Web Services.

SOA può anche essere vista come uno stile dell'architettura dei sistemi informatici che permetta la creazione delle applicazioni sviluppate, combinando servizi debolmente accoppiati e interoperabilità degli stessi. Questi servizi interoperano secondo una definizione formale, come per i WSDL, indipendente dalla piattaforma sottostante e dalle tecnologie di sviluppo (come Java, Ruby on Rails, ...). Le applicazioni in esecuzione su una piattaforma possono anche utilizzare servizi in esecuzione su altre, come con i Web services, facilitando quindi la riusabilità.

A seguire potremo vedere come nel nostro caso abbiamo ritenuto opportuno utilizzare un'architettura differente che meglio si presta ai nostri scopi. CoolTraveling è infatti un progetto con basso impatto computazionale ricco di integrazioni verso servizi esterni.

Come è possibile notare dallo schema in figura 3.2 l'architettura è stata progettata per essere modulare e scalabile in ogni suo componente, dalle istanze contenenti le web application ai layer di cache e database sottostanti. Tutte le componenti utilizzate verranno ampiamente descritte in seguito nel capitolo sul Cloud Computing.

In figura 3.3 possiamo invece vedere come i vari moduli dell'architettura dialogano tra loro interfacciandosi con i servizi esterni per il wrapping e la gestione dei dati.
Di seguito verranno descritte tutte le componenti interne alla Web App di CoolTraveling:

Server

Il server è un'istanza Amazon EC2 con sistema operativo Linux che verrà ampiamente discusso nel capitolo relativo al Cloud Computing.

Web Server: Apache

Il web server che abbiamo scelto per gestire le connessioni provenienti dal load balancer è Apache. Abbiamo ritenuto essere la scelta migliore in quanto già utilizzato in passato sia da noi che da molte altre persone con ottimi risultati. Dal nostro punto di vista serviva un Web Server in grado di integrarsi con Web Application scritte in Ruby e Apache, tramite moduli esterni lo permette.

Application Server: Phusion Passenger

Come anticipato poco sopra, Apache da solo non è in grado di gestire una Web Application sviluppata in Ruby on Rails. E' necessario quindi configurare un Application Server specifico all'interno di Apache per la gestione delle Web App. L'application server selezionato è Passenger[1]. Tramite Phusion Passenger è possibile configurare le Web Application realizzate in RoR facendo si che Apache instradi correttamente le connessioni dirette a loro. Nel file di configurazione è possibile definire nei minimi dettagli la configurazione della propria Web App, configurando:

[1]Phusion Passenger è anche conosciuto con il nome di mod_rails o mod_rack: http://www.modrails.com/

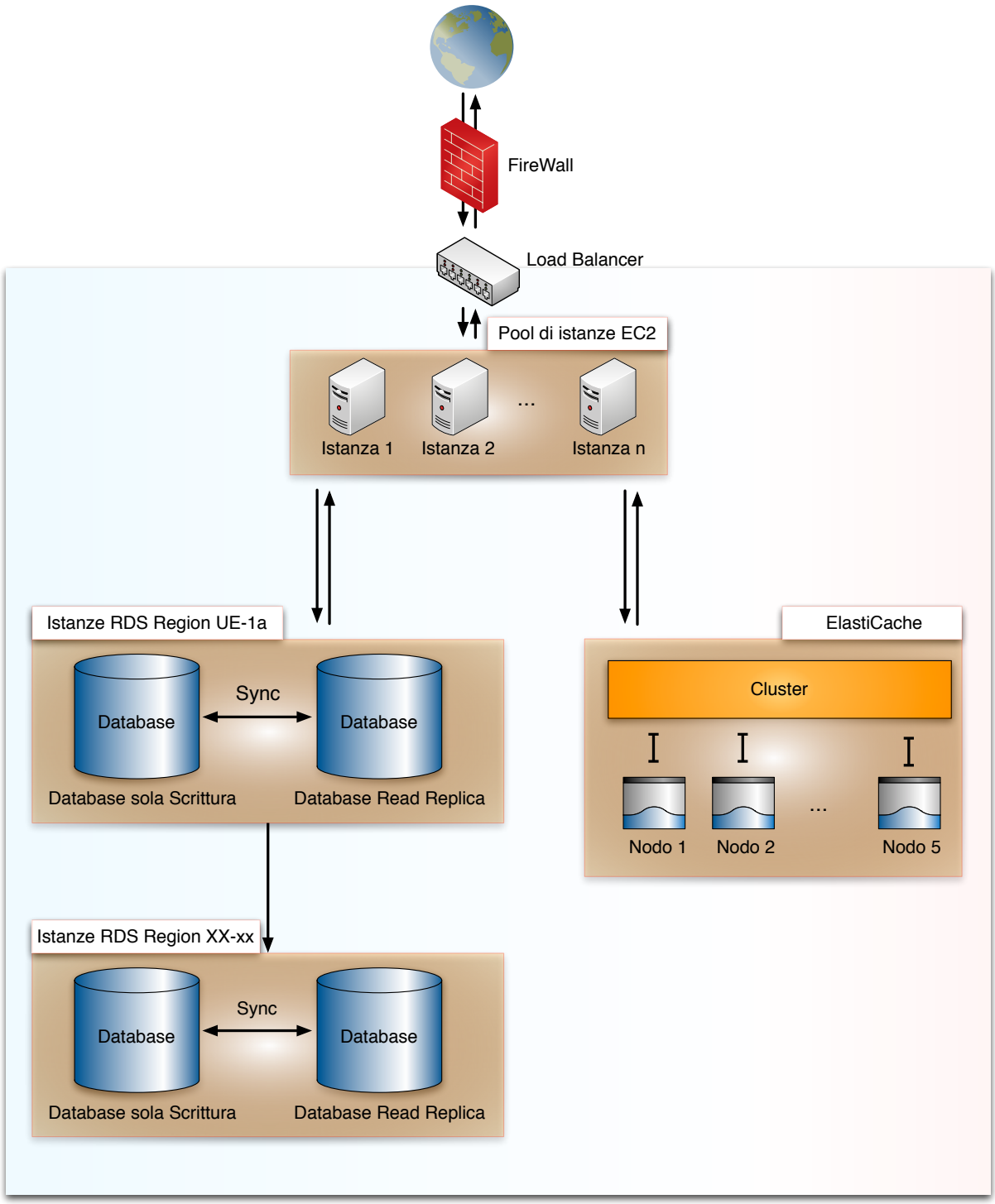


Figura 3.2: Architettura Backend CoolTraveling

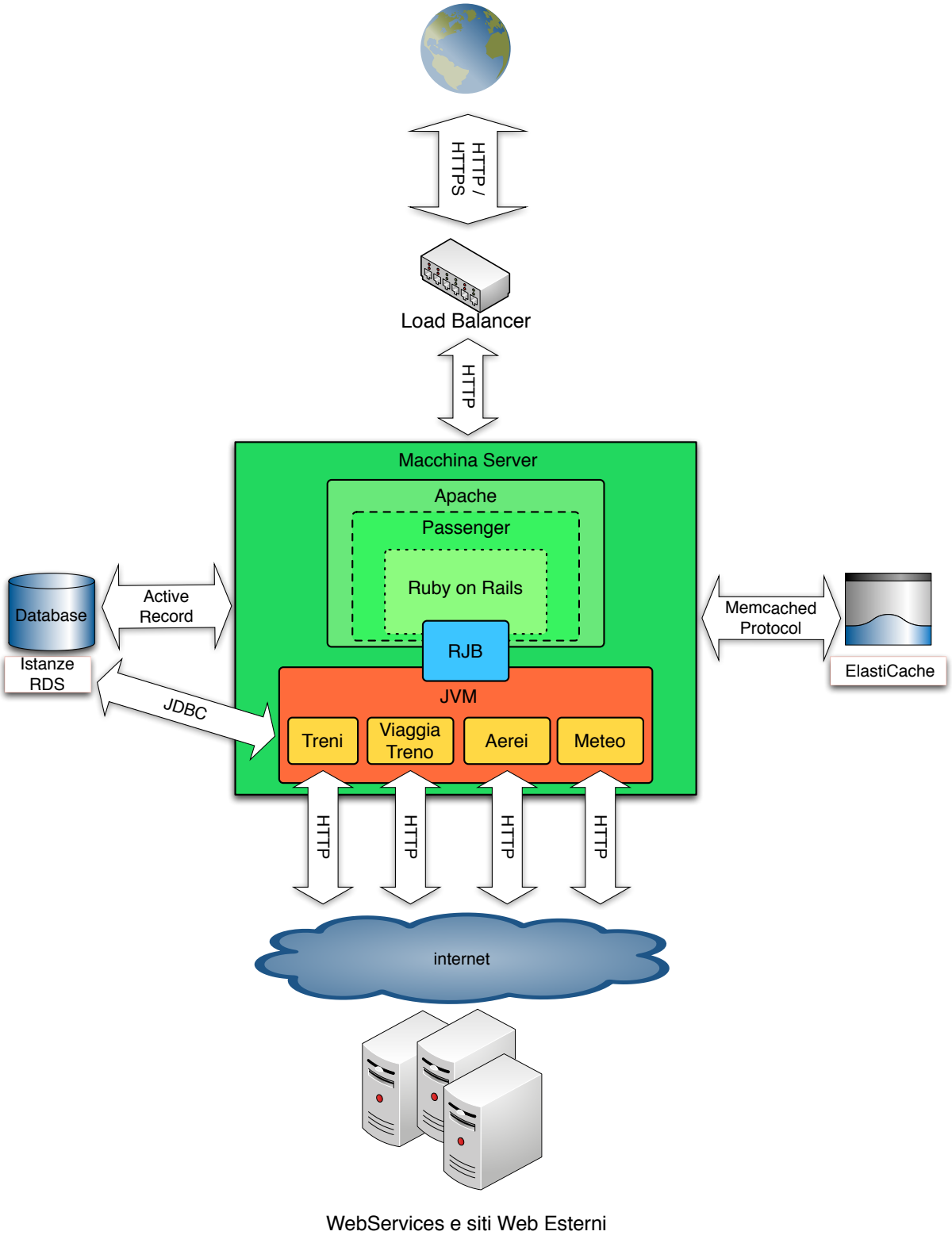


Figura 3.3: Architettura Web App interna di CoolTraveling

- Porta di accesso
- Indirizzi permessi
- Public folder
- Pool di istanze

Web Application: Ruby on Rails

Abbiamo deciso di adottare Ruby on Rails per la realizzazione della Web Application perché abbiamo riscontrato notevoli vantaggi rispetto al noto rivale di settore Java. Maggiori dettagli saranno definiti nella sezione 3.2.

Connettori eastern: RJB per JVM

Nella nostra architettura abbiamo integrato un connettore RJB (Ruby Java Connector) per consentire l'accesso a moduli scritti in java senza dover necessariamente avere una web application in java operativa. Nella sezione 3.3.3 sono ampliamente descritte le motivazioni di questa scelta.

Database: Active Record & JDBC

L'accesso al database è effettuato da due diverse parti della nostra architettura:

- Rails: Active Record
- Java: JDBC.

Gli accessi effettuati dalla Web Application scritta in Ruby on Rails sono effettuati mediante Active Record, un modulo specifico di Rails per il mapping del database come classi/oggetti. Per quanto riguarda invece l'accesso da parte dei moduli sviluppati in Java abbiamo ritenuto opportuno utilizzare il noto connettore JDBC.

Cache: Memcached

Durante l'attività di go-live dell'applicazione ci siamo resi conto che molte richieste effettuate dagli utenti erano simili o addirittura uguali. Inoltre, data la natura del nostro servizio, la maggior parte del tempo di attesa degli utenti è incentrato sul tempo di risposta dei servizi esterni. La soluzione identificata per abbattere questa latenza è

stata l'introduzione di un livello di cache. Abbiamo utilizzato il protocollo Memcached in quanto è ampiamente diffuso e utilizzato su molti sistemi di Cloud quali Amazon. Anche questo argomento verrà trattato con maggior dettaglio sia nel Capitolo relativo ai servizi che nel Capitolo dedicato al Cloud Computing.

3.2 Perché Ruby on Rails

Tutto ciò che verrà scritto in questo paragrafo non vuole essere un confronto su i due sistemi di sviluppo backend, ma tratta semplicemente della nostra esperienza con il framework Ruby on Rails e i vantaggi che ha portato per soddisfare le nostre esigenze di sviluppo. Siamo venuti a conoscenza del framework Ruby on Rails durante un seminario tenuto da un Evangelist IBM al Politecnico di Milano. Nel seminario sono state enfatizzate tutte le caratteristiche potenzialità di Rails unito al linguaggio di scripting Ruby. I driver di decisione che ci hanno portato a tale scelta per la realizzazione del prototipo sono i seguenti (che sono anche le fondamenta su cui si basa il framework stesso):

- **DRY**(Don't Repeat Yourself) ovvero cercare di avere in un unico punto della propria architettura lo spezzone di codice che tratta una specifica funzionalità/configurazione
- **Convention over configuration** con dispendio minimo di tempi dovuti a varie configurazioni è possibile avviare un'applicazione funzionante in pochi minuti
- **Backed in testing**
- **Minimal code with maximum effect** autoesplicativo. Sfrutta enormemente la potenza di scripting data dal linguaggio Ruby

Per completare la panoramica sul framework è doveroso sottolineare i seguenti aspetti:

- Web Application framework scritto in Ruby
- Ruby è un linguaggio di scripting Object Oriented
- MVC
- Database agnostic

- Open source

Ruby on Rails, per tutte le motivazioni esposte precedentemente, ha rappresentato per CoolTraveling uno strumento fondamentale per il suo successo.

3.3 Ambiente di sviluppo

L'ambiente ideale per lo sviluppo di applicazioni Ruby on Rails è sicuramente Linux. Dopo aver scelto una distribuzione fornita dal nostro cloud provider, abbiamo poi configurato quest'ultima sulla base delle nostre esigenze. Dovendo sviluppare una applicazione web ed avendo familiarità con Apache (come Web Server) e MySQL (come DMS) abbiamo configurato una piattaforma LAMP[2] sulla distribuzione di lavoro.

3.3.1 Apache e Passenger

Come verrà ampliamente descritto successivamente all'interno del capitolo relativo al Cloud Computing, una delle difficoltà incontrate per la ricerca del nostro provider è stata proprio l'impossibilità da parte di molti di gestire l'hosting di progetti sviluppati con Ruby on Rails[3]. Sin dall'inizio abbiamo testato tutti gli Application Server e i Proxy Web Server compatibili con Rails.

Dopo aver provato gran parte delle configurazioni, abbiamo identificato due soluzioni ideali:

- **Testing in locale**: Mongrel
- **Produzione**: modrails/Phusion Passenger e Apache2
- **Produzione Java**: JRuby e Glassfish[4]

[2]LAMP è un acronimo che indica una piattaforma per lo sviluppo di applicazioni web che prende il nome dalle iniziali delle componenti software con cui è realizzata: GNU/**L**inux (sistema operativo), **A**pache (Web Server), **M**ySQL (DMS), **P**erl/PHP e/o Python (Linguaggi di scripting)

[3]Si ricorda che il panorama sia italiano che internazionale di 3 anni vedeva in modo ben differente l'adozione di Ruby on Rails come valida alternativa a Java

[4]Questa configurazione è stata la prima che abbiamo utilizzato in produzione prima di hostare l'intero progetto su Amazon

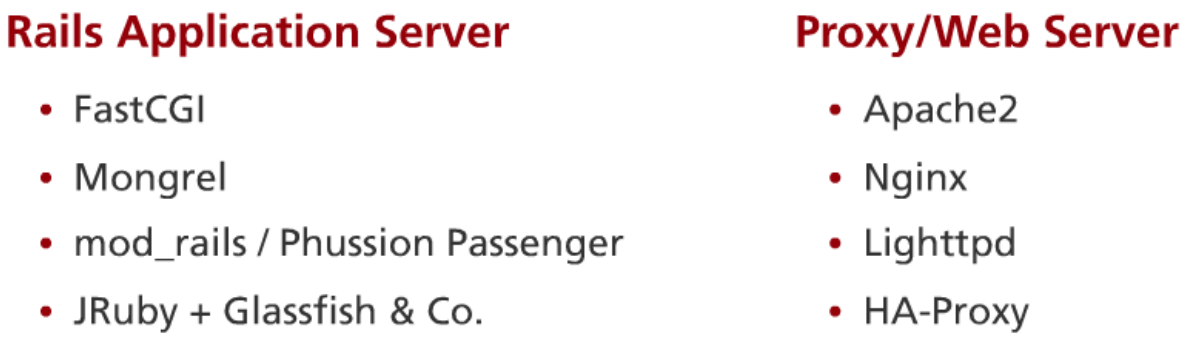


Figura 3.4: Rails Application Server e Proxy/Web Server Compatibili con Ruby on Rails

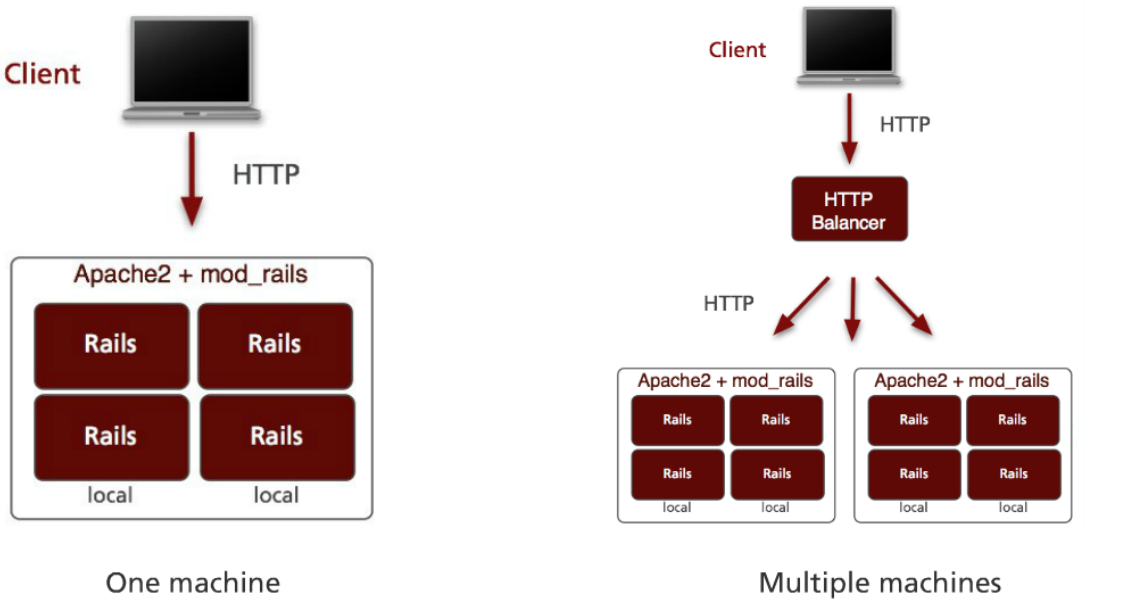


Figura 3.5: Configurazione modrails/Phusion Passenger e Apache2

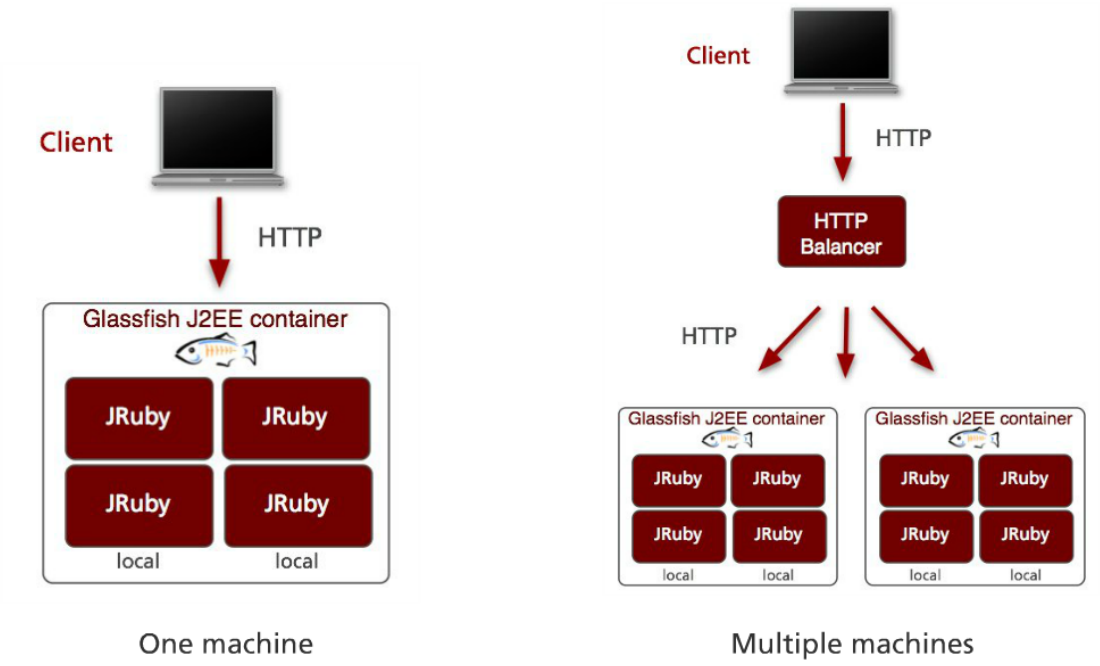


Figura 3.6: Configurazione JRuby e Glassfish

3.3.2 Ruby Gems utilizzate

- datanoise-actionwebservice (2.3.2)
- json (1.6.5)
- mysql (2.8.1)
- rails (2.3.11, 2.3.10, 2.3.2)
- rjb (1.3.9)

3.3.3 Integrazione Java in Ruby on Rails

Come verrà ampiamente spiegato nel capitolo 5 relativo ai Servizi, è stato necessario integrare alcune classi Java in Ruby on Rails. Attualmente esistono diverse modalità di integrazione, noi abbiamo utilizzato:

- JRuby
- RJB

JRuby

JRuby è una Ruby virtual machine scritta in Ruby che può essere istanziata nella JVM. Uno dei principali vantaggi è la possibilità di referenziare delle classi in Java dall'enviroment Ruby. Questa feature da' quindi la possibilità al programmatore di scrivere del codice in Ruby che si basa su librerie Java, di conseguenza è possibile l'utilizzo di oggetti Java come se fossero nativi Ruby. Inoltre la presenza di Java permette l'integrazione di diverse feature non presenti in Ruby, come ad esempio Transaction Management, JMS, JMX, che rendono quindi Ruby on Rails più enterprise.

RJB

RJB (Ruby Java Bridge) [5] è un bridge che usa le Java Native Interface (JNI). Le JNI sono usate principalmente per l'integrazione tra C e Java, si tratta di API ad alte performance. Le performance di RJB sono buone e il tool è in continuo sviluppo. I vantaggi nell'utilizzo sono molteplici:

- è leggero, richiede solo poche librerie e si riesce ad installare in poco tempo

- è semplice da usare: dopo qualche riga di configurazione si accede alle classi e ai metodi Java come se fossero nativi Ruby

L'unico svantaggio è non possedere un protocollo bidirezionale Java - Ruby, di conseguenza non sarà possibile inserire del codice Ruby in classi Java.

Nella prima fase del progetto la nostra scelta era ricaduta su JRuby soprattutto perché, come già detto in precedenza, era stato difficile trovare un hosting che ospitasse Ruby on Rails nativo. Purtroppo però le performance di JRuby non ci hanno convinto a causa dell'enorme utilizzo di memoria RAM e di lentezza nell'avvio e nell'esecuzione di alcune classi. In seguito abbiamo percorso la strada di RJB che si è rivelata essere la migliore. Uno dei principali fattori di scelta è stata la velocità e facilità d'integrazione (vedi immagine 3.7). Inoltre le performance sono decisamente migliorate e l'occupazione di memoria RAM è diminuita in maniera considerevole.

[5] Ulteriori dettagli possono essere reperiti sul sito ufficiale: http://rjb.rubyforge.org/

```
Rjb::load(classpath='./lib/j2RClass/antlr-runtime-3.2.jar:./lib/j2RClass/wrapperTrenitalia.jar')
@@data=Rjb::import('java.util.Date')
Rjb::import('java.util.ArrayList')
@@filestream = Rjb::import('weather.WeatherWrapper')
@weather=@@filestream.new(city)
@weatherDays = @weather.getAllDay()
```

Figura 3.7: Frammento di codice Java in Ruby

3.3.4 Da WSDL a REST

Inizialmente tutta la nostra architettura, dal punto di vista del protocollo di comunicazione si basava su WSDL[6]. La scelta è derivata dal fatto che l'utilizzo di un protocollo SOAP sarebbe stato perfetto per questa tipologia di servizio e affine al nostro background. Durante lo sviluppo, più precisamente a pochi giorni dalla consegna del progetto per il concorso, ci è stato consegnato il cellulare Vodafone H1, vinto nella precedente competizione WHYMCA. Dopo i primi test abbiamo subito capito che qualcosa non andava e che il Vodafone H1 era il principale indiziato per il malfunzionamento. Purtroppo la scoperta è stata fatta il giorno stesso della consegna del device con relativa platea Vodafone desiderosa di vedere la nostra applicazione up and running sul loro gioiellino.
Ogni tentativo di risolvere il problema live è stato vano e dopo un'attenta analisi portata avanti nelle ore successive abbiamo identificato la falla nella mancata gestione da parte del cellulare Vodafone H1 con piattaforma Vodafone360 del protocollo di comunicazione SOAP. Dovendo partecipare al concorso e sapendo che il test da parte della giuria sarebbe poi stato condotto su questa tipologia di device abbiamo ritenuto opportuno migrare istantaneamente il protocollo di comunicazione passando da WSDL a REST. Dopo circa un paio di giorni di lavoro, in perfetto timing con la consegna del progetto siamo riusciti a rimappare tutti i servizi mediante il paradigma REST[7].
Nei giorni successivi alla pubblicazione della nostra App abbiamo prontamente contattato il team di sviluppo della piattaforma Vodafone360 informandoli dell'incompatibilità del loro prodotto con interfacce WSDL su protocollo SOAP. Il team di sviluppo ha accolto con grande interesse la nostra segnalazione ripromettendosi di risolvere questa mancanza al più presto. Purtroppo la prematura chiusura del progetto Vodafone360

[6]Il Web Services Description Language (WSDL) è un linguaggio formale in formato XML utilizzato per la creazione di documenti per la descrizione di Web Service.

[7]REST: Representational State Transfer è un paradigma per la realizzazione di applicazioni web che incarna al suo interno i concetti di GET, POST, PUT, DELETE del protocollo HTTP

non ha sicuramente incentivato la risoluzione di molti bug noti della piattaforma, uno tra tutti quello da noi segnalato.

3.3.5 Performance

In questa sezione sono descritte le performance ottenute dal servizio a fronte dei test effettuati gestendo lo storico delle richieste su database o su un layer di cache apposito come Amazon Elastic Cache. Nella prima colonna della tabella possiamo notare i dati relativi alla gestione delle richieste utente su database. Nella seconda colonna della tabella vengono invece elencate le tempistiche ottenute grazie all'introduzione della cache a supporto del database. Come è possibile notare la cache è in grado di ridurre notevolmente i tempi di attesa dovuti alle latenze delle query effettuate su database riducendo più del 60% i tempi di attesa. Ricordiamo inoltre che nella versione di CoolTraveling le richieste venivano gestite tramite database. Solo in un secondo momento, anche grazie a strumenti messi a disposizione da Amazon, siamo stati in grado di gestire in modo efficiente e indipendente un livello di cache distribuito fruibile da più istanze contemporaneamente. Nel primo caso si può notare come le tempistiche della cache siano peggiori rispetto a quelle del database. La motivazione è che la prima richiesta che viene effettuata non può per definizione sfruttare alcuna ottimizzazione dovuta da dati presenti nello storico, quindi il tempo di attesa è strettamente legato ai servizi esterni. Il dato in questione è quindi indipendente dal servizio di gestione delle ricerche precedenti adottato.

3.4 MVC: Model

La parte dei modelli è generata in Rails e scritta in Ruby. Ogni modello è direttamente mappato su DB relazionale che controlla consistenza e persistenza dei dati rispetto al modello.

3.4.1 Il modello logico dei dati

Il modello logico dei dati raffigurato nel database descritto successivamente è in grado di gestire le ricerche effettuate dagli utenti e mappare un viaggio completo con servizi di trasporto e servizi accessori. Quando un utente si iscrive al servizio viene automaticamente aggiunto nella tabella *Users*. Successivamente ogni sua richiesta

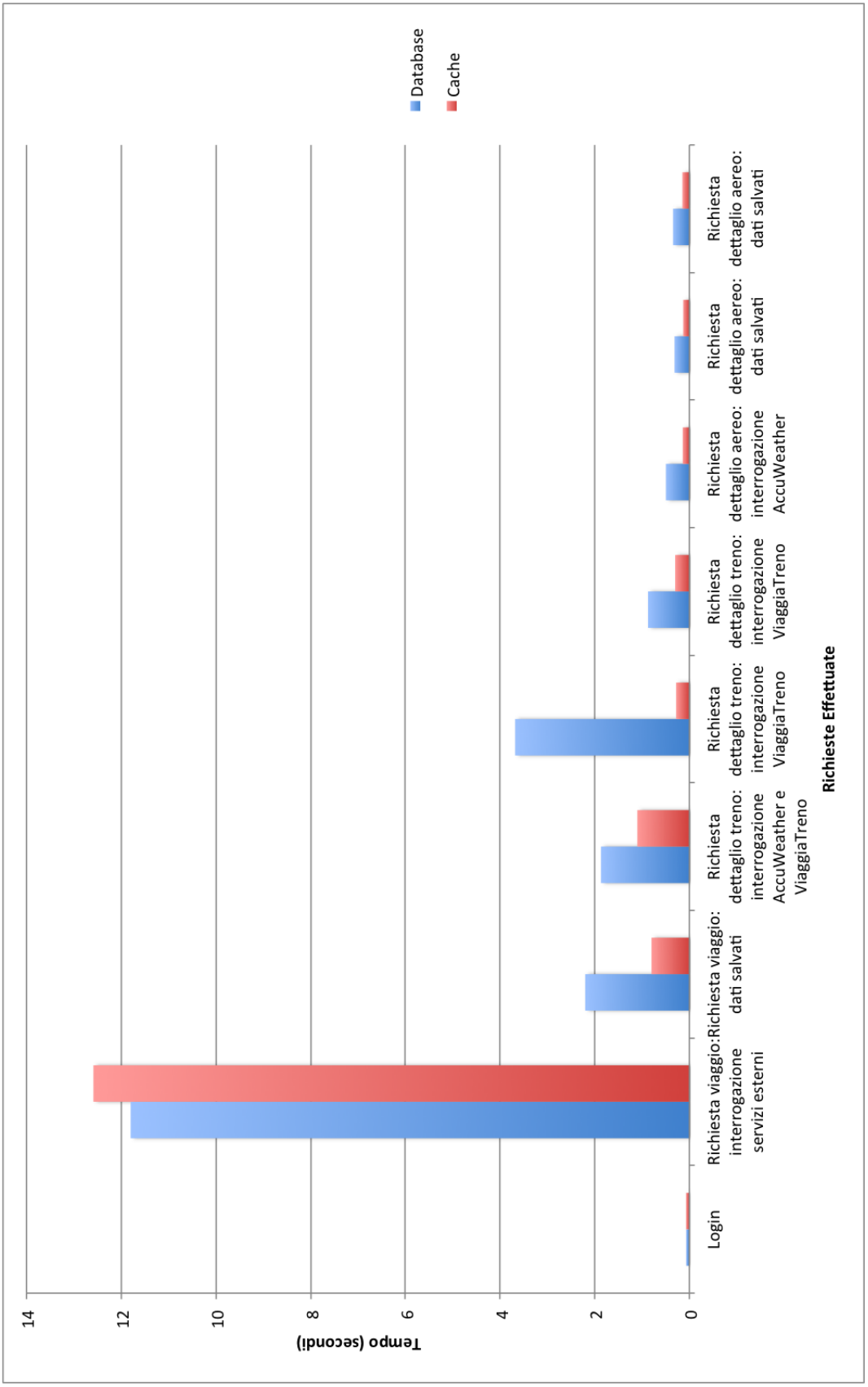


Figura 3.8: Grafico performance: Database vs Cache

effettuata sarà mappata nella tabella *Requests*. Da ogni richiesta viene generata una nuova tupla nella tabella *Routes*. Per ogni *Route* vengono generate n tuple nella tabella *Trips* a seconda delle tratte trovate. Ogni *Trip* può avere un dettaglio di due diverse tipologie:

- Flight_details: dettagli legati alla soluzione di viaggio aereo
- Train_details: dettagli legati alla soluzione di viaggio treno.

Questi dettagli sono raggiungibili tramite chiavi esterne alle tabelle di treni e aerei definite precedentemente.
Esistono inoltre delle tabelle di supporto che contengono informazioni utili al calcolo dei viaggi e sono le seguenti:

- Airportitalies: contenente tutti i dettagli relativi agli aeroporti italiani, comprese latitudine e longitudine
- Bedandbreakfasts: contenente tutti i B&B italiani mappati da bed-and-breakfast.it.
- Coordinateitalias: contenente i dettagli delle coordinate geografiche di tutti i comuni italiani
- Trainitalies: contenente i dettagli di tutte le principali stazioni ferroviarie italiane

Grazie a questa progettazione del modello logico, l'impatto legato all'introduzione di nuovi servizi è davvero minimale. Sarà infatti necessario generare solamente la tabella relativa ai dettagli della tratta con il nuovo mezzo introdotto. Anche se non attualmente implementato, la struttura è stata pensata in modo da valutare la futura introduzione di una nuova feature: **viaggi con tratte multiple su servizi misti**. In questo modo è possibile proporre soluzioni di viaggio complesse a partire da esigenze specifiche di ogni singolo utente. Sarà infatti possibile percorrere la tratta Los Angeles - Cagliari in due modi differenti:

- Tratta Aerea: Los Angeles - Roma e Roma - Cagliari
- Tratta Aerea: Los Angeles - Milano e Tratta Treno: Milano - Roma e Tratta Traghetto: Roma - Cagliari.

3.4.2 Progettazione Concettuale

Di seguito è presentato lo schema ER e successivamente una rappresentazione di tutti gli attributi presenti nelle varie entità.

3.4.3 Tabelle relazionate

Le seguenti tabelle permettono di gestire tutte le richieste degli utenti e i relativi dettagli.

Users

- *Descrizione:* Tabella contenente tutti gli utenti registrati al servizio e i relativi dati.
- *Chiave primaria:* id

Sessions

- *Descrizione:* Tabella contente tutte le connessioni attive nel database nell'ora corrente. È attiva una stored procedure che ogni ora ripulisce le sessioni scadute spostandole nel db history.
- *Chiave primaria:* id
- *Chiave esterna:* user_id

Requests

- *Descrizione:* Tabella contenente tutte le richieste della giornata. È attiva una stored procedure che provvede ogni giorno a spostare le richieste vecchie nel db history.
- *Chiave primaria:* id
- *Chiave esterna:* user_id

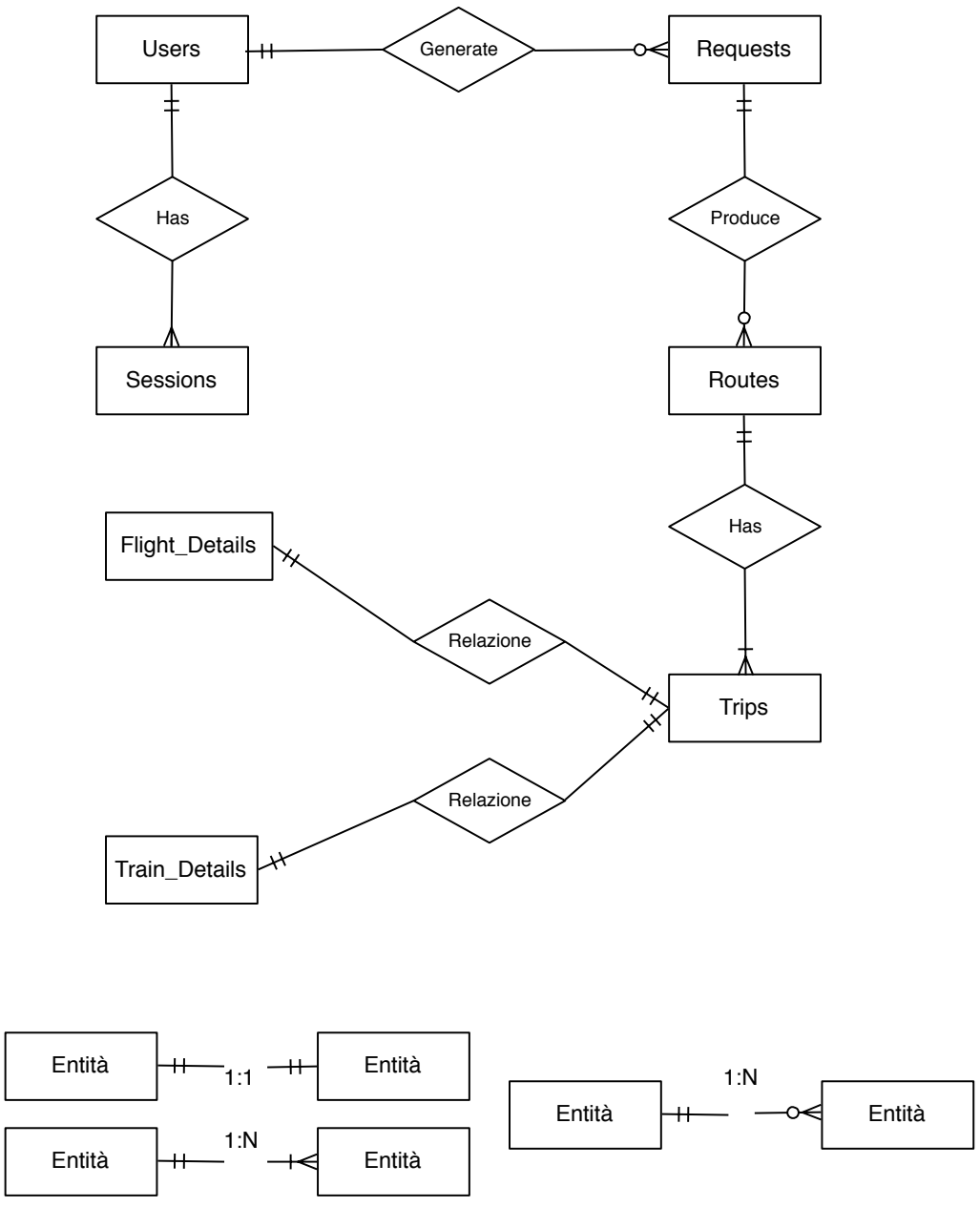


Figura 3.9: Diagramma Entità - Relazione

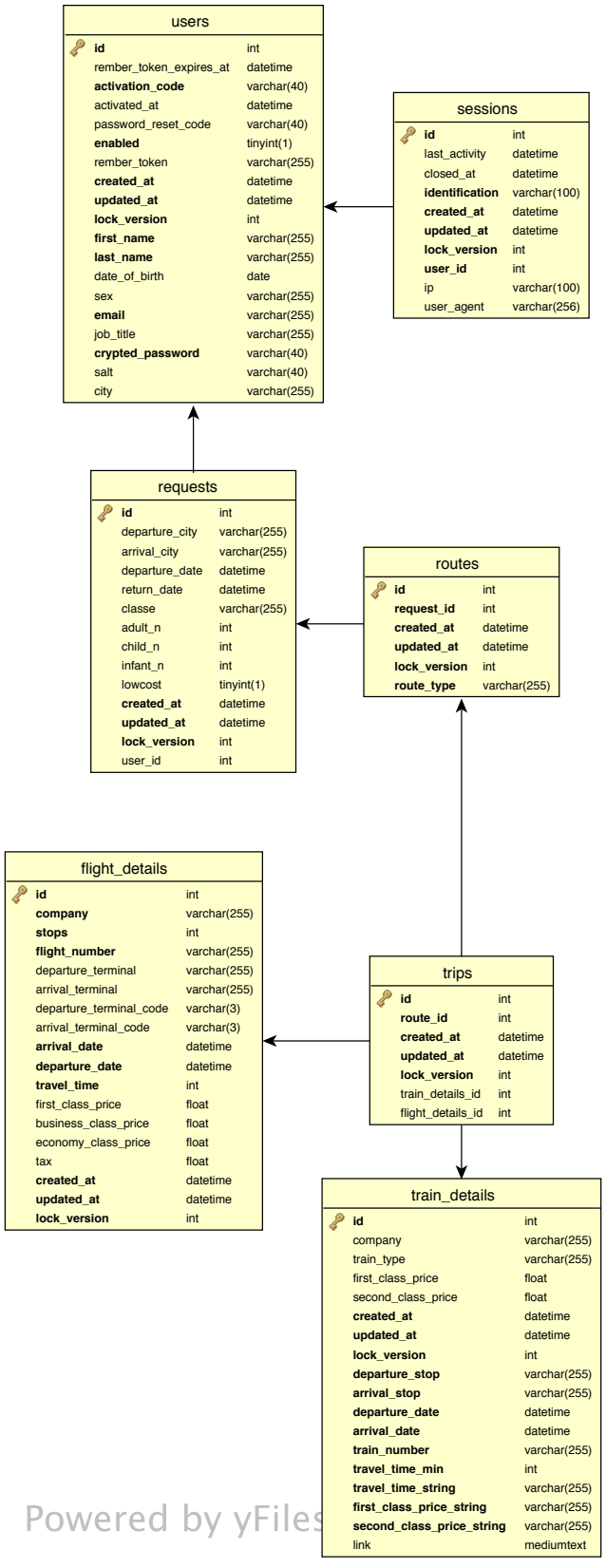


Figura 3.10: MySQL Entity Diagram - 1

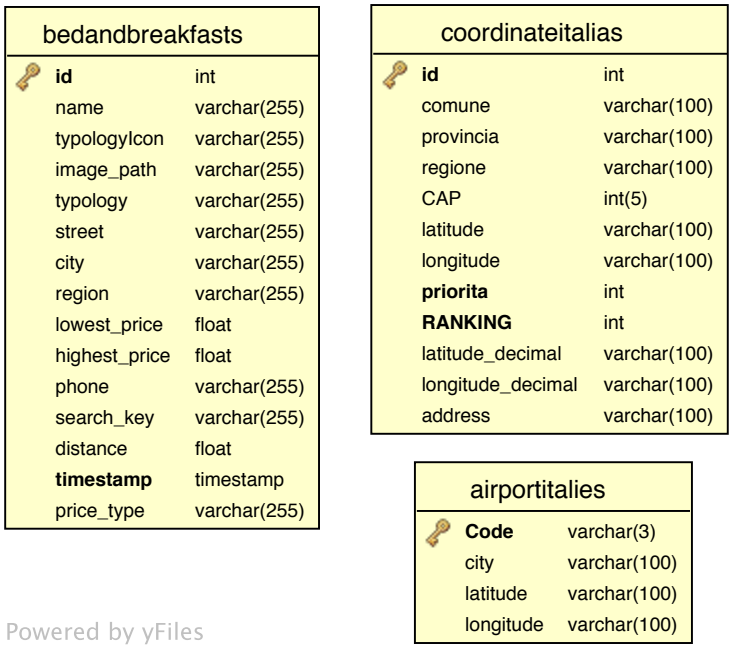


Figura 3.11: MySQL Entity Diagram - 2

Routes

- *Descrizione:* Tabella che contiene tutte le routes generate dalle varie request. Una route identifica una possibile soluzione di viaggio.
- *Chiave primaria:* id
- *Chiave esterna:* request_id

Trips

- *Descrizione:* Tabella che contiene tutti i trips generati dalle routes. Se esistono più trip per una routes significa che questa soluzione di viaggio prevede cambi.
- *Chiave primaria:* id
- *Chiave esterna:* route_id , train_details_id , flight_details_id

Flight_details

- *Descrizione:* Tabella contenente tutte le informazioni sui voli generati dai trips relativi a soluzioni di volo.

- *Chiave primaria:* id

Train_details

- *Descrizione:* Tabella contenente tutte le informazioni sui treni generati dai trips relativi a soluzioni di treno.
- *Chiave primaria:* id

3.4.4 Elenco tabelle non relazionate

BedAndBreakfasts

- *Descrizione:* Contiene il riferimento a tutti i Bed and Breakfast italiani usati.
- *Chiave primaria:* id

CoordinateItaly

- *Descrizione:* Tabella di riferimento per tutti i comuni italiani.
- *Chiave primaria:* id

AirportItaly

- *Descrizione:* Tabella di riferimento per gli aeroporti italiani.
- *Chiave primaria:* id

TrainItaly

- *Descrizione:* Tabella di riferimento per le stazioni italiane.
- *Chiave primaria:* id

Database History

Il database History viene utilizzato per evitare di sovraccaricare il database principale. In questo nuovo database vengono ogni ora spostate le sessioni scadute e ogni giorno le richieste vecchie. Ha la stessa struttura del database principale e sfrutta due stored procedure migrare: richieste e sessioni.

3.5 MVC: View

I dati una volta elaborati e aggregati devono essere resi disponibili ai client che interrogano la WebApplication.

3.5.1 Web Tier

Il primo step necessario è legato all'interpretazione delle richieste utente pervenute alla WebApplication. Una volta interpretate le richieste devono invocare i relativi comandi sulla business logic della WebApplication consentendo l'elaborazione del comando ricevuto. A questo punto i dati, dopo esser stati estratti, dovranno essere rielaborati per essere forniti all'utente in modo a lui chiaro e comprensibile. Nel nostro caso non esiste un vero e proprio layer dedicato al view, del MVC. Come vedremo le pagine web non vengono create dinamicamente dalla WebApplication ma sono gestite autonomamente dai client (a meno dei dati).

3.5.2 Client Tier

Il client è il mezzo di comunicazione ed interazione che gli utenti hanno con la WebApplication. Tramite il client è possibile effettuare le richieste e visualizzare le risposte. I comandi utenti, eseguiti sull'interfaccia client della WebApplication, vengono tradotti in chiamate Rest / SOAP che verranno interpretate ed eseguite. Queste ultime consentiranno di poter recuperare i dati necessari da riproporre al client.

3.6 MVC: Controller

Il controller racchiude il cuore della logica di business della WebApplication. È proprio in queste classi che vengono esaminati ed elaborati i dati provenienti dal Model al fine di presentarli correttamente al View layer.

Esiste un controller dedicato per ogni model mediante il quale e' possibile gestire al meglio le sinergie che si vengono a creare in una struttura complessa. Inoltre abbiamo predisposto un controller per ogni accesso ai webservice, in particolare (come visto precedentemente), avremo:

- **UserManagerController:**

  - *login:* Gestisce la procedura di login da parte degli utenti, andando a controllare nel database l'esistenza o meno della loro utenza.
  - *registrazione:* Gestisce la procedura di registrazione, inviando una mail per l'attivazione dell'account appena creato.
  - *inviaCodiceAttivazione:* Consente di inviare il codice di attivazione, sopra citato, all'utente in modo da prevenire la creazione di utenti non realmente associati ad un indirizzo di mail valido.
  - *recuperaPassword:* Gestisce la procedura le casistiche di smarrimento password, consentendo l'inserimento di una nuova.

- **InputHelperController:**
  - *valida:* Gestisce la validazione delle città. Ovvero verifica nel database dei comuni italiani, l'input dell'utente. Nel caso in cui non vi sia un match esatto sono proposti all'utente tutti i risultati simili trovati.

- **CoolTravelingController**
  - *richiestaViaggio:* Questo è il controller che gestisce una request proveniente dall'utente, la interpreta e prepara le richieste da effettuare in parallelo ai vari servizi di viaggio supportati (treno e aereo). Una volta ricevute entrambe le risposte impacchetta tutto e invia la risposta contenente tutte le soluzioni di viaggio trovate ordinate tra loro. Questa volta per ottimizzare la connessione e il traffico dati sono inviati solo i dati indispensabili alla visualizzazione delle tratte, non i dettagli approfonditi di ogni singola soluzione.
  - *richiestaViaggioCompleto:* La richiesta viaggio completo consente di recuperare i dettagli approfonditi di ogni soluzione trovata. In questo modo si evita di inviare inutilmente una grande mole di informazioni non richieste dagli utenti. Questo comando ovviamente deve essere lanciato successivamente alla richiestaViaggio, e deve puntare ad una soluzione viaggio realmente trovata nella ricerca.
  - *richiestaBBVicini:* La richiesta Bed and Breakfast vicini consente di trovare tutti i Bed and Breakfast vicini ad una città. In questo modo è possibile vi-

sualizzare agli utenti un elenco di accommodation possibili, inviando sempre una mole ridotta di dati.

- *richiestaBB:* Questo metodo inoltra la richiesta di Bed and Breakfast al database quando già si conosce l'id esatto da recuperare. Questa volta visto che l'utente ha già letto le informazioni generali, saranno inviate tutte le informazioni dettagliate relative al Bed and Breakfast, ottimizzando così i dati inviati.
- *richiestaWeather* La richiesta Weather gestisce il meteo relativo ad una città per tutta la settimana all'interno della quale la richiesta è effettuata. Non vengono inoltrate richieste meteo se il periodo del quale si stanno cercando le previsioni supera 7 giorni, in quanto non e' garantita l'attendibilità dei dati.

Capitolo 4

Cloud Computing

In questo capitolo verrà ampliamente discusso e trattato nei minimi dettagli il concetto di Cloud Computing applicato ad un caso reale: dalla sua concezione alla scelta di un sistema in grado di garantire scalabilità, performance e riduzioni costi difficilmente raggiungibili con le tecnologie tradizionali.

4.1 Introduzione

Con il termine cloud computing ci si riferisce a tutte quelle tecnologie che abilitano all'utilizzo di servizi e risorse virtuali messe a disposizione dal fornitore del servizio. I servizi e le risorse che tipicamente vengono offerti da chi fa del cloud computing il proprio core business sono molteplici e diversificate, ma condividono l'approccio e l'essenza ovvero la fruizione di servizi e risorse virtuali e non fisiche. Il cloud computing in estrema sisntesi permette quindi, mediante un semplice browser, di gestire e lavorare su computer reali cablati nella nuvola. Esaminando a fondo il prodotto ci si accorge come il cloud computing non sia altro che un aggregatore di risorse che rese virtuali possono essere meglio gestite e sfruttate da una platea meno tecnica ed esperta di quanto sarebbe invece necessario.

Eistono diverse soluzioni di cloud computing attualmente in commercio:

- SaaS (Software as a Service):

  Permette l'utilizzo di software applicativi mediante un accesso remoto che spesso può essere effettuato mediante un semplice browser. Questa semplicità di fruizio-

ne seguita dalla dirompente esplosione del web ha fatto si che le tecnologie SaaS abbiamo riscosso un grande successo, diventando subito molto efficaci e popolari.

- PaaS (Platform as a Service)

  Concettualmente incarna tutti i principi del SaaS e li estende fornendo ai propri utilizzatori non solo un semplice software ma un'intera piattaforma completamente virtualizzata. Esempi... Un esempio di Platform as a Service è ad esempio Office365.

- IaaS (Infrastructure as a Service)

  Con questo acronimo venono identificati i servizi che offrono esplicitamente l'utilizzo di hardware in remoto. Gli utenti possono quindi, con delle semplici richieste web, configurare e perfezionare le specifiche dell'hardware utilizzato.

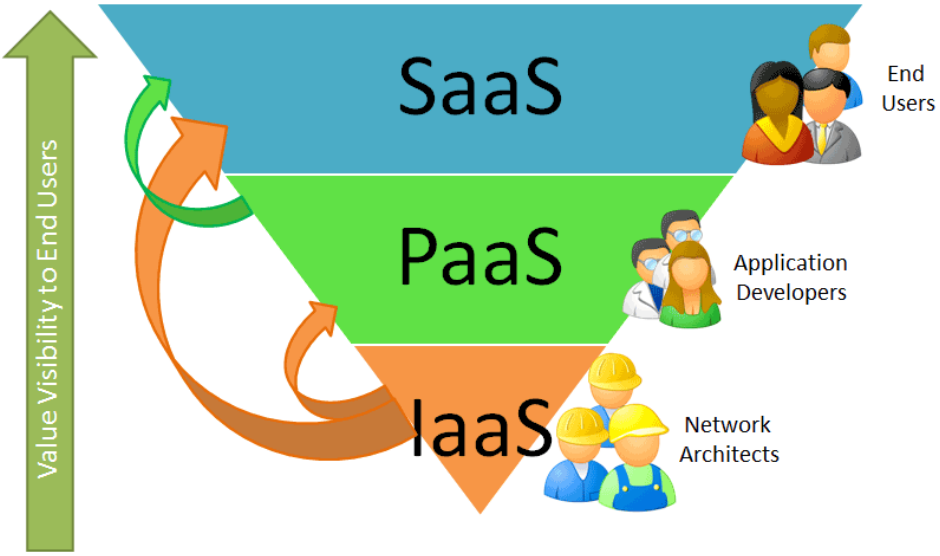


Figura 4.1: Rappresentazione grafica delle diverse tipologie di servizi cloud presenti sul mercato.

In tutte queste configurazioni è sempre presente la figura del cloud provider ovvero di chi offre il servizio, fornendo server virtuali, storage o intere applicazioni mediante un modello "pay-per-use". I clienti non acquistano un prodotto ma ne

pagano per il suo utilizzo. Vedremo poi come i modelli di business, soprattutto quello adottato da Amazon, siano spesso legati proprio ai volumi di consumo del servizio, valorizzando ogni singola risorsa che possa essere consumata dal cliente.

4.2 Il Mercato: Top 5 Cloud Computing Provider

4.2.1 Amazon

Figura 4.2: Amazon Web Services.

4.2.2 Microsoft Azure

Figura 4.3: Microsoft Azure.

4.2.3 Google App Engine

4.2.4 Rackspace

4.2.5 Seeweb

Seeweb è tra le prime realtà di hosting in Italia che ha iniziato ad offrire servizi in Cloud accompagnati dalla consueta professionalità e affidabilità dei servizi Seeweb.

Nel panorama cloud i servizi offerti sono i seguenti:

- Cloud Hosting: Il servizio base per l'ingresso nel mondo del cloud. Nonostante si tratti di un servizio entry-level gli utenti hanno a disposizione un servizio di hosting di tutto rispetto che compete alla grande con i principali fornitori del

Figura 4.4: Google App Engine

Figura 4.5: Rackspace

Figura 4.6: Seeweb

settore. Sono infatti disponibili configurazioni per i principali Application Server come Tomcat, JBoss, Zope e ambienti protetti per l'uso di Ruby On Rails. Inoltre nel canone annuo del servizio sono già compresi i servizi a consumo tipici di questa tipologia di servizio come storage, banda minima garantita e traffico, senza dover quindi incappare in spiacevoli sorprese causate dai volumi generati.

- Cloud Server: Il cloud server è un prodotto basato interamente sul cloud hosting che mette a disposizione degli utenti delle configurazioni aggiuntive quali la gestione del firewall dei file di log personalizzati oltre a tante altre funzionalità di livello enterprise.

- Cloud Appliance: Il cloud Appliance è un nuovo servizio nato per la gestione semplificata delle reti VPN. Un'ottimo prodotto da affiancare alle tipiche offerte cloud per gestire in sicurezza le comunicazioni con le farm in cloud remote.

- Cloud Object Storage: Servizio di storace che permette la gestione di grandi quantità di dati online nel cloud grazie ad una semplice interfaccia web.

- Cloud Streaming: Piattaforma di streaming video compatibile con la maggior parte delle piattaforme in commercio, mobile comprese.

4.3 Amazon

4.3.1 I prodotti

Per brevità e chiarezza nell'analisi elencheremo solo i principali prodotti offerti da Amazon sul panorama Cloud:

- *Amazon Elastic Compute Cloud (EC2)*: è il cuore di Amazon AWS. Dal pannello di controllo EC2 è infatti possibile, con pochi semplici click, configurare potenza di calcolo nel cloud. Una soluzione davvero comoda ed immediata, che messa a disposizione dei developer consente loro di gestire i propri servizi in maniera facile veloce e scalabile.
  Lavorare con Amazon EC2 equivale ad avere a portata di mano una web farm virtuale dove poter creare nuove macchine, denominate istanze nel mondo Amazon, configurarle e avviarle ed effettuare il deploy delle proprie App in pochi istanti.

- *Amazon Elastic MapReduce*: Con il termine Elastic MapReduce Amazon identifica una particolare tecnologia, anch'essa fruibile tramite web service, che facilita le analisi su grandi moli di dati. Le applicazioni possono quindi essere molteplici, dal data mining al data warehousing sino all'analisi di complessi file di log. Sostanzialmente l'Elastic Map Reduce è utile in tutte le analisi che implicano l'uso di un set di dati.

- *Auto Scaling*: è una delle funzionalità del cloud di Amazon più straordinarie ed utili al fine di gestire in modo ottimale prestazioni e costi della propria architettura sul cloud. Grazie a questo servizio è infatti possibile far aumentare o diminuire il numero di istanze dedicate ad un progetto in modo del tutto automatico. Sarà necessario impostare solamente nelle apposite configurazioni del servizio le condizioni che abilitano l'upgrade o il downgrade della farm e automaticamente Amazon farà tutto per noi. L'Auto Scaling si basa fortemente sul servizio di Cloud Watch, dove è possibile configurare sofisticatissimi sistemi di alert e analisi delle istanze attive. Con un po' di esperienza e test è possibile realizzare un'architettura completamente scalabile in modo automatico. Ovviamente l'occhio umano è sempre indispensabile per gestire eccezioni non risolvibili in modo automatico. Bisogna inoltre tener sempre presente che l'avvio di nuove istanze comporta un diretto impatto sul costo della farm virtuale, quindi è sempre bene tener monitorate le attività automatiche impostate.

- *Elastic Load Balancing*: è il load balancer pensato da Amazon per il cloud. Come si può intuire dal nome è in grado di adattarsi in funzione delle istanze che sono ad esso associate. La funzione tipica di un load balancer è smistare il traffico tra le varie macchine associate. Visto che nel cloud di Amazon queste istanze possono aumentare o diminuire in modo automatico, come abbiamo visto precedentemente, è indispensabile avere un servizio di load balancing che supporti questa funzionalità. L'Elastic Load Balancing è in grado con piccole configurazioni di scovare eventuali istanze non funzionanti per i motivi più svariati, down della macchina, down della region associata, ecc... ed escluderle quindi dal servizio in modo automatico. Successivamente se una serie positiva di check sarà riscontrata, le istanze ritenute a rischio saranno reintegrate nel servizio. L'Elastic Load Balancing è un servizio indispensabile per garantire scalabilità e affidabilità della propria architettura.

- *Amazon Route 53*: è un Domain Name System ad alte prestazioni che Amazon offre ai propri clienti. Il servizio si integra ovviamente alla perfezione con i restanti servizi del cloud di Amazon, rendendo quindi facile ed immediata l'integrazione garantendo inoltre un tempo di risposta a bassissima latenza per i servizi interni.

- *Amazon Virtual Private Cloud (VPC)*: permette di gestire nel cloud di Amazon una vera e propria area riservata. Una sorta di rete privata dove poter configurare la topologia, la scelta degli ip, delle tabelle di routing, ecc.. In questo modo sarà possibile configurare al meglio i propri servizi e macchine associate suddividendo la rete in sottoreti e regolando gli accessi con semplici regole. Per garantire un livello di sicurezza di tutto rispetto il VPC supporta senza problemi l'utilizzo di Hardware Virtual Private Network (VPN). In questo modo sarà quindi possibile collegare in modo sicuro datacenter dislocati su altre farm piuttosto che semplici servizi esterni che devono viaggiare su connessioni sicure.

- *Amazon Relational Database Service (RDS)*: è un servizio che permette ai clienti Amazon di configurare un database relazionale altamente scalabile in pochi semplici click, grazie ad un'interfaccia web molto intuitiva. Quando si configura un database non si fa altro che configurare un'istanza configurata ed ottimizzata precedentemente da Amazon per avere alte prestazioni con i database relazionali. Vengono supportati senza problemi i principali database presenti in commercio come MySQL o ORACLE. La differenza tra l'utilizzo di un'istanza RDS e di una semplice istanza EC2 per la gestione di un database è fondamentale nella scelta della propria architettura. Amazon RDS introduce un sostanziale extra costo che viene ripagato nel tempo in quanto molti processi verranno automatizzati o resi scalabili in modo automatico. Utilizzando un'istanza RDS si hanno i seguenti benefici:

  - Scalare le dimensioni dell'istanza database in modo automatico
  - Scalare le storage disponibile indipendentemente dall'istanza
  - Avere backup automatizzati di default
  - Poter ripristinare il DB da snapshot effettuati periodicamente da Amazon in modo automatico
  - Sfruttare l'ottimizzazione di un'istanza sviluppata in collaborazione con i produttori delle piattaforme di database più famose.

  - Monitorare tutti i parametri di funzionamento sia dell'istanza che del database
  - Poter impostare alert automatizzati con Cloud Watch legati a parametri propri dei database
  - Database replicati su singola region o anche su più region, per garantire estrema affidabilità.

- *Amazon DynamoDB*: è l'approccio Amazon a NoSQL database, mantenendo tutti i vantaggi e le proprietà di scalabilità descritte precedentemente.

- *Amazon SimpleDB*: è l'approccio Amazon ai database non relazionali sgravando sempre l'utente dai normali compiti di database administration.

- *Amazon CloudWatch*: è il servizio ufficiale per il monitoring dei prodotti AWS nel cloud. Grazie a questo servizio è possibile monitorare un grandissimo numero di parametri per ogni singolo servizio, mantenendo quindi sotto controllo l'intera operatività della nostra rete nel cloud. Amazon CloudWatch è lo strumento ideale anche per l'ottimizzazione dei propri prodotti in quanto consente di produrre metriche e analisi sia a livello di istanze EC2 che per la maggior parte dei servizi AWS.

- *Amazon Simple Notification Service (SNS)*: Amazon Simple Notification Service è un servizio che vive in stretta simbiosi con il CloudWatch, in quanto permette l'invio in modo automatico di notifiche in funzione dell'andamento dei propri prodotti su Amazon. Le tipologie di notifiche disponibili sono molteplici e vanno dalla semplice mail all'SMS. Trattandosi di un sistema a push notications gli utenti non sono costretti a monitorare costantemente i propri servizi. Basterà loro configurare correttamente il CloudWatch e in modo automatico grazie a SNS le notifiche arriveranno tempestivamente. Come è facile intuire, la potenza di questo servizio va ben oltre la semplice segnalazione di notifiche sui guasti o malfunzionamenti dei prodotti. Con una adeguata gestione può diventare un completo sistema di gestione e comunicazione di qualsiasi tipo di notifica si ritenga opportuno gestire per il proprio business.

- *Amazon ElastiCache*: è un web service che facilita il deploy, la gestione e la scalabilità di servizi che fanno un grande uso di in-memory caching system a dispetto

dei comuni sistemi basati su hard disk o database. ElastiCache è perfettamente compatibile con Memcached, sistema memory object caching tra i più famosi. Amazon rende quindi facile il passaggio ad ElastiCache per chi ha già integrato nel proprio framework Memcached. Come nella maggior parte dei servizi Amazon, anche in questo caso, l'introduzione di ElastiCache ridurrà l'effort necessario al monitoring e alla gestione del sistema di caching. Operativamente è possibile lanciare 1 terabyte di cache in pochi semplici click. Successivamente potremo configurare le nostre App dare loro tutta la potenza di ElastiCache. ElastiCache si appoggia al servizio Amazon EC2 per la generazione dei nodi contenenti la memoria RAM utilizzata per la cache. Questo è un grosso vantaggio perché ci consente di monitorare ogni noto con tutti gli strumenti messi a disposizione da Amazon Cloud Watch.

- *Amazon Simple Storage Service (S3)*: è uno strumento di storace virtuale. Ideale per la gestione dei dati nel cloud. Il caricamento e gestione dei dati è consentito sia da un'interfaccia web che mediante developer api, in modo da consentire una gestione sempre più ottimizzata e scalabile per gli sviluppatori.

- *Amazon Elastic Beanstalk*: è uno strumento che facilita e velocizza oltremodo il deploy e la gestione delle nostre applicazioni nel cloud di Amazon. Grazie ad Amazon Elastic Beanstalk, infatti, è possibile effettuare il deploy di un App semplicemente effettuando il subtim dell'App da un'interfaccia web di configurazione. Successivamente Amazon si prenderà cura, in modo automatico, della gestione dello spazio su disco necessario, eventuali load balancing, auto-scaling delle istanze e monitoraggio dello stato di servizio dell'App. Grazie a questo prodotto Amazon disaccoppia la reale necessità di affiancare alla release di semplici application nel cloud della necessità di progettare e pianificare un'adeguata infrastruttura supporto. Ovviamente qualora poi lo si ritenga opportuno è sempre possibile intervenire manualmente e correggere le impostazioni definite da Amazon per noi. Il servizio è attualmente gratuito e supporta Apache Tomcat per la gestione di App sviluppate in Java o virtualizzate in ambiente Java. Si riceverà invece normale addebito per tutti i servizi sfruttati dalle App lanciate tramite Elastic Beanstalk. Essendo un servizio altamente automatizzato e molto impattante a livello di costi, in funzione dei servizi usati, è bene monitorare adeguatamente il livello delle risorse ipotecate mensilmente.

Tutti i controlli sono disponibili sia tramite pannello web che via shell, grazie al set di comandi resi disponibili da Amazon al fine di rendere sempre più automatizzabile ogni singola mansione.

4.3.2 Il modello di business

Il modello di business adottato da Amazon incarna alla perfezione il concetto di Pay-per-Use, uno dei cavalli di battaglia di tutte le piattaforme cloud. Amazon con i prodotti offerti in cloud è riuscita a costruire un modello di business fenomenale che si adatta alla perfezione ai bisogni delle aziende. Per quasi la totalità dei prodotti è infatti possibile scegliere il modello idoneo al proprio business. Tipicamente esiste un differenziale di costo a seconda della tipologia di contratto stipulata nella richiesta effettuata:

- Risorse consumate
- Risorse prenotate
- Risorse prenotate a lungo termine.

Esiste inoltre un'insolita strategia messa in pratica da Amazon, al fine di educare la propria clientela: Pagare per le risorse riservate e non utilizzate. Nel cloud come in tutti gli altri servizi IT spesso i clienti riservano risorse/prodotti senza mai farne un reale utilizzo. Il caso più eloquente è quello degli indirizzi IP, spesso vengono acquistati in banchi considerevoli senza un reale motivo. Su Amazon sono disponibili indirizzi pubblici in modo totalmente gratuito a patto che vengano utilizzati. Indirizzi non utilizzati saranno addebitati all'incauto cliente.

4.3.3 Amazon AWS

Tutti i servizi Amazon sopra descritti sono accessibili tramite web service, da qui il nome *Amazon AWS: Amazon Web Services*. Ogni singolo servizio può essere creato, configurato e distrutto direttamente tramite API. Questo aspetto non è da sottovalutare in quanto ci ha permesso di realizzare processi di gestione automatizzati della nostra architettura altrimenti impossibili da gestire, uno su tutti: l'autoscaling.
Nel tempo Amazon ha realizzato anche un eccellente pannello web dove sono ormai fruibili quasi tutti i prodotti sopra descritti, rendendo ancora più facile ed immediata la gestione delle proprie configurazioni.

4.4 CoolTraveling on the Cloud

Il progetto CoolTraveling è nato, nella sua versione primordiale, nelle reti domestiche delle nostre case. Successivamente, in preparazione al lancio per il concorso Vodafone App Star, abbiamo iniziato a valutare i vari servizi di hosting disponibili sul mercato. La ricerca è stata lunga e complessa in quanto due anni fa le soluzioni Cloud in Italia scarseggiavano e i servizi di hosting tradizionali non garantivano la scalabilità che avevamo pensato per il nostro prodotto. Dopo una prima fase di valutazione abbiamo sottoscritto il servizio di Seeweb, all'epoca uno dei primi cloud provider italiani.

4.4.1 La scelta

Il prodotto che abbiamo selezionato è Seeweb Cloud Hosting, già descritto precedentemente. Abbiamo deciso di scegliere Seeweb per diversi fattori:

- Risorse consumate:
- Risorse prenotate:
- Risorse prenotate a lungo termine:

A differenza di Seeweb, l'offerta di Amazon è molto più articolata e complessa, quindi inizialmente abbiamo ritenuto opportuno scegliere il prodotto plug and play più semplice e facilmente configurabile. Una delle differenze più evidenti tra Seeweb ed Amazon infatti è proprio legata all'assistenza tecnica, che se nel primo caso viene fornita compresa nel presso del cloud nel secondo è invece fatturata separatamente come un servizio aggiuntivo. La nostra scelta è ricaduta sulle configurazioni base del prodotto di CloudHosting che si sono rivelate idonee per il lancio del servizio al fine della valutazione del concorso. Successivamente, dopo qualche mese di analisi e lavoro, abbiamo notato come l'offerta selezionata fosse troppo limitativa sul fronte configurazioni ed abbiamo riaperto la ricerca per effettuare una migrazione del servizio di hosting. Essenzialmente cercavamo un prodotto molto simile a quello già selezionato ma con la possibilità di configurare e gestire in completa autonomia la macchina nel cloud a noi dedicata. All'interno del panorama SeeWeb l'unico prodotto idoneo è il CloudServer, ma per via dei costi abbiamo deciso di provare i servizi di Amazon ed effettuare una comparazione dell'offerta.

Il prodotto di CloudHosting selezionato presso seeweb ha le seguenti caratteristiche:

- Spazio disco: 10 GB
- Traffico 3000 GByte/mese
- Banda minima garantita: 10 Mbps
- Ram: 2Gb
- SSH
- Costo: 240 euro/anno
- HelpDesk: 365/7/24
- Firewall: non previsto

Il prodotto selezionato presso Amazon ha le seguenti caratteristiche:

- Spazio disco: 5 GB
- Spazio S3: 30 GB
- Traffico 3000 GByte/mese
- Banda minima garantita: 10 Mbps
- Ram: 613Mb
- SSH
- Costo: Free per un anno
- HelpDesk: Non compreso
- Firewall: configurabile

Nonostante le caratteristiche tecniche inferiori i servizi Amazon sono ottimizzati in modo tale da fornire performance difficilmente raggiungibili con altri fornitori. Una delle principali differenze tra Seeweb e Amazon infatti risiede nelle istanze utilizzate. Mentre su Seeweb si ha solamente la possibilità di scelta del sistema operativo e in alcuni casi della sua versione, su Amazon si può selezionare l'immagine di qualsiasi

distribuzione si intenda caricare nel cloud. Inoltre Amazon in collaborazione con le più grandi case produttrici ha realizzato delle versioni di sistema operativo ottimizzate per il cloud.
Nel nostro caso, in entrambe le situazioni abbiamo selezionato Linux come sistema operativo. Nel primo caso, con Seeweb, la distribuzione scelta è stata l'unica all'epoca disponibile: Debian. Su Amazon, sebbene fosse sempre disponibile Debian, abbiamo preferito una distribuzione ottimizzata per il cloud di Amazon realizzata appositamente dal team di Alestic. Nel caso specifico una derivazione di Ubuntu.
Tuttavia dopo la vittoria del concorso avendo costituito una società e visto l'interesse succitato sul progetto da parte del pubblico abbiamo deciso di investire anche sulla piattaforma di cloud computing che avrebbe ospitato il nostro servizio. In questo scenario il partner ideale si è confermato essere Amazon, in quanto avrebbe messo a nostra disposizione una serie di servizi e strumenti di livello enterprise con modularità sui costi.

4.4.2 L'architettura

Il cloud di Amazon ci ha permesso di scoprire e testare numerosi prodotti, tutti descritti precedentemente riuscendo a ricreare nel cloud un'architettura difficilmente realizzabile nella realtà se non a fronte di forte investimento iniziale. Infatti uno dei motivi principali per il quale abbiamo optato per il cloud computing è il grande costo, nonché investimento iniziale, nell'avere in casa tutta la tecnologia hardware necessaria al mantenimento live di CoolTraveling. Oltre ai costi non bisogna sottovalutare le skill necessarie alla gestione di apparecchiature hardware di questa tipologia per le quali ci saremmo sicuramente dovuti appoggiare a consulenti esterni.
Nella prima fase, ovvero la migrazione da Seeweb ad Amazon, abbiamo utilizzato un'unica istanza EC2 di tipologia micro. In questo modo siamo riusciti a verificare le performance del cloud di Amazon e migrare il servizio. Successivamente con la crescita di CoolTraveling abbiamo deciso di progettare un'adeguata architettura che potesse scalare nel tempo. Ogni scelta intrapresa è stata principalmente dettata da necessità intrinseche del servizio calate nel migliore dei modi sui prodotti Amazon.
Suddivideremo, per semplicità nelle descrizione, l'architettura delineata.

1. Load Balancing: Amazon ELB

   Per poter gestire al meglio molte connessioni contemporaneamente abbiamo de-

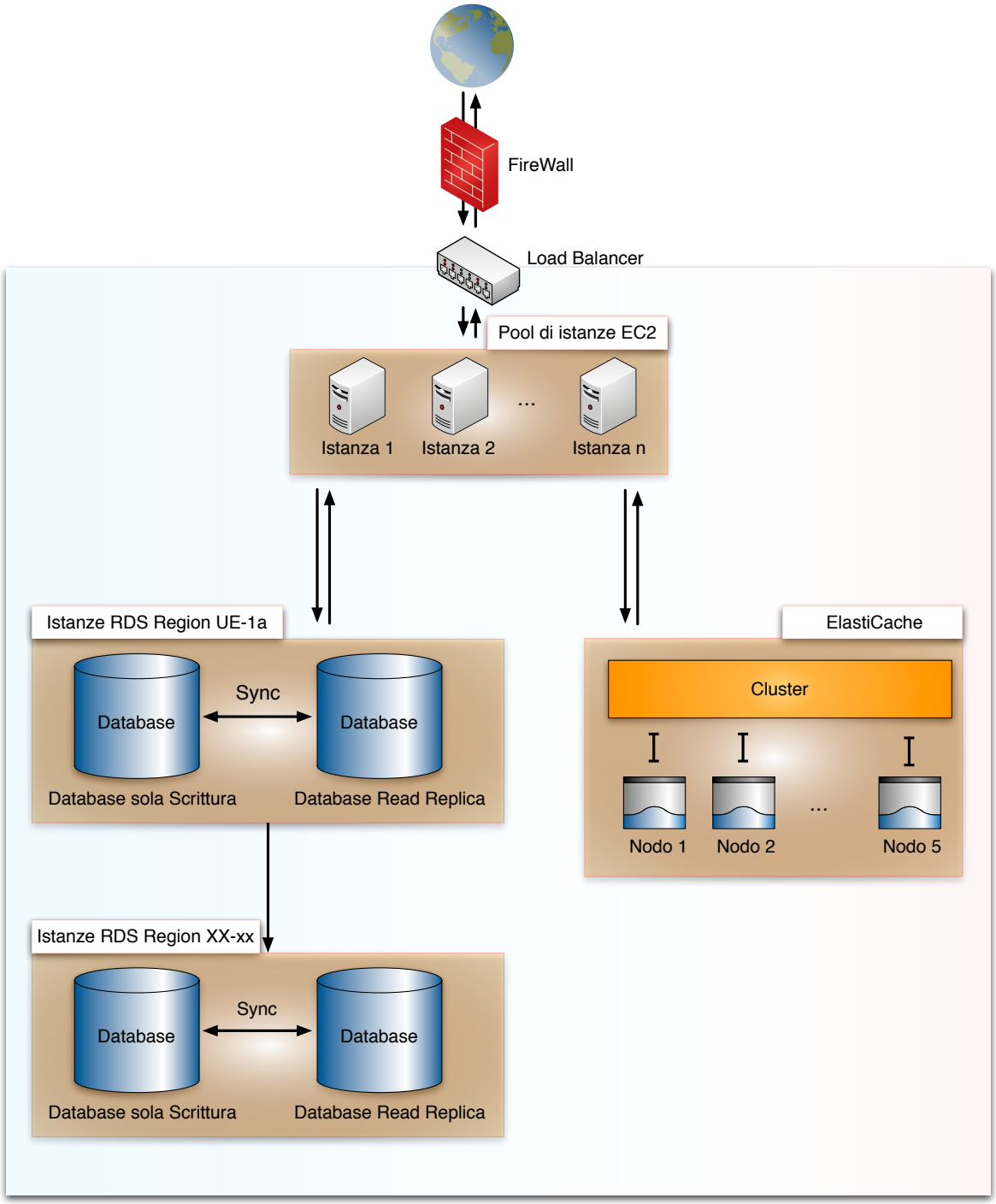


Figura 4.7: Architettura server esterna

ciso di introdurre un load balancer all'ingresso del servizio. In questo modo ogni connessione effettuata dagli utenti sarà diretta sul nodo del load balancer che si preoccuperà di instradare la connessione alle macchine attive e pronte a prendere in carico le richieste. Dopo aver configurato il load balancer è indispensabile associare le macchine alle quali potrà indirizzare le connessioni ricevute. Inizialmente su Seeweb non avevamo configurato alcun load balancer quindi stavamo costruendo un prodotto difficilmente scalabile in termine di elasticità e performance se non aumentando la potenza della macchina utilizzata. In questo modo invece è possibile agganciare al load balancer tante piccole istanze, con costi ridotti rispetto ad un unica macchina ad alte performance, lasciando al load balancer la scelta di scegliere quella più opportuna alla quale inviare le connessioni. A livello di load balancer è anche possibile effettuare un forward mapping delle porte a seconda delle necessità. Questo servizio è molto comodo quando sulle macchina operano più web application su porte diverse.
Amazon mette a disposizione nella configurazione del load balancer anche i parametri utilizzati dal sistema di check delle istanze attive associate. In questo modo è possibile configurare sulle proprie web app una url specifica che sarà chiamata ad intervalli regolari per verificare che il servizio su una specifica istanza sia up and running. Qualora ci dovessero essere delle latenze nella risposta l'health check del load balancer imposterà un feedback negativo sull'istanza specifica. Nel nostro caso abbiamo ritenuto opportuno configurare i seguenti parametri:

- Ping target: URL e porta utilizzata per il check
- Timeout: 5 secondi
- Interval: 30 secondi
- Unhealthy Threshold: 2, ovvero l'upper bound di check non andati a buon fine
- Healthy Threshold: 10, ovvero il numero di test positivi per il reintegro di un'istanza

2. Istanze di calcolo: Amazon EC2
   Il cuore dell'architettura, ovvero dove risiede tutta la potenza di calcolo è in questa sezione. Avendo a monte un load balancer abbiamo configurato due macchine gemelle in grado di smistare in modo del tutto indipendente tra loro le richieste

ricevute. In questo modo anche eventuali downtime di una delle due istanze non possono causare down del servizio ma solo dei rallentamenti. Tutte queste logiche sono gestite a livello di load balancer. All'interno delle singole istanze invece risiedono le web app che permettono a CoolTraveling di funzionare.
Come descritto in precedenza per poter utilizzare un'istanza Amazon è necessario scegliere un'immagine di una distribuzione di un sistema operativo che si ritiene idonea al proprio business. Nel nostro caso la scelta è ricaduta su distribuzioni Linux. Lavorando con Amazon EC2 abbiamo trovato molto utile la possibilità di configurare delle distribuzioni ad hoc per i propri scopi a partire da immagini vanilla rilasciate dai principali vendor in collaborazione con Amazon e quindi ottimizzate per il cloud computing. Una volta configurata la propria istanza è possibile congelarla creandone un'immagine. Questa immagine potrà poi essere usata in futuro per dar vita a nuove istanze che saranno del tutto identiche a quella appena configurata. In questo modo è possibile avere l'autoscaling del servizio anche tramite API invocate direttamente da servizi web.
Un'altra funzionalità che abbiamo trovato molto utile nel corso dei mesi di lavoro è stato il pannello di monitoring delle istanze direttamente presente dal pannello web fornito da Amazon. In questo modo è sempre possibile tenere sotto controllo l'andamento delle proprie macchine non dovendo necessariamente tener aperte per monitoring molteplici shell. Molto spesso infatti ci si trova a lavorare con più shell aperte per singola macchina sia per eseguire istruzioni che per monitorarne le performance. In questo modo Amazon ha dotato i propri clienti di uno strumento in grado di far visualizzare le performance in realtime direttamente da un'interfaccia web.
Abbiamo configurato inizialmente due micro instance, descritte precedentemente, successivamente abbiamo effettuato una migrazione e siamo passati a due small instance.
Dettagli Small Instance:

- RAM: 1.7GB
- CPU: 1 EC2 Compute Unit
- Storage: 160GB
- Architettura: 32bit

Al fine di garantire un alto livello di affidabilità, in caso di picchi del servizio, abbiamo configurato un'istanza aggiuntiva di tipologia large. Avendo l'istanza un costo molto elevato abbiamo ritenuto opportuno configurarla e tenerla pronta per l'uso nel momento del bisogno. A seguire una breve descrizione delle caratteristiche dell'istanza: Dettagli Large Instance:

- RAM: 7.5GB
- CPU: 4 EC2 Compute Unit
- Storage: 850GB
- Architettura: 64bit

Dai test effettuati è risultato esser molto difficile mettere sotto sforzo la large instance, soprattutto se messa in parallelo con le altre due small instance.

3. Database: Amazon RDS
   Come descritto precedentemente Amazon mette a disposizione un prodotto completo per la gestione del database al di fuori di un'istanza EC2. Questa soluzione ha molteplici vantaggi e nel nostro caso specifico abbiamo deciso di adottarla per i seguenti motivi:
   - Facilità di configurazione dell'istanza: operation system independent
   - Facilità di configurazione di MySql: i parametri di default sono già ottimizzati per il cloud
   - Scalabilità immediata e automatizzabile grazie all'autoscaling
   - Ridondanza dei dati grazie al Multi-AZ Deployment
   - Gestione automatizzata e configurabile di upgrade, backup

   Nello specifico l'istanza che abbiamo scelto e configurato presenta le seguenti caratteristiche:
   - Tipologia RDS instance: small

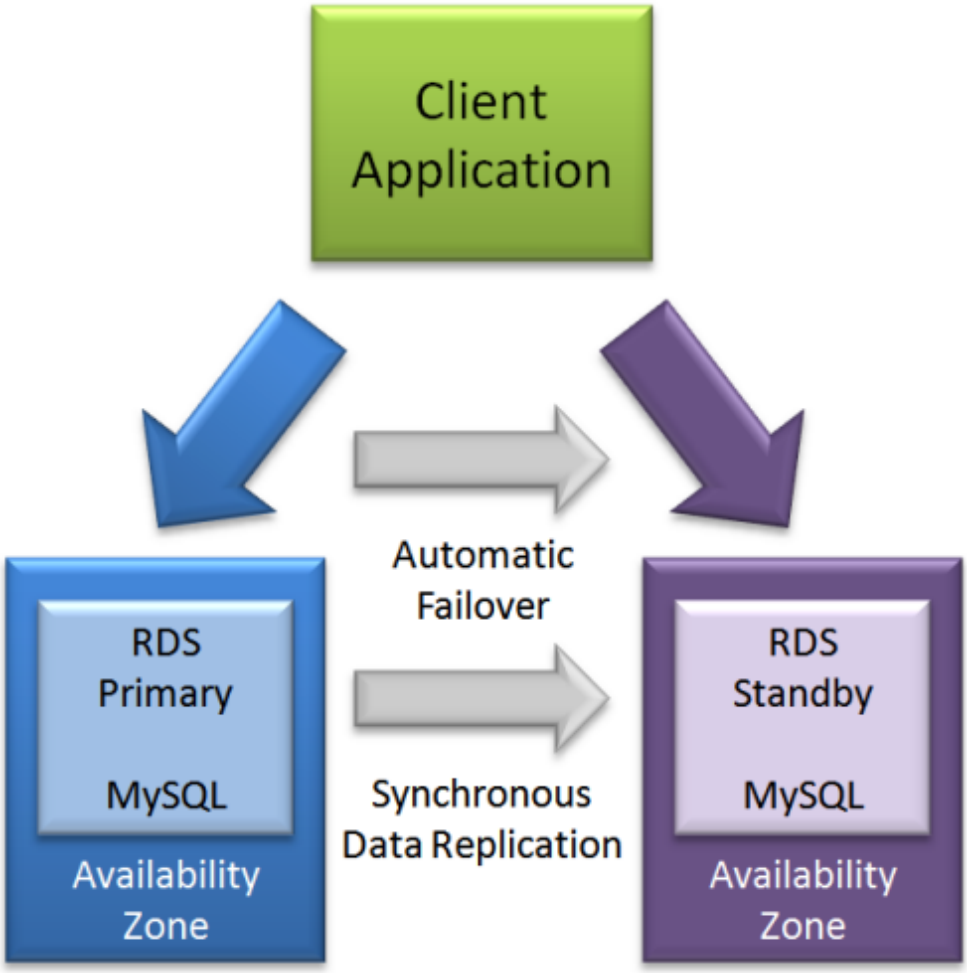


Figura 4.8: Amazon RDS - Multi AZ Deployment

- My SQL Version: 5.1.57
- Dimensione: 20Gb
- Multi Az Deployment: attivo
- Read Replica: attivo
- Configurazioni speciali: trigger attivi

Come è possibile intuire dalla configurazione soprastante il prodotto è molto standard e l'intervento umano al fine di configurarlo è stato davvero minimale. In pochi semplici click è possibile ottenere un'istanza RDS funzionante con elevate performance. Così come per l'EC2 anche RDS ha un pannello di monitoring molto interessante e customizzabile che permette agli utenti di tener sempre sotto controllo il proprio database. Qualora le configurazione disponibili da pannello web non siano sufficienti è sempre possibile fare ricorso alle API messe a disposizione da Amazon e configurare a mano tutti i dettagli del caso. In CoolTraveling siamo stati costretti ad adottare questa procedura solo per piccoli accorgimenti, uno su tutti l'attivazione dei trigger altrimenti disabilitati di default.

4. Cache: Amazon Elastic Cache
   Vista la natura di CoolTraveling abbiamo sin da subito ritenuto opportuno trovare un valido meccanismo di caching in grado di essere application indipendent. Questo requisito è fondamentale in quanto essendo l'architettura altamente scalabile ogni singolo componente può essere duplicato o triplicato in modo del tutto automatico. Avere quindi una cache legata alle singole istanze sarebbe stato un grosso vincolo. Fortunatamente Amazon ha inserito un prodotto molto interessante per sopperire a questo bisogno: Elastic Cache. Elastic Cache è completamente compatibile con il protocollo Memcached che già utilizzavamo in locale sulla singola macchina quando il nostro server era dislocato nella nostra mansarda. Grazie a questa compatibilità introdurre il layer di cache distribuito è stato davvero facile ed immediato. Gli step necessari sono stati sostanzialmente tre:

   (a) Creazione del Cluster di cache su Amazon

   (b) Sostituzione dell'endpoint della cache locale con l'endpoint di Elastic Cache

   (c) Calibrazione del servizio in funzione dei nodi di cache introdotti

Inizialmente abbiamo introdotto un solo cache Cluster con due nodi da 1.3 Gb ciascuno. Abbiamo condotto alcuni test alla ricerca del numero ideale di nodi e dimensione di quest'ultimi al fine di massimizzare il livello del servizio concludendo che da due a cinque nodi è il range ideale per il nostro tipo di servizio.

5. Sicurezza: Amazon Security Group
   La sicurezza all'interno di Amazon è gestita mediante dei gruppi che possono essere configurati al fine di permettere o negare degli accessi. Questa politica è condivisa nella maggior parte dei prodotti Amazon, in questo modo è possibile condividere le configurazioni realizzate minimizzando il tempo investito in questa attività. Un gruppo di sicurezza è identificato da un nome, un id e dalle sue configurazioni. Tutto ciò permette di abilitare o inibire particolari accessi ai prodotti Amazon. Nel nostro caso abbiamo configurato un gruppo di sicurezza per permettere l'accesso alle istanze Amazon EC2 utilizzare per far girare le Web Application. Senza questa configurazione sarebbe infatti impossibile anche solamente accedere via shell alle singole macchine. Come anticipato precedentemente i gruppi di sicurezza possono essere condivisi tra più prodotti Amazon. In alcuni casi questa condivisione è addirittura indispensabile per permettere il dialogo tra due tecnologie. Il caso di cui stiamo parlando è quello di Elastic Cache: per permettere ai moduli di Elastic Cache di comunicare con le istanze Amazon EC2 è indispensabile che queste appartengano allo stesso gruppo di sicurezza e che questo gruppo di sicurezza sia autorizzato per poter essere usato in questo preciso scopo, ovvero la cache. Nel caso del database (Amazon RDS) invece è possibile utilizzare gruppi di sicurezza diversi. Nei gruppi di sicurezza è possibile permettere o negare l'accesso a prodotti Amazon. Tipicamente una delle configurazioni standard per adempiere a questo scopo è la gestione delle interfacce e relative porte di accesso. Filtrando semplicemente indirizzo ip e porta, ad essi associati, i gruppi di sicurezza di Amazon sono in grado di avvicinarsi al ruolo di un firewall nella sua accezione più commerciale del termine, ovvero inibizione degli accessi non graditi.

6. Auto Scaling L'auto scaling è la possibilità di scalare la propria architettura sul cloud in modo automatico. Abbiamo identificato due diverse tipologie di auto scaling:

   (a) Black Box: in funzione di parametri sistemistici

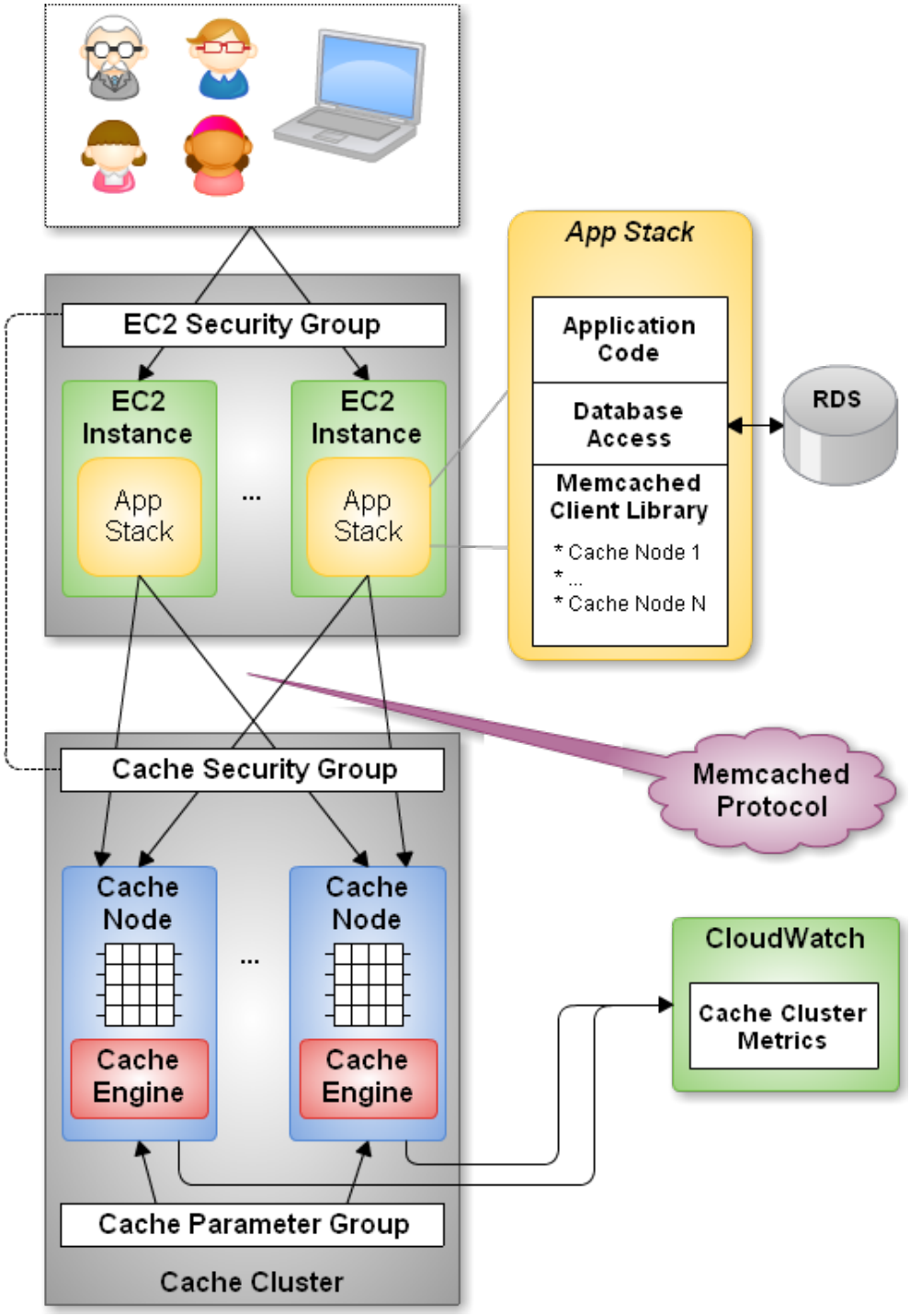


Figura 4.9: Amazon Elastic Cache

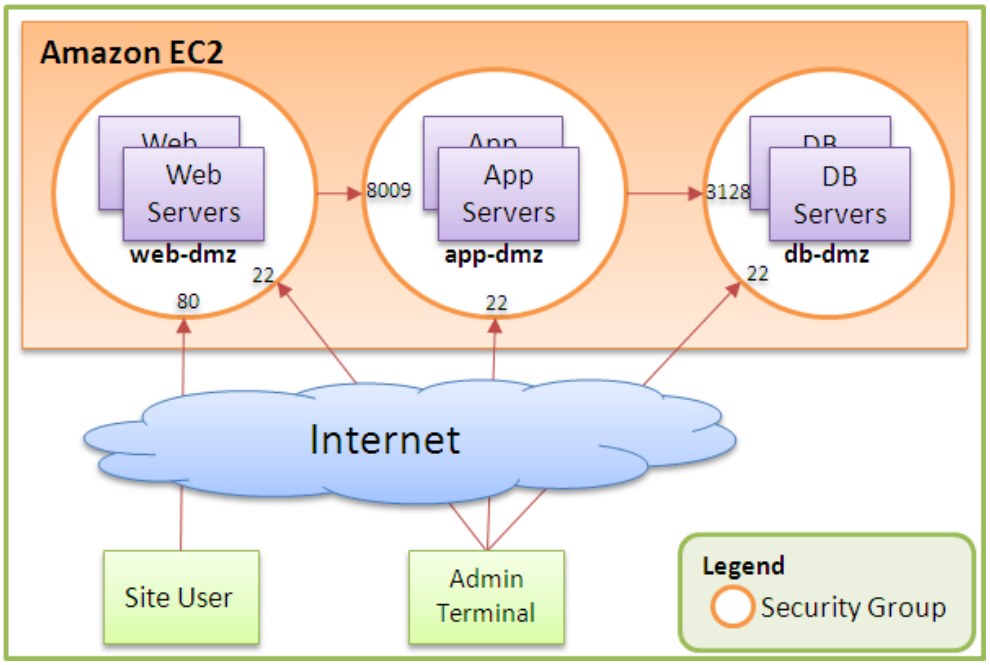


Figura 4.10: Amazon Security Group

(b) Web App driver: in funzione di KPI misurati internamente

Nel primo caso l'autoscaling è regolamentato dai parametri sistemistici delle macchine amazon su cui girano le web app. Un esempio può essere l'eccessivo consumo di CPU piuttosto che l'out of memory della RAM.
Affinché questo strumento possa funzionare correttamente è necessario configurare il Cloud Watch, un prodotto Amazon che consente di settare allarmi su specifiche risorse delle macchine. Attraverso il cloud watch è possibile impostare un livello di soglia superato il quale verrà attivato il segnale di alert. Successivamente, a discrezione del sistemista, sarà possibile gestire questo alert in dieresi modi, come ad esempio l'auto scaling. Nel secondo caso invece è grazie a KPI misurati internamente alla web application, quindi prettamente software/prestazionali, che è possibile attivare l'auto scaling sopra citato. Dalla nostra esperienza sul campo possiamo dire che nonostante le grandi potenzialità dell'autoscaling in funzione di KPI interni è molto difficile creare una metrica adeguata a tutte le casistiche che il mondo reale è in grado di concretizzare su un prodotto così altamente dipendente da servizi esterni. Abbiamo quindi ritenuto essere un miglior benchmark per l'autoscaling la sola analisi dei parametri sistemistici.
Nella nostra architettura viene automaticamente avviata una nuova istanza se si

verificano le seguenti condizioni:

- CPU: superamento soglia 80% per più di 10 minuti
- RAM: 20% di memoria rimasta libera per più di 5 minuti
- Connessioni: oltre 5000 connessioni al secondo (upper bound cautelativo, dato che durante i mesi di attività CoolTraveling non ha mai dovuto sopportare carichi di questo tipo
- HealthHost: Deve esserci sempre almeno un'istanza attiva

Ovviamente quando i parametri rientrano nei livelli di norma l'architettura è in grado di effettuare il downscaling in autonomia evitando così di gravare eccessivamente sui costi.

4.4.3 Region

Un'altro concetto molto importante all'interno della piattaforma di Amazon è il concetto di *region*. Tutta l'architettura amazon AWS è dislocata in diverse web farm sparse ormai per tutto il mondo. Attualmente le region attive sono le seguenti:

- US EAST (Northern Virginia)
- US WEST (Oregon)
- US WEST (Northern California)
- EU (Ireland)
- ASIA PACIFIC (Singapore)
- ASIA PACIFIC (Tokyo)
- SOUTH AMERICA (Sao Paulo)
- US GovCloud

Vista la dimensione di ciascuna farm la struttura interna è stata suddivisa in ulteriori parti. All'interno di ogni region sono presenti delle zone. Nel nostro abbiamo scelto come region quella europea basata a Dublino in Irlanda. All'interno di questa region sono identificate tre zone:

- ue-west-1a
- ue-west-1b
- ue-west-1c

Anche se formalmente la farm europea è unica, quando si configura un prodotto amazon vi è la possibilità di selezionare anche la specifica zona della region all'interno della quale si vuole configurare il proprio servizio. E' consigliabile configurare tutti i propri servizi all'interno di un'unica region in modo che non ci sia una fatturazione di traffico extra dovuto allo scambio di dati tra zone differenti. Avere la possibilità di configurare prodotti su zone differenti all'interno della stessa region è però un enorme vantaggio dal punto di vista della ridondanza. Eventuali guasti infatti spesso si concentrano su una zona specifica della region interessata. Il 7 Agosto 2011 la farm irlandese di Amazon a causa di una tempesta di fulmini ha subito dei gravi danni e un'intera zona della region è stata compromessa. La sfortuna ha voluto che parte della nostra architettura fosse interessata dal guasto. Nel caso specifico tutti i pannelli web di Amazon hanno subito un forte rallentamento, probabilmente dovuto al numero di tentativi di accesso da parte di tutte le persone interessate dal guasto. Oltre al pannello web fortemente rallentato alcune istanze, tra le quali anche la nostra, sono prima state riavviate e poi sospese. Amazon ha rilasciato continui update periodici avvisando tempestivamente tutti i possessori di istanze interessate dal guasto sullo stadio dei lavori per un eventuale recupero dati. Al termine del periodo di crisi solo il 5% delle istanze interessate dal guasto è stato dichiarato totalmente irrecuperabile, in questo 5% era ovviamente presente anche la nostra istanza.
La notte stessa del guasto abbiamo tempestivamente provveduto a risolvere la situazione senza attendere che Amazon mettesse una pezza al problema, in quanto era evidente che una calamità naturale di questa entità avrebbe potuto mettere in ginocchio qualunque colosso della tecnologia. La soluzione che abbiamo identificato è stata l'istanziare una nuova macchina del tutto identica a quella inaccessibile facendola nascere in una zona non interessata dal guasto. Fortunatamente, utilizzando il servizio di autoscaling, avevamo configurato un'istanza del tutto identica a quella precedentemente attiva. Siamo quindi riusciti in poco tempo a rimappare il servizio su una zona differente della region minimizzando il downtime.

Come è possibile vedere nell'articolo di TechCrunch, assieme alla nostra Web App anche qualcuno di più famoso ha avuto seri problemi di stabilità. Questo espediente ci

Figura 4.11: TechCrunch: 8 Agosto, down Web Farm UE Ireland Amazon

ha fatto capire l'importanza di avere prodotti replicati su più region e allo stesso tempo istanze pronte a partire su zone diverse della stessa region. Fortunatamente l'istanza RDS contenente il database non è stata interessata da alcun guasto importante e siamo riusciti a recuperare i dati senza alcun problema.

4.4.4 I costi

Considerando il costo orario dei seguenti prodotti Amazon:

- Load Balancer
- EC2 Micro Instance
- RDS small instance
- Elastic Cache small

la nostra architettura ha avuto un costo medio mensile di circa 200$. E' importante specificare il costo medio in quanto utilizzando i sistemi di autoscaling spesso vengono attivate più istanze contemporaneamente.

Capitolo 5

Servizi: Servizi: Viaggiatreno, Meteo, Accommodation

5.1 Estrazione ed integrazione di siti web

Recentemente internet ha visto nascere un gran numero di siti web e motori di ricerca che offrono i servizi più disparati, dalla ricerca indirizzata a domini specifici, quali ad esempio voli, hotel, concerti, ristoranti, ad una più generale, fornita dai più comuni motori di ricerca (Google, Yahoo!). A partire da questa "visione" risulta comodo distinguere tra servizi esatti, le cui risposte possono essere caratterizzate da un unico risultato (come ad esempio i servizi di geolocalizzazione) non vincolato da nessuna classificazione e servizi di ricerca la cui peculiarità è fornire un elenco di risposte in un determinato ranking.

Dopo aver classificato i diversi servizi presenti in rete, è necessario selezionare quali di questi effettivamente possano essere interessanti dal punto vista dei contenuti informativi offerti e dalla possibilità di essere interrogati ed estratti.

Per quanto riguarda i contenuti informativi bisogna porre l'attenzione su ciò che il sito effettivamente mette a disposizione, sia per quanto riguarda la precisione delle risposte riferite ad un determinato dominio, sia per la qualità e la quantità di queste ultime; è abbastanza inutile avere un sito che viene aggiornato raramente e che, in un range di tempo molto ampio, fornisca poche risposte; infine è indispensabile che il sito sia sempre reperibile e abbia dei tempi di risposta relativamente accettabili.

Non meno importante è la questione relativa alle modalità di interrogazione ed estrazione. Dal nostro punto di vista il primo ostacolo da affrontare per il wrapping

di un sito è sicuramente la gestione del link: per gestione si intende la possibilità di creare un URL valido che ci permetta anche di avere una certa quantità di procedure invocabili, in modo da rispettare nella maniera più precisa possibile la richiesta in input ed evitare anche di avere come risposta dei dati di cui non si ha bisogno.
In questo contesto altamente eterogeneo, la nostra attenzione si è concentrata su servizi atti a soddisfare l'obiettivo di CoolTraveling e allo stesso tempo che soddisfino requisiti che rendano il processo di wrapping il più lineare e semplice possibile. A tale scopo, sono stati identificati i seguenti servizi:

- **ViaggiaTreno.it** per la ricerca dello stato e della puntualità dei treni
- **AccuWeather.com** per recuperare informazioni relative al meteo
- **Bed-And-Breakfast.it** per la ricerca di eventuali B&B, Hotel e Accomodation presenti nella città di destinazione

Nei successivi paragrafi verranno analizzati due dei quattro servizi elencati (il procedimento di parsing, al netto del codice, è standard e vale per tutti i siti oggetto di wrapping), in modo da capire in maniera approfondita come sono stati realizzati i vari wrapper e le difficoltà che potrebbero presentarsi.

5.1.1 ViaggiaTreno.it: analisi ed estrazione

Viaggiatreno.it è il servizio di trenitalia che permette di conoscere in tempo reale lo stato di tutti i treni che stanno circolando sulla rete gestita da RFI.

Abbiamo deciso di interrogare ed estrarre le informazioni dall'interfaccia mobile e non da quella web per una serie di motivi:

- l'interfaccia mobile ha dei tempi di risposta molto più brevi dato che la sua struttura è molto semplice e non contiene pubblicità o riferimenti a siti esterni
- la struttura della pagina HTML è molto regolare e questo ha facilitato la progettazione della grammatica ANTLR
- l'interfaccia web è basata in Adobe Flash e quindi la sua interrogazione ed estrazione sarebbe stata molto più complicata e non ci avrebbe fornito alcun dato in più rispetto a quella mobile

Il servizio può essere interrogato attraverso il *numero del treno*, il *nome della stazione* oppure conoscendo *partenza e destinazione*. Noi abbiamo deciso di utilizzare solo il primo parametro dato che siamo sempre a conoscenza del numero del treno.

Di seguito l'elenco dei parametri che abbiamo estratto dalle pagine web:

- Stazione di partenza
- Stazione di destinazione
- Ora partenza programmata
- Ora partenza reale
- Binario di partenza previsto
- Binario di partenza reale
- Arrivo programmato
- Arrivo previsto
- Binario di arrivo previsto
- Binario di arrivo reale
- Ora arrivo prevista
- Ora arrivo reale
- Ritardo

Se il parametro *ritardo* assume un valore positivo significa che il treno è in ritardo, se è negativo allora è in anticipo, se è uguale a zero è in perfetto orario.

Durante la progettazione del layer di Cache (che verrà approfondito nella sezione 5.2 a pagina 96) abbiamo deciso di non memorizzare le risposte di questo servizio dato che l'aggiornamento dello stesso avviene in real time e si rischierebbe di dare all'utente un'informazione "vecchia".

5.1.2 AccuWeather.com: analisi ed estrazione

AccuWeather[12] è una società americana che offre servizi di previsioni meteorologiche a livello mondiale basati su informazioni provenienti da numerose fonti, comprese le osservazioni meteorologiche e la raccolta di dati del National Weather Service (NWS), da organizzazioni al di fuori degli Stati Uniti e anche da società non meteorologiche come l'Environmental Protection Agency e dalle forze armate.

AccuWeather ha introdotto diverse innovazioni nel campo della meteorologia: ha ridefinito i valori degli indici di vento freddo e caldo con un unico valore noto come "AccuWeather Exclusive RealFeel Temperature". La formula per calcolare questo valore include gli effetti della temperatura, umidità, vento, l'intensità della luce del sole, nuvolosità e le precipitazioni.
Accuweather, a differenze degli altri servizi analizzati, da come output un file xml facilmente analizzabili in Ruby tramite semplici parser standard offerti dalla sua libreria.

Analisi dettagliata: costruzione dell'URL e regolarità del codice

È abbastanza semplice costruire una URL di interrogazione per AccuWeather.com in quanto gli unici parametri di cui necessita sono il continente, lo stato e il nome della città (inseriti come valori per location). Esempio potrebbe essere il seguente:
`http://forecastfox.accuweather.com/adcbin/forecastfox/weather_data.asp?location=EUR;IT;-;MILANO&metric=1&partner=forecastfox` .

Il compito risulta semplificato qualora volessimo ricercare le previsioni meteorologiche di una città Statunitense, in quanto l'unico valore da fornire alla variabile "location" è lo zip code della città stessa. Da questo link si ricava il file XML[10] dal quale verranno estratte alcune informazioni significative: la massima e la minima temperatura giornaliera e notturna per un range di date a partire dal giorno corrente fino ai successivi 9 giorni. Purtroppo AccuWeather non consente di effettuare una ricerca stagionale del meteo al di fuori degli Stati Uniti. In figura 5.1 è mostrato un frammento di codice XML delle previsioni del primo giorno. Naturalmente la regolarità del codice è scontata dato che si tratta di un file XML e la sua gestione avviene semplicemente scandendo l'albero associato.

```
- <day number="1">
  + <url></url>
    <obsdate>2/19/2008</obsdate>
    <daycode>Tuesday</daycode>
    <sunrise>7:01</sunrise>
    <sunset>17:47</sunset>
  - <daytime>
      <txtshort>Mostly sunny</txtshort>
      <txtlong>Mostly sunny</txtlong>
      <weathericon>02</weathericon>
      <hightemperature>13</hightemperature>
      <lowtemperature>2</lowtemperature>
      <realfeelhigh>13</realfeelhigh>
      <realfeellow>6</realfeellow>
      <windspeed>1</windspeed>
      <winddirection>SSW</winddirection>
      <windgust>5</windgust>
      <maxuv>3</maxuv>
      <rainamount>0.00</rainamount>
      <snowamount>0.00</snowamount>
      <precipamount>0.00</precipamount>
      <tstormprob>0</tstormprob>
    </daytime>
  - <nighttime>
      <txtshort>Increasing cloudiness</txtshort>
      <txtlong>Increasing cloudiness</txtlong>
      <weathericon>38</weathericon>
      <hightemperature>13</hightemperature>
      <lowtemperature>2</lowtemperature>
      <realfeelhigh>13</realfeelhigh>
      <realfeellow>6</realfeellow>
      <windspeed>0</windspeed>
      <winddirection>S</winddirection>
      <windgust>1</windgust>
      <maxuv/>
      <rainamount>0.00</rainamount>
      <snowamount>0.00</snowamount>
      <precipamount>0.00</precipamount>
      <tstormprob>0</tstormprob>
    </nighttime>
  </day>
```

Figura 5.1: Frammento di codice XML del servizio AccuWeather.com

5.1.3 Bed-And-Breakfast.it: analisi ed estrazione

Bed-And-Breakfast.it è il punto di riferimento più completo per la ricerca di un bed and breakfast (ma non solo) in tutta Italia. Attraverso questo portale è possibile avere informazioni esaustive riguardo la struttura dove si vorrà soggiornare: ci sono molte foto dettagliate, tutti i vari contatti (telefono, mail, indirizzo...), la descrizione della zona e la relativa mappa. Bed-and-breakfast.it permette inoltre di effettuare la prenotazione direttamente dal sito e, per quanto riguarda i proprietari, pubblicizzare le strutture in maniera semplice e veloce.

Analisi dettagliata: costruzione dell'URL e regolarità del codice

La gestione dell'URL è molto semplice:
`http://www.bed-and-breakfast.it/cerca.cfm?q=san+marco+in+lamis`
Infatti, come mostrato nel link, basta fornire come valore per la variabile "q" il nome della città. In figura 5.2 è mostrato un esempio di pagina di ricerca per la città di Tropea. Il codice HTML è molto regolare e ci permette di estrarre tutte le informazioni necessarie (figura 5.3). Per effettuare una ricerca più precisa abbiamo ritenuto necessario estrarre anche la via di ogni struttura, ciò però comporta, come per l'estrazione del prezzo in TicketOne.it, la necessità di far riferimento ad un'altra pagina web e quindi stabilire una nuova connessione. Questo naturalmente non gioca a nostro favore perchè aumenta i tempi di processamento (ad esempio, in una pagina contenente 25 risultati, bisogna effettuare altre 25 connessioni stateless).

Interrogazione di Bed-And-Breakfast.it

Dopo un'attenta analisi dell'URL e del codice HTML iniziamo ad esporre le varie procedure attraverso le quali i metodi da noi implementati estrarranno e metteranno a disposizione i dati in maniera del tutto autonoma: partendo da un Item in input contenente le chiavi indispensabili per la ricerca, si passa alla vera e propria interrogazione e conseguente wrapping della pagina.

Di seguito sono illustrati i passi necessari per la procedura:

1. Estrazione dei parametri identificati univocamente da chiavi specifiche secondo i vincoli stabiliti dall'interfaccia standard.
   Al fine di costruire un link valido, occorre ricevere in input la seguente chiave:

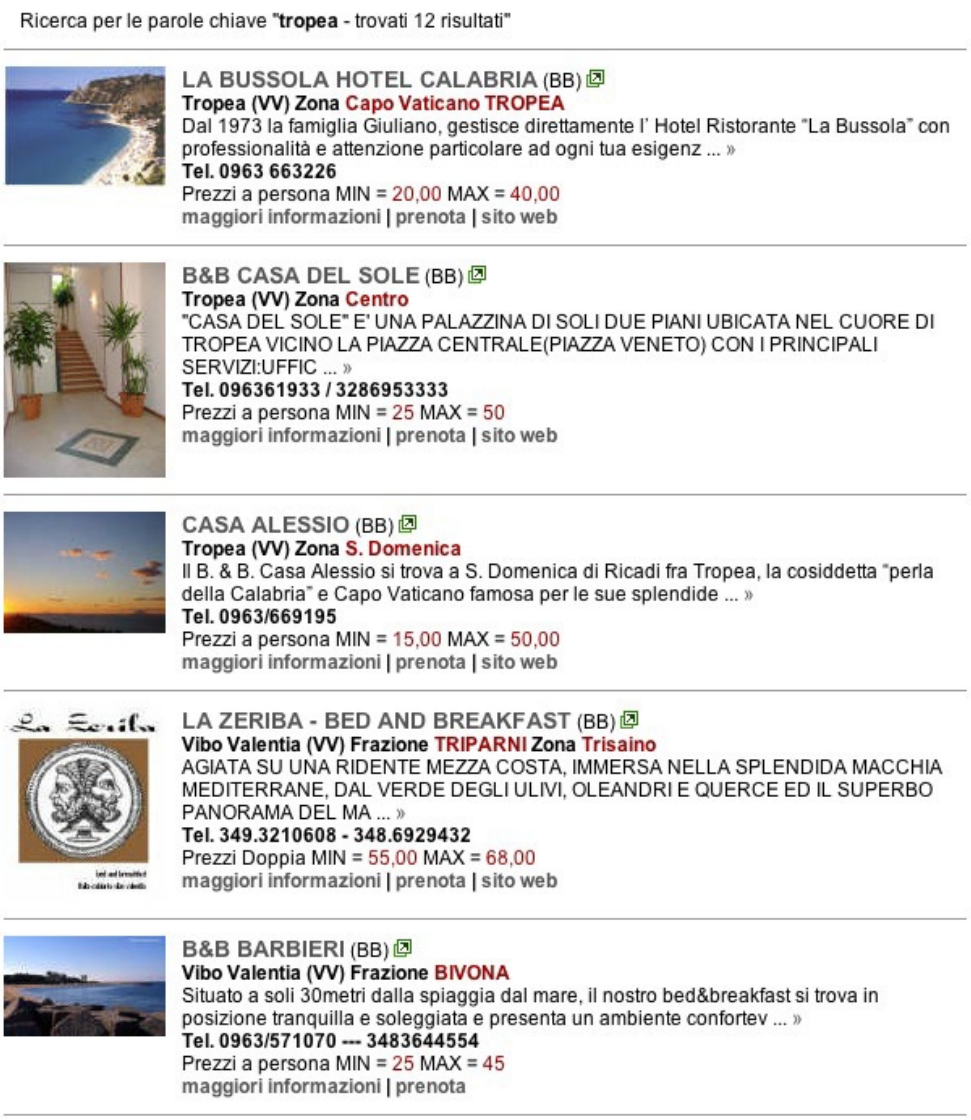


Figura 5.2: Home page del sito www.bed-and-breakfast.it

```
<span class="titolo-bb"><a href="pagina.cfm?ID=6140&IDregione=3">
LA BUSSOLA HOTEL CALABRIA</a></span>
<div class=\'testo\'>Bed and Breakfast</span>
<br><span class="info-bb">Tropea (VV)
Zona <span class="evidenziato">Capo Vaticano TROPEA</span></span>
<a href="pagina.cfm?ID=6140&IDregione=3">
<span class="info-bb">Tel. 0963 663226</span>
<!-- inizio PREZZI --->
<br>Prezzi a persona MIN = <span class="evidenziato">20,00</span>
MAX = <span class="evidenziato">40,00</span>
<!-- fine PREZZI --->
```

Figura 5.3: Frammento di codice HTML del sito www.bed-and-breakfast.it

- *KEY_CITY*, corrispondente alla città nella quale si vuole cercare la struttura ricettiva

2. Dopo aver estratto correttamente le chiavi dall'Item si procede alla creazione del link.

3. A questo punto è possibile stabilire la connessione internet per catturare lo stream HTML.

4. L'HTML così immagazzinato viene dato "in pasto" allo *StreamReader* di ANTLR. In seguito lo stream viene passato come parametro di ingresso del lexer: in questo momento avviene il vero e proprio wrapping della pagina web, *BBLexer lexer=new BBLexer(input);* .

5. Dopo aver wrappato la pagina, verrà prelevato il contenuto informativo utilizzando il comando **lexer.getArrayList()** che da in output un ArrayList di oggetti di tipo *BedAndBreakfast*. L'ArrayList viene scandito in modo da riempire un secondo ArrayList (questa volta di Item) associando ad ogni chiave *KEY_BB* l'oggetto *BedAndBreakfast*.

5.2 Cache - Ottimizzazioni

Considerando che lo smartphone ormai è divenuto parte integrante della nostra giornata e che gli utenti sono soliti utilizzarlo per riempire i momenti "morti", è necessario porre attenzione alla velocità di risposta attesa da parte dell'utente, ovvero quel limite di tollerabilità a cui quest'ultimo è abituato per non abbandonare l'applicazione che sta utilizzando, sia essa relativa a news o ricerca di voli.
A partire da una ricerca condotta sul tema[16], si è scoperto che dopo 3 secondi di attesa, il tasso di abbandono per i device mobile sono molto simili a quelli calcolati su desktop (vedi Figura 5.4). Sulla base di tali tesi, valutando in maniera ingegneristica il problema, ci siamo resi conto che anche in questo caso era possibile apportare miglioramenti nei tempi di risposta attesi, ponendo attenzione al trade-off tra il tempo di risposta stesso e l'attendibilità dei risultati.
Infatti, dalle prime analisi effettuate durante l'utilizzo dell'app da parte di un campione di utenti, si è riscontrata la necessità di gestire in maniera autonoma richieste molto simili in modo da garantire un tempo di risposta immediato, senza effettuare nuove

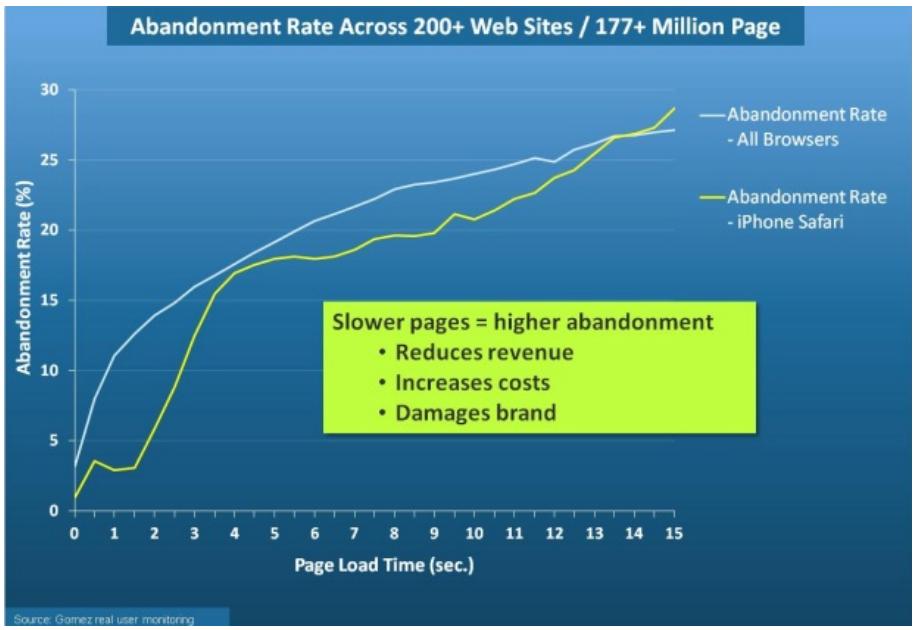


Figura 5.4: Confronto tasso di abbandono delle pagine tra desktop e mobile

richieste a servizi esterni. Questo approccio ci consente anche di gestire eventuali fornitori futuri che offrono un servizio a pagamento basato sul numero di query effettuate. Il sistema di caching in Rails ci permette di soddisfare la nostra esigenza in maniera semplice attraverso l'utilizzo della libreria **Memcached**.
Tutttavia prima di ogni implementazione, è indispensabile affrontare alcune tematiche rilevanti per permettere di valutare al meglio il trade-off sopracitato:

- **Clustering dei servizi di trasporto**: le prenotazioni effettuate per treni ed aerei presentano delle logiche differenti su alcuni fattori chiave come ad esempio i costi di una tratta. Si riscontra ovviamente che il costo di un biglietto aereo ha una frequenza di aggiornamento nettamente superiore a quella che è possibile osservare per quanto riguarda i treni (molto statica e non cambia durante la giornata). Di conseguenza, per la natura dei servizi che sono stati coinvolti, è necessario gestire in maniera autonoma la cache degli stessi, realizzandola quindi a livello di singolo servizio.

- **Cache Aerei**: i valori relativi alle ricerche effettuate dagli utenti, per le considerazioni definite nel punto precedente, vengono mantenute in cache secondo il sistema definito nei prossimi paragrafi.

- **Cache Treni**: i valori relativi alle ricerche effettuate dagli utenti verranno mantenuti in cache per un periodo di tempo pari a 24 ore.

5.3 Memcached per Ruby on Rails

Memcached è un sistema di caching basato su chiave-valore, per la memorizzazione di piccole porzioni di dati arbitrari (Stringa, Oggetto generico) derivanti dai risultati di chiamate su database, api esterne o rendering delle pagine. Sviluppato originariamente da Danga Interactive per LiveJournal (una community virtuale), attualmente è utilizzato da numerosi siti quali YouTube, Facebook e Twitter. Anche Amazon WebService utilizza memcached attraverso la sua soluzione ElastiCache che è stata trattata nel Paragrafo 4.3.

5.3.1 Cache Aerei: La nostra soluzione

Come accennato precedentemente il sistema di caching che gestisce i voli deve essere altamente dinamico in quanto i costi dei voli cambiano in maniera repentina durante il giorno. Quindi vi è la necessità di un algoritmo che permetta di gestire i tempi di validità dei valori presenti in cache sulla base di logiche che permettano di evitare "inutili" chiamate ai servizi ridefinendo il *tempo di expiring*.

Vediamo insieme come la cache abbia fortemente contribuito a rendere più efficiente CoolTraveling. La cache in questo servizio viene utilizzata per salvare le soluzioni appena richieste dagli utenti. Così facendo le richieste simili provenienti da diversi utenti saranno riconosciute dal sistema il quale le restituirà direttamente agli utenti riducendo drasticamente i tempi di attesa. Diversamente dai treni, i dati degli aerei, hanno delle informazioni che variano più velocemente (prezzo, posti disponibili, ecc ...). E' stato quindi necessario introdurre un expiration time della cache molto piccolo. L'expiration time è il periodo di tempo in cui riteniamo valide le informazioni all'interno della cache, dopo di che se nuovamente necessarie bisognerà richiederle nuovamente al servizio esterno per ricevere le informazioni eventualmente aggiornate. Per non determinare un expiration time fisso a priori abbiamo strutturato il sistema in modo che sia possibile variare l'expiration time in real time, tenendo in considerazione il tempo effettivo di validità dei dati stessi.

Come si può vedere dallo pseudo-codice al momento del caricamento delle informazioni nella cache viene associato un expiration time dopo la quale il sistema di cache eliminerà l'informazione.

function RICERCA_SOLUZIONI_AEREI(*partenza*, *arrivo*, *datetime*)
    resp = get_soluzioni_aeri_from_cache(*partenza*, *arrivo*, *datetime*)
    if *resp* != *NULL* **then**
        return *resp*
    end if
    resp = request_aeri(*partenza*, *arrivo*, *datetime*)
    add_aerei_in_cache(*partenza*, *arrivo*, *datetime*, *experation_time*)
    New.Thread {verifica_vecchie_soluzioni(*partenza*, *arrivo*, *datetime*, *resp*) }
    return *resp*
end function

function VERIFICA_VECCHIE_SOLUZIONI(*partenza*, *arrivo*, *datetime*, *new_data*)
    old_data = ricerca_db_vecchi_dati(*partenza*, *arrivo*, *datetime*)
    if *old_data* == *NULL* **then**
        return
    end if
    if *is_equal*(*new_data*, *old_data*) **then**
        experation_time = *experation_time* * 1,1
        if *experation_time* > *MAX_EXPIRATION_TIME* **then**
            experation_time = ÊMAX_EXPIRATION_TIME
        end if
    else
        experation_time = *experation_time* * 0,9
        if *experation_time* < *MIN_EXPIRATION_TIME* **then**
            experation_time = ÊMIN_EXPIRATION_TIME
        end if
    end if
end function

Grazie a questo sistema CoolTraveling si è quindi dotato di un sistema di caching

intelligente e in grado di adattarsi nel tempo in funzione delle richieste utente. A seguire vedremo la logica su cui tutto si basa.

Quando in cache non ci sono le informazioni richieste dall'utente dovremo necessariamente interrogare i servizi esterni, prima di restituire le informazioni all'utente. Quando invece in cache ci sono le informazioni richieste dall'utente ma sono scadute, ovvero ritenute troppo vecchie è comunque necessario interrogare i servizi esterni. In questo caso specifico però, successivamente all'invio dei dati all'utente verrà avviato un nuovo processo in grado di rendere la cache autonoma nella gestione del tempo di validità. Le casistiche sono sostanzialmente due:

- Dati scaricati coincidenti con i dati in cache e scaduti
- Dati scaricati più aggiornati dei dati in cache scaduti.

Nel primo caso l'algoritmo incrementerà il tempo di validità della cache visto che i nuovi dati scaricati sono identici a quelli ritenuti scaduti. Mentre nel secondo caso verrà ridotto il tempo di validità. Per evitare attese da parte dell'utente tutti questi controlli sono eseguiti in parallelo, alla risposta che viene a lui fornita, sfruttando la funzione Thread.new(...). Quando la verifica sarà completata i tempi di validità delle cache saranno aggiornati e disponibili per future consultazioni.

Per garantire solidità all'algoritmo abbiamo introdotto un valore minimo e massimo per il tempo di validità della cache. Dalle analisi sui dati è stato osservato come nel weekend ci sia una maggiore dinamicità sul servizio che porta ad una naturale diminuzione del tempo di validità della cache in quanto i servizi interrogati offrono di volta in volta informazioni aggiornate a fronte delle numerose prenotazioni ricevute. Durante la settimana si vede invece un graduale aumento del tempo di validità della cache.

5.4 Diagramma delle classi

Di seguito sono rappresentati i *diagrammi delle classi*[1] dei processi utilizzati per l'acquisizione delle informazioni dai diversi servizi web. In tali diagrammi vengono

[1] Un *diagramma delle classi* è un'evoluzione del diagramma entità-relazione (ER). In un diagramma delle classi si rappresentano le entità significative del sistema e le relazioni che esistono fra le entità. È possibile rappresentare diversi tipi di classi (astratte, parametriche, ...) e diversi tipi di relazioni (ereditarietà, associazioni, composizioni, ...).

rappresentate le principali entità che compongono i diversi wrapper e l'insieme delle relazioni che le legano.

5.4.1 Wrapper di viaggiatreno

Vedi Figura 5.5;

5.4.2 Wrapper del weather

Vedi Figura 5.6;

5.4.3 Wrapper dei bed and breakfast

Vedi Figura 5.7;

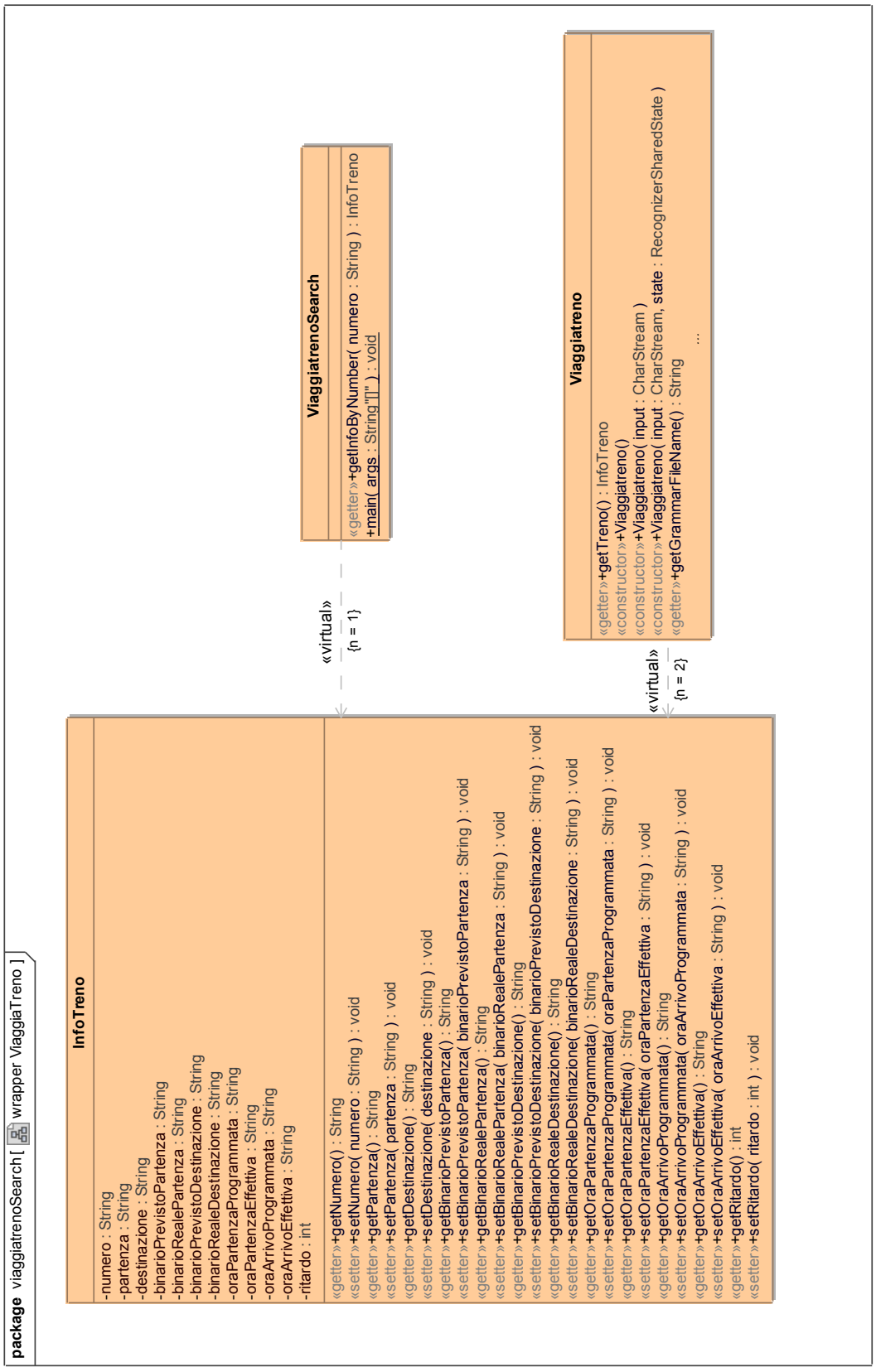


Figura 5.5: Diagramma delle Classi - Wrapper di Viaggiatreno

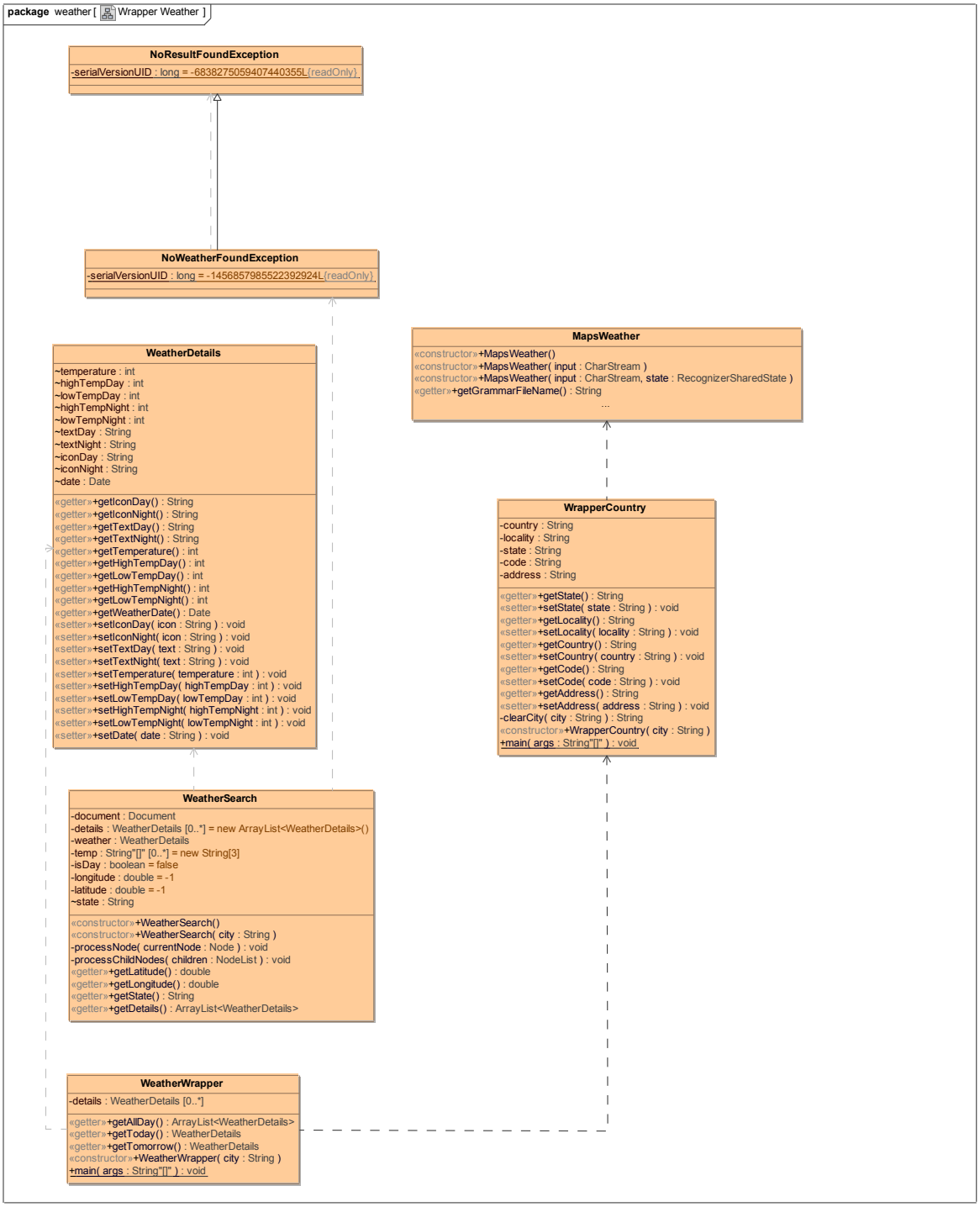


Figura 5.6: Diagramma delle Classi - Wrapper del Weather

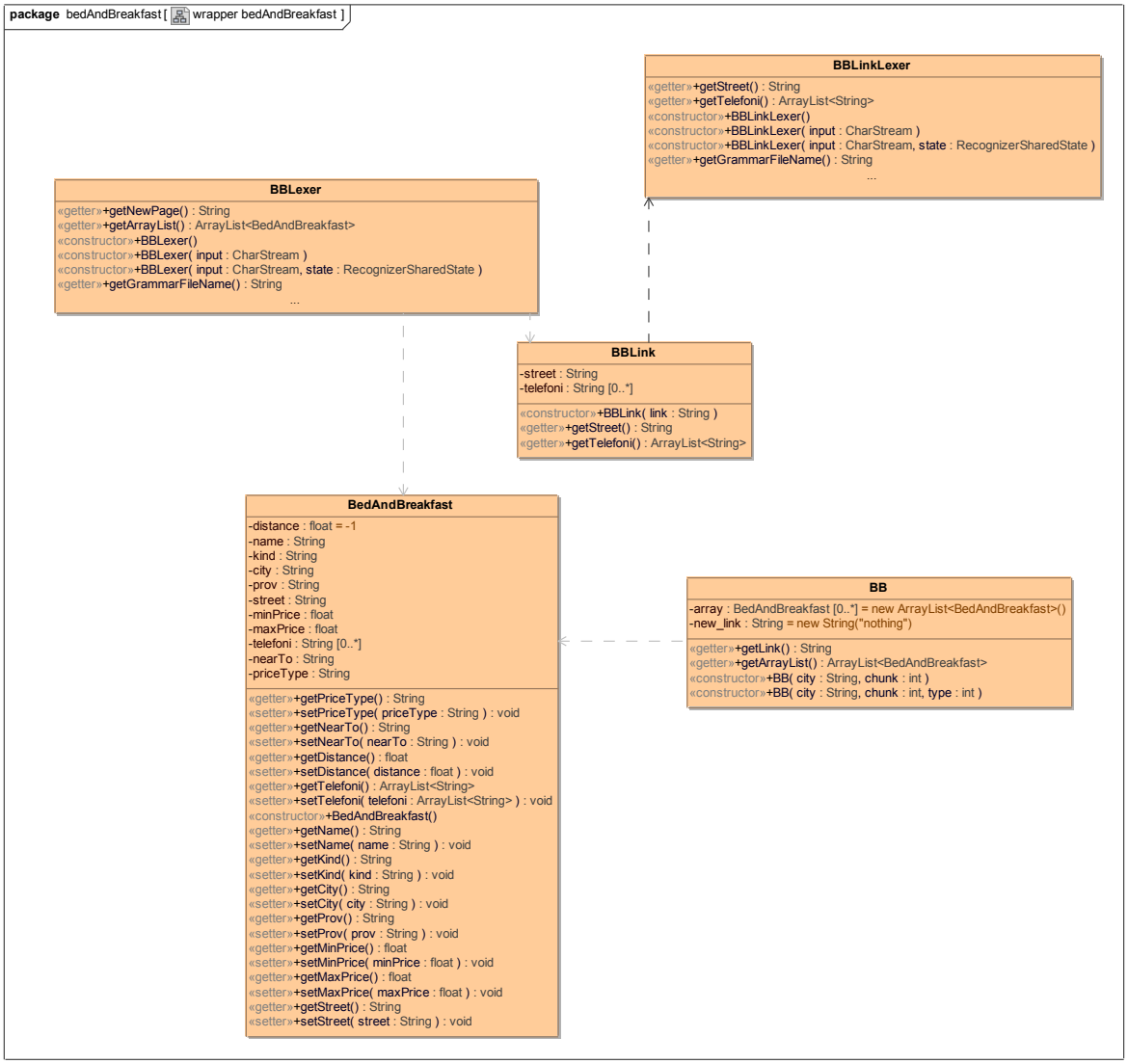


Figura 5.7: Diagramma delle Classi - Wrapper dei Bed And Breakfast

Capitolo 6

L'applicazione mobile

La sezione relativa all'applicazione mobile mira ad illustrare tutte le features realizzate per fornire un client per device mobile al fine di usufruire dei servizi realizzati. Vedremo quindi tutte le scelte intraprese dal punto di vista tecnologico, tenendo presente che l'investimento maggiore dal punto di vista del client tendeva all'ottimizzazione della parte software con l'hardware(device fisico) imposto dalle regole del concorso descritto nei precedenti capitoli. Successivamente, verranno descritte le innovazioni introdotte dopo il concorso, volte a rendere anche device con piattaforma iOs e Android compliant con il client realizzato. Infine, sono stati delineati gli studi di usabilità che hanno portato alla produzione dei mockup utili a porre delle base per lo sviluppo nativo di una applicazione iOs.

6.1 Vodafone 360

Vodafone 360 è una piattaforma per smartphones e PC progettata per sostituire il portale vodafone live, che integra al suo interno social network, come facebook e twitter, servizi di IM come windows live, servizi di gestione contatti come google contacts e un proprio store di applicazioni.

E' stata lanciata il 24 settembre 2009 da Vodafone in partnership con Samsung[1]. Samsung assieme a Vodafone si è quindi presentata sul mercato con due smartphones, cavalli di battaglia della suddetta piattaforma: Samsung M1, Samsung H1.
Di seguito le caratteristiche principali del modello Samsung Vodafone 360 H1:

[1]Partner unico ed unico fornitore dell'hardware su cui è installato il sistema operativo Vodafone 360

Figura 6.1: Piattaforma Vodafone 360

- Display AMOLED 3.5" touch screen
- GPS (AGPS)
- OS Linux Limo R2
- Processore ARM Cortex A8 600 MHz
- Wifi/Fotocamera
- Memoria interna microsd da 8 o 16 GB

Tali devices erano anche i primi ad implementare una logica di applicazioni Widget su Home Screen.

6.1.1 Piattaforma di sviluppo

La piattaforma Vodafone 360 è stata sviluppata per permettere la realizzazione di apps/widget web oriented compliant con le specifiche del W3C, permettendo l'utilizzo di tutti gli strumenti per lo sviluppo web come HTML, javascript(e librerie associate

Figura 6.2: Samsung H1

come jquery, motools, ecc) e CSS. I maggiori vincoli nello sviluppo di apps e widget sono da associare ai limiti intrinseci del browser presente nel telefono. Vodafone, in partnership con Samsung, decise di adottare come browser standard una versione customizzata del famoso browser per smarthpone Opera mini. Nonostante gli sforzi effettuati per customizzare il browser e dare un'ottima resa nella gestione dei widget, è necessario segnalare alcune lacune e limitazioni:

- Supporto limitato per gli eventi DOM
- Impossibilità nell'eseguire script in background
- Supporto AJAX molto limitato
- Nessun supporto ad animazioni e transizioni
- I widget non funzionavano se si era in modalità offline
- Forti limitazioni sugli usi classici di internet come ad esempio relative chat

La nostra esperienza, nello specifico, ci ha portato a superare dei limiti intrinseci della piattaforma delineati precedentemente: *Nessun supporto ad animazioni e transizioni*. Come si può notare dal prototipo realizzato, l'applicazione CoolTraveling presenta transizioni animate durante il passaggio tra le diverse view.

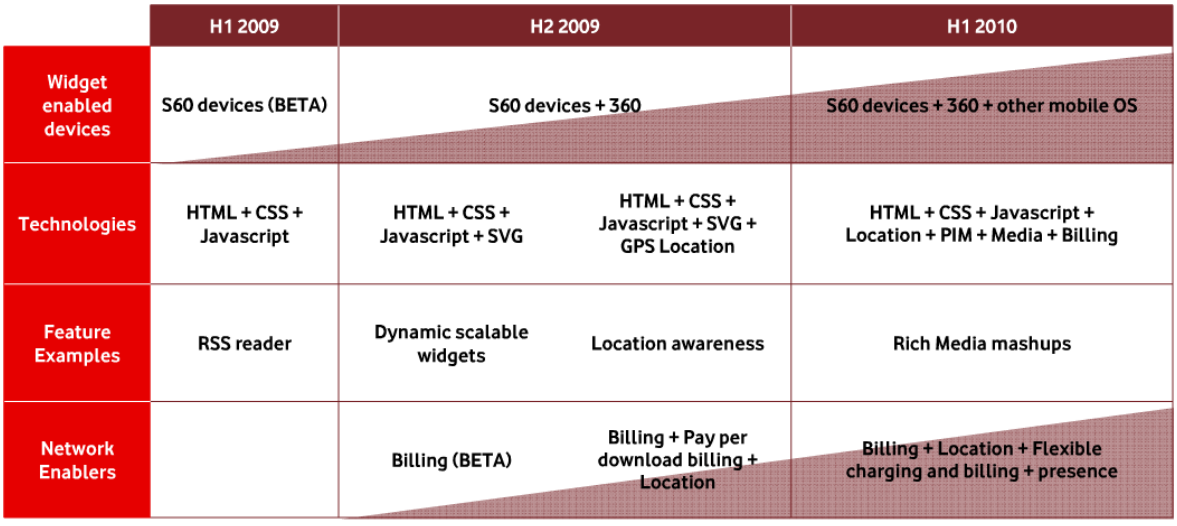


Figura 6.3: Roadmap di sviluppo su 360: Devices e Apis

6.2 Il Prototipo

Il prototipo di CoolTraveling è stato interamente realizzato secondo le guideline delineate nella sezione precedente diventando quindi un widget[2] a tutti gli effetti. Definiamo Widget, nello specifico web Widget, una piccola applicazione che può essere installata e eseguita all'interno di un ambiente web. Durante il concorso non abbiamo avuto modo di effettuare studi di usabilità e mockup grafici indispensabili per ottenere un prodotto di ottima qualità. Vedremo quindi a posteriori l'analisi effettuata per i due punti sottolineati. Abbiamo strutturato il client in 4 parti fondamentali:

- Accesso: login e registrazione
- Homepage: ricerca soluzioni
- Comparazione: confronto soluzioni di viaggio
- Analisi: approfondimento dettagliato di singole soluzioni di viaggio
  - Treno
  - Aereo
  - B&B
  - Meteo

Gli strumenti utilizzati per lo sviluppo del prototipo sono:

- HTML
- CSS
- Javascript: motools, jquery

6.2.1 Accesso

Quando l'utente clicca sul widget all'interno della piattaforma Vodafone 360 viene portato alla sezione di accesso. Al primo avvio verrà richiesto all'utente di scegliere la lingua predefinita per i futuri accessi: le lingue disponibili attualmente sono italiano e

[2]Il termine è stato applicato per la prima volta agli elementi dell'interfaccia utente durante il Project Athena negli anni '80, deriva dalla contrazione dei termini window e gadget.

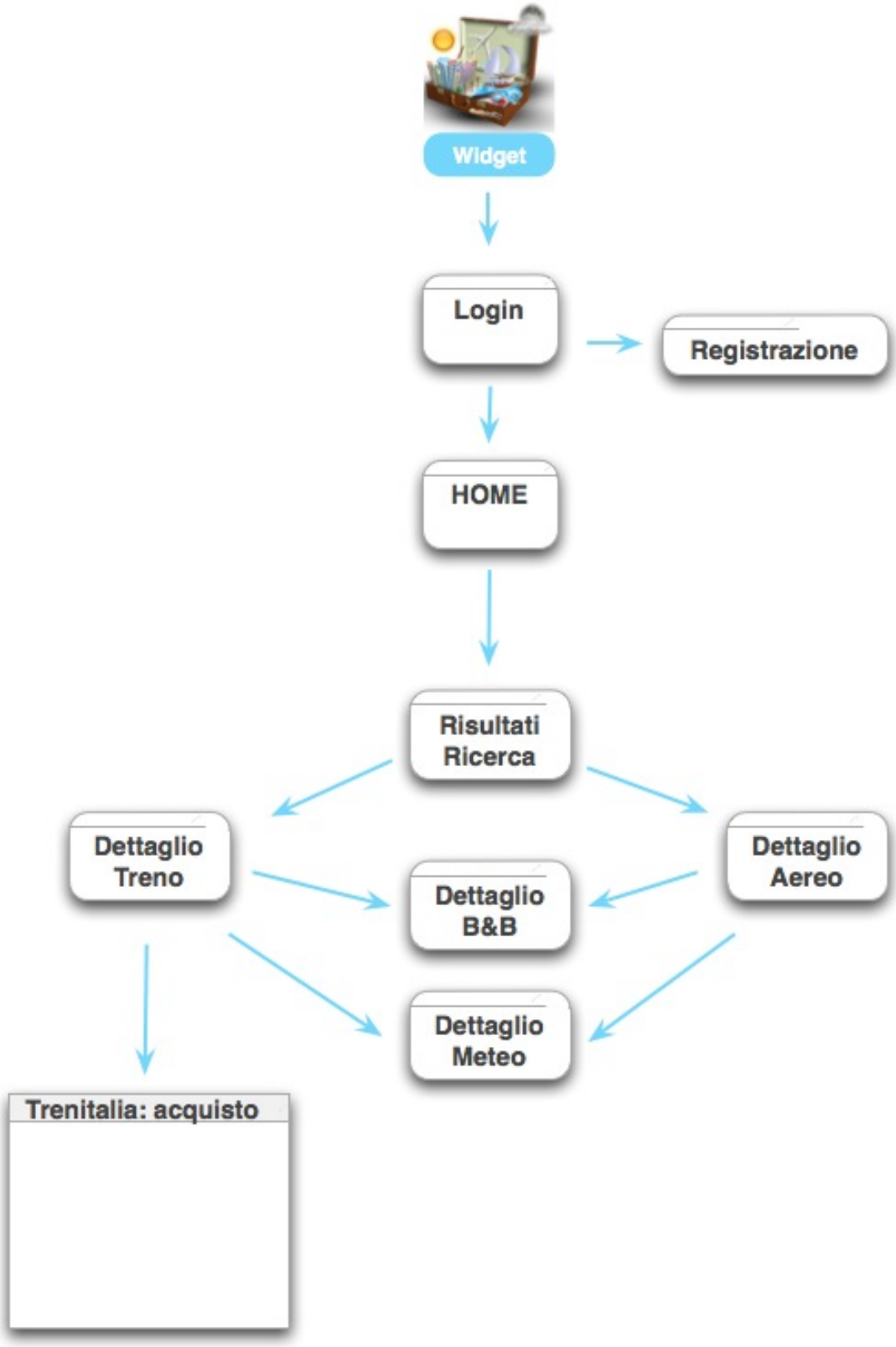


Figura 6.4: View Map CoolTraveling Widget

inglese. Successivamente l'utente visualizzerà una schermata di login, all'interno della quale, se già registrato, potrà procedere ed utilizzare l'app, altrimenti potrà dirigersi verso la schermata di registrazione. La fase di registrazione prevede l'inserimento di alcuni dati personali, utilizzati successivamente per scopi statistici e prevede il classico processo con double optin: l'utente riceverà una mail di conferma e il suo account verrà immediatamente attivato dopo aver verificato il proprio indirizzo mail.

6.2.2 Homepage

La home page permette di effettuare ricerche di treno e/o aerei. E' possibile inserire le seguenti informazioni:

- Città di partenza
- Città di destinazione
- Data/orario andata
- Data/orario ritorno (se selezionato)
- Opzioni: possibilità di includere/escludere i differenti mezzi di trasporto

E' opportuno sottolineare l'assistenza fornita all'utente durante la compilazione dei campi relativi alla città di partenza e di arrivo[3]. Quando l'utente inserisce la città di partenza o di arrivo, dopo aver completato l'inserimento, un sistema automatico di validazione verifica l'esistenza del comune inserito. Possono essere presenti differenti scenari:

- Se la città inserita non è presente nel database di CoolTraveling (in cui sono mappati tutti i comuni italiani), un'icona di warning notificherà l'utente dell'impossibilità da parte del sistema di riconoscere il dato inserito. Cliccando sull'icona di warning, verrà visualizzato un messaggio che spiega all'utente di verificare la città inserita in quanto non risulta essere italiana e che quindi la ricerca continuerà per i soli aerei (gli aeroporti non hanno la necessità di essere mappati dato che la richiesta viene direttamente inoltrata al servizio che interroga Expedia.it).
- Se la città inserita è presente, un'icona di conferma notificherà l'utente dell'avvenuto riconoscimento del dato inserito da parte del sistema.

[3]Feature realizzata per il solo territorio italiano

- Qualora la città inserita rappresenta per il sistema un caso di ambiguità (e.g., Porto), una combo box comparirà a supporto dell'utente, elencando le possibili soluzioni da selezionare. Riteniamo essere lo strumento di validazione di estrema importanza all'interno dell'applicazione sia per l'utente, che viene guidato nell'inserimento dei dati, sia dal punto di vista delle performances lato backend. Infatti, grazie al sistema di validazione server-side fatto in casa, i servizi esterni sono chiamati solamente quando si ha la certezza, o quasi, che i dati inseriti siano sintatticamente e semanticamente corretti.

6.2.3 Comparazione

Una volta effettuata correttamente la ricerca, verranno visualizzati i risultati trovati da CoolTraveling. Per ogni soluzione, sono disponibili le seguenti informazioni:

- Data e ora di partenza
- Data e ora di arrivo
- Città di partenza
- Città di arrivo
- Prezzo
- Durata
- Possibili cambi
- Logo

L'utente ha la possibilità di ordinare le soluzioni per:

- Partenza
- Prezzo
- Durata

Nel caso in cui l'utente effettui una ricerca A/R, gli verrà data la possibilità di poter visualizzare all'interno di una sola schermata, il dettaglio dei viaggi di andata e ritorno scelti.

6.2.4 Analisi

Nel processo di analisi viene descritta la sezione più prettamente legata al dettaglio delle soluzioni scelte. Al suo interno, l'utente può attingere ad un livello di informazioni aggiuntivo rispetto a quello finora visto. Nelle soluzioni inerenti agli aerei ad esempio, sono evidenziati:

- Nome della compagnia aerea
- Numero del volo
- Codice aeroporto

Per quanto riguarda i treni, invece:

- Numero del treno
- Eventuale ritardo
- Prezzo 1a e 2a classe
- link a pagina acquisto Trenitalia.com

Il punto di forza di questa sezione è non solo la possibilità di visualizzare con maggior dettaglio le soluzioni di viaggio, ma permette di scoprire le condizioni meteo e le accomodation della città di partenza e di arrivo. Quest'ultimi presentano sezioni di dettaglio specifiche del servizio selezionato. Accedendo alle informazioni del meteo sarà possibile visualizzare le previsioni meteorologiche per tutta la settimana e relative alla città scelta. Il livello di dettaglio delle condizioni meteorologiche è molto accurato e presenta due osservazioni, diurna e notturna, corredate da relative immagini, temperatura minima e massima. L'icona che permette di richiamare il dettaglio approfondito del meteo, raffigura sempre le condizioni presenti nel giorno in cui è stata effettuata la ricerca.

Per quanto riguarda le accomodation, qualora l'utente fosse interessato ad approfondire la ricerca, verrà portato verso una schermata che visualizzerà la lista delle accomodation presenti nella città selezionata. In tale pagina, potrà visualizzare per ogni soluzione, il nome dell'acomodation, la tipologia di accomodation (e.g., hotel, B&B, ecc) e il prezzo minimo per persona. Cliccando su una delle soluzioni proposte, l'utente potrà approfondire ulteriormente il dettaglio dell'accomodation. All'interno di questa sezione, vengono fornite le seguenti informazioni:

- Tipologia di accomodation
- Descrizione
- Indirizzo
- Sito Web
- Email
- Telefono
- Prezzo minimo
- Prezzo massimo
- Tipologia di prezzo (e.g., a persona, a famiglia, a settimana, ecc)

Tutti i contenuti messi a disposizione dell'utente possono essere fruiti direttamente via mobile. Ad esempio, se presente, è possibile visualizzare il sito web piuttosto che effettuare un'azione *click to call* e contattare direttamente la struttura alberghiera.

6.3 iOs & Android

Visto il successo di CoolTraveling durante il concorso, abbiamo cercato la strada più veloce per far approdare il nostro progetto sulle piattaforma leader sul mercato: iOs e Android. Dopo un primo studio di fattibilità sul porting del nostro progetto verso il nativo (Objective-C per iOs e Java per Android), abbiamo ritenuto essere eccessivamente dispendioso, in termini di tempo e effort, la sua realizzazione. Pertanto le nostre attenzione si sono prima di tutto concentrate sui nascenti ambienti crossplatform[4] ma poi hanno tentato di trovare una soluzione nei nuovi paradigmi web, come HTML5. Il problema che riscontriamo in quest'ultima soluzione sta nell'impossibilità di raggiungere in poco tempo un importante bacino di utenza dato dal presidio dei rispettivi store. La soluzione più immediata per il porting del progetto, senza introdurre complessità nello sviluppo, consistette nell'incapsulare all'interno di una web view i contenuti web della nostra application. In questo modo gli utenti avrebbero avuto la percezione di poter utilizzare un'applicazione sempre presente nella loro home screen.

[4]Gli ambienti cross-platform verranno trattati dettagliatamente nei capitoli successivi

Durante i successivi test, ci siamo resi conto che le performance e la qualità a cui le applicazioni sono giunte lato client ci ha spinti a pensare che l'implementazione nativa del progetto sia l'unica strada percorribile per il futuro. I cali di performance rispetto ad una applicazione nativa sono stati calcolati nell'ordine di circa 30/40% rispetto ad una normale visualizzazione nativa.

6.4 Mock-up applicazione

Di seguito le immagini del mock-up per iOs. (da immagine 6.5 a 6.14)

Figura 6.5: Mock-up: schermata di avvio

Figura 6.6: Mock-up: schermata di scelta viaggio

Figura 6.7: Mock-up: schermata con le città impostate

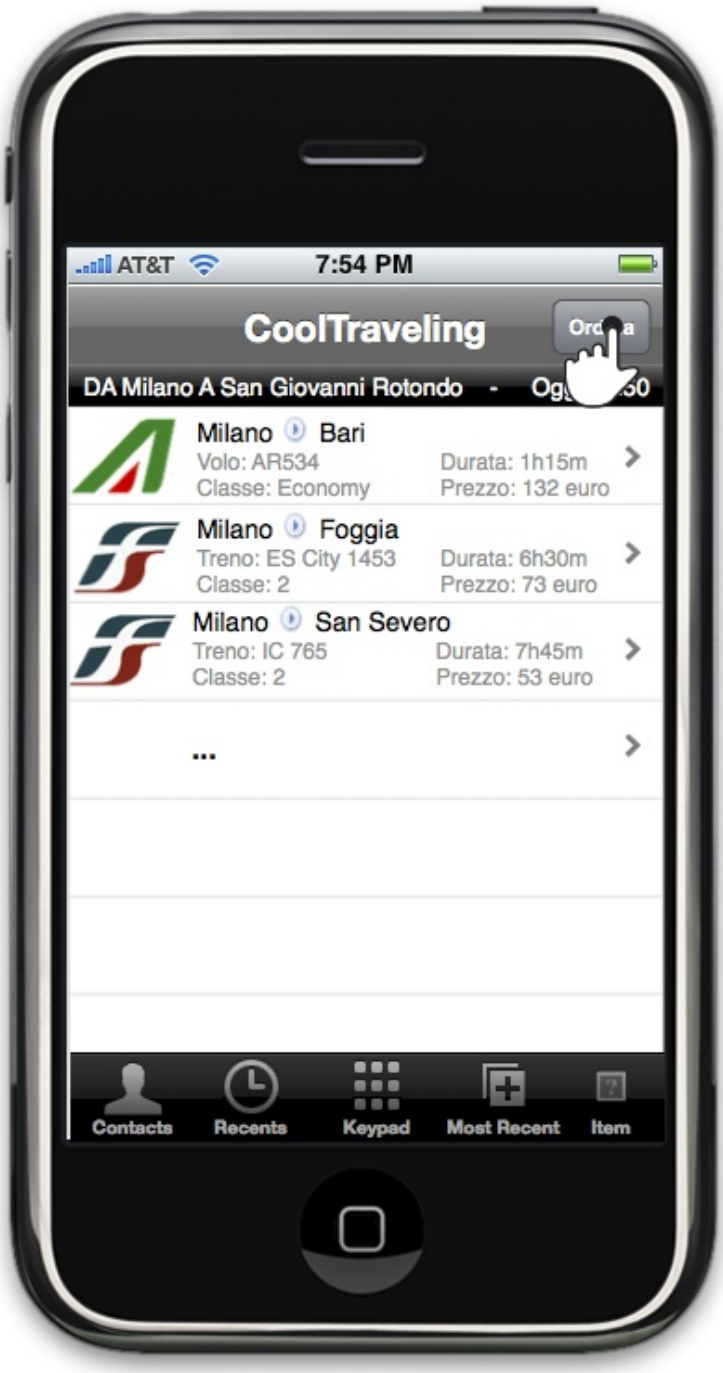


Figura 6.8: Mock-up: elenco soluzioni

Figura 6.9: Mock-up: ordinamento soluzioni

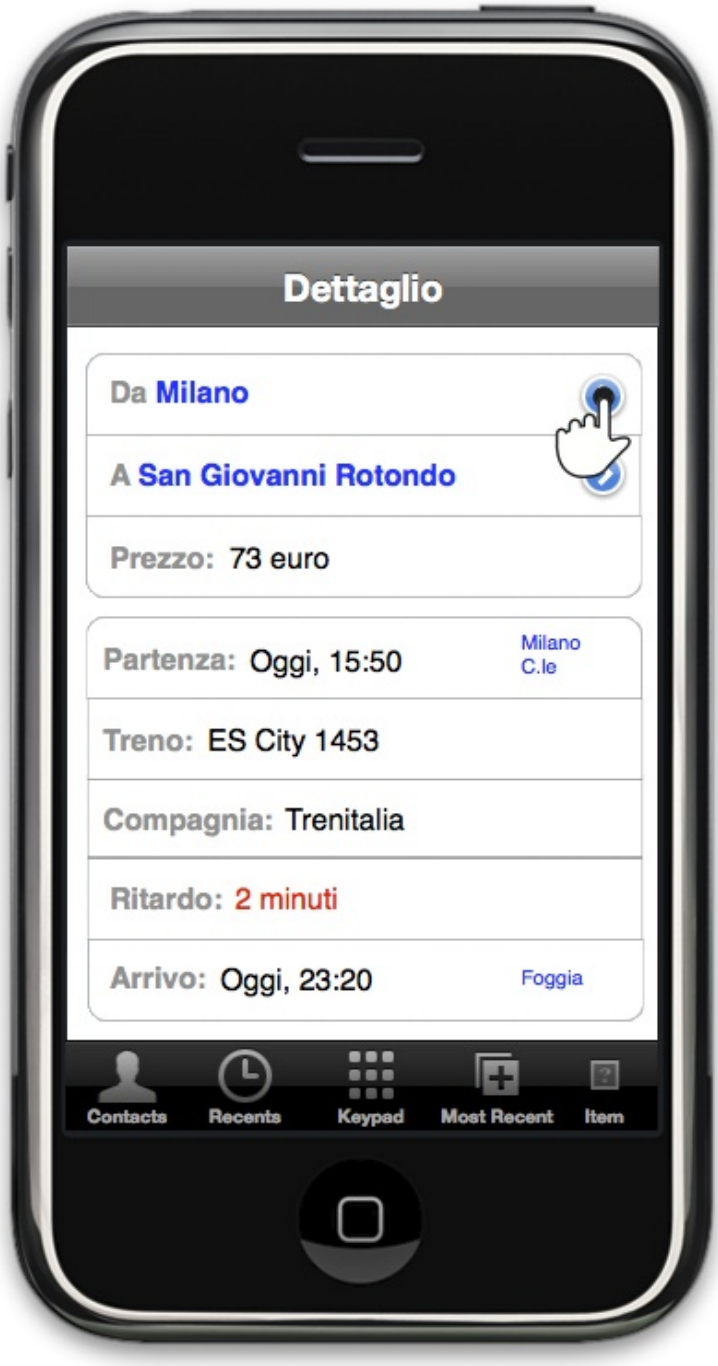


Figura 6.10: Mock-up: dettaglio soluzione

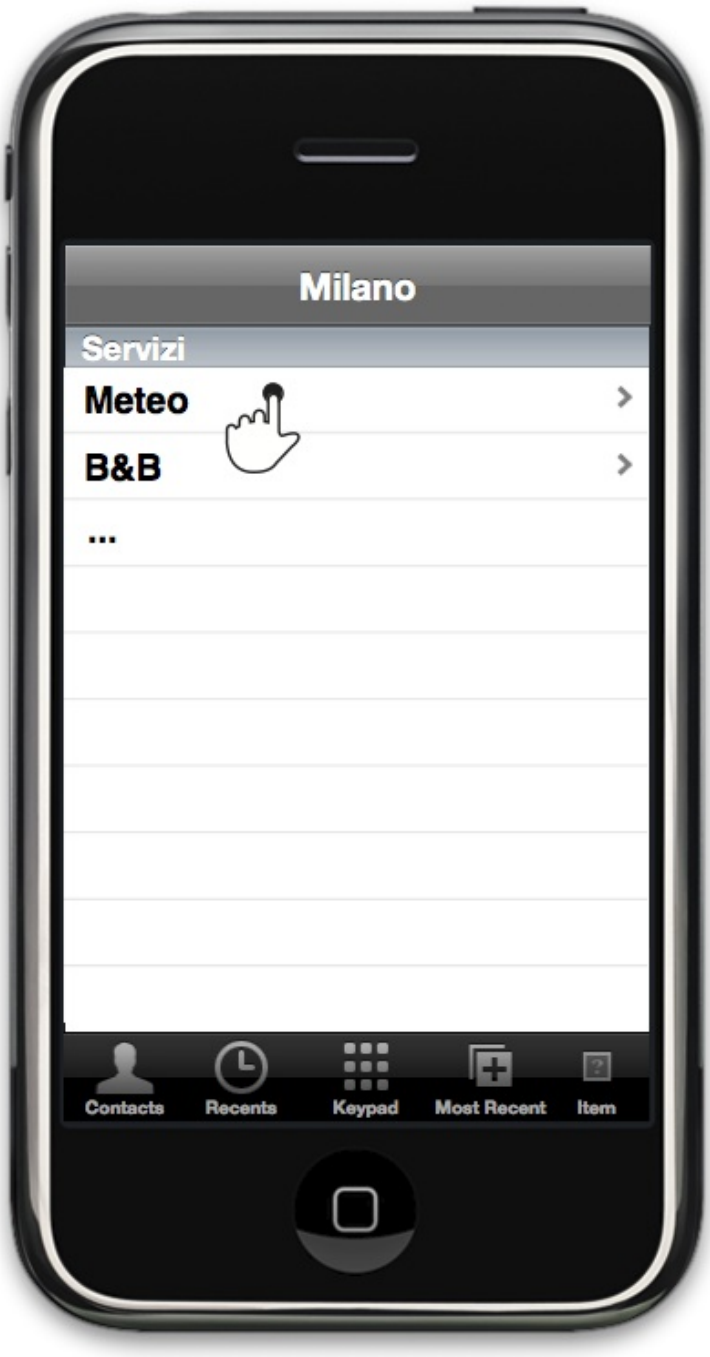


Figura 6.11: Mock-up: informazioni aggiuntive

Figura 6.12: Mock-up: meteo

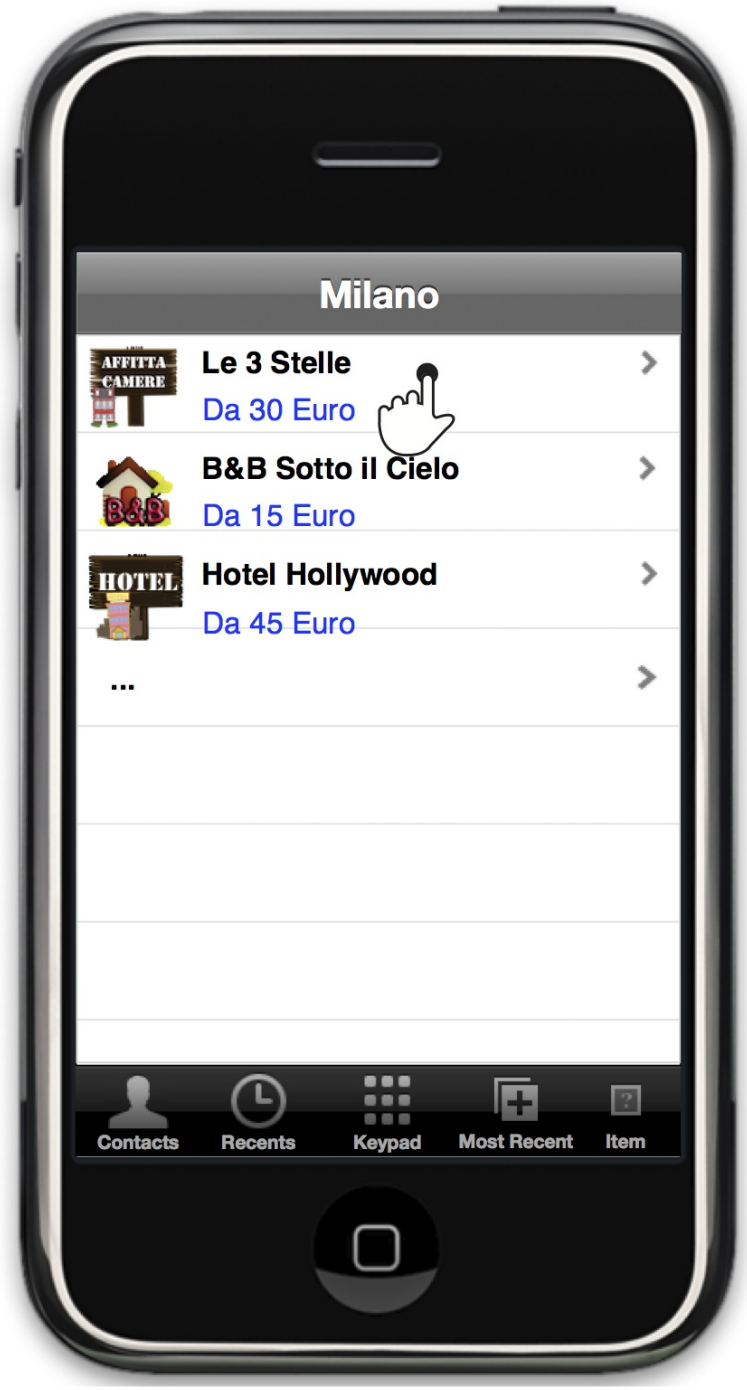


Figura 6.13: Mock-up: Bed and breakfast

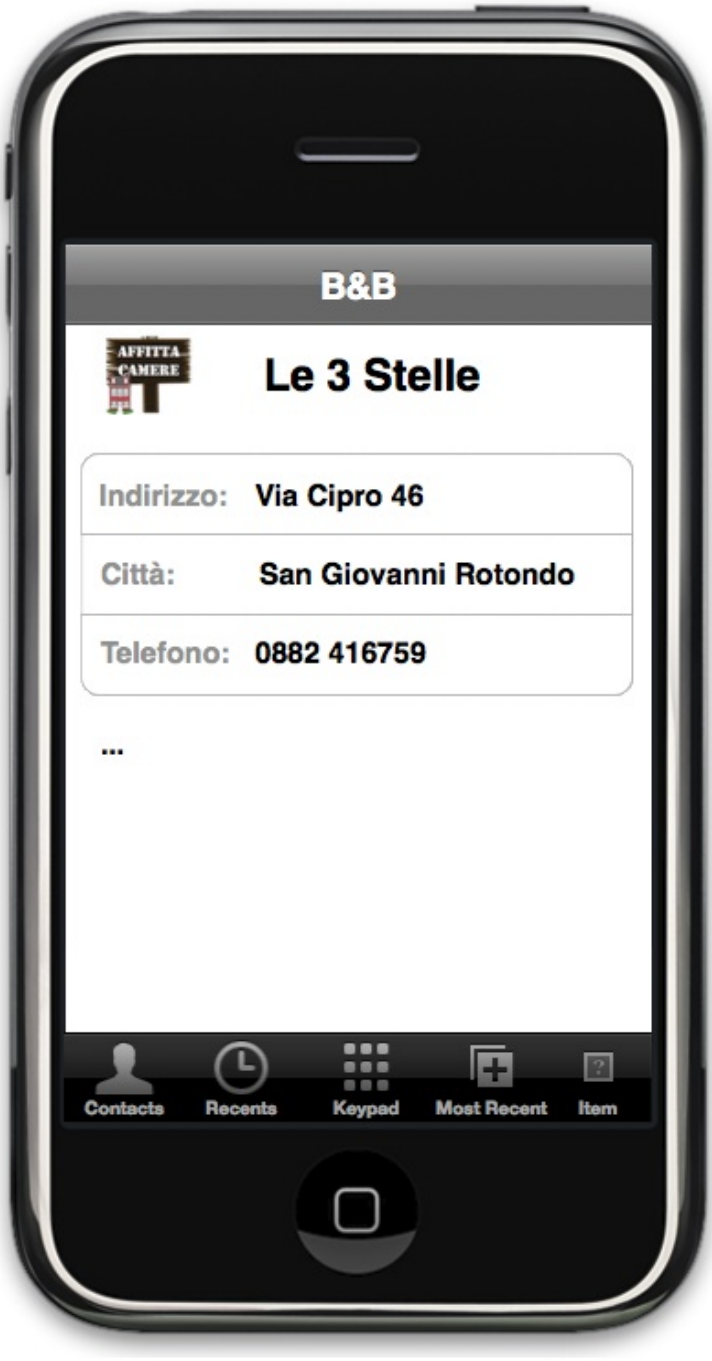


Figura 6.14: Mock-up: dettaglio Bed and breakfast

6.5 Design

6.5.1 Modello di navigazione

Per meglio modellare i percorsi di navigazione che determinano il modo in cui gli utenti interagiscono con CoolTraveling, si usa il diagramma UX (User eXperience) in figura 6.15, tramite il quale è possibile definire quelle che saranno le caratteristiche del sistema dal punto di vista dell'utente. Viene quindi rappresentata la superficie di CoolTraveling, ovvero l'insieme di pagine consultabili e i dati in esse contenuti.

6.5.2 Diagramma di analisi

L'obiettivo di questa fase di analisi è di isolare gli elementi di presentazione, di controllo e di interazione con i dati corrispondenti ai livelli logici descritti nel paragrafo relativo all'architettura. Per meglio modellare l'insieme di classi e le relazioni che legano tali componenti viene usato il modello Boundary-Control-Entity che nei nostri diagrammi corrispondono rispettivamente ai colori azzurri-rossi-gialli (vedi figura 6.16).

- *boundary*: indica le interfacce esterne di CoolTraveling verso l'utente, le classi marcate da questo stereotipo offrono metodi di navigazione e presentazione.
- *control*: rappresenta una classe che realizza (interamente o in parte) la logica applicativa di CoolTraveling offerta agli utenti attraverso le interfacce boundary, utilizzando le risorse informative.
- *control*: identifica una classe di accesso ai dati di interesse, le classi control di CoolTraveling sfruttano queste risorse per soddisfare le richieste effettuate dagli utenti attraverso le interfacce boundary. Nelle classi identificate da questo stereotipo sono definiti (1) attributi e (2) metodi, ovvero un sistema per interfacciare e manipolare i dati presenti nella base di dati. Come osservazione si noti che il concetto di entity è molto vicino a quello di Model descritto nel capitolo 3.4.

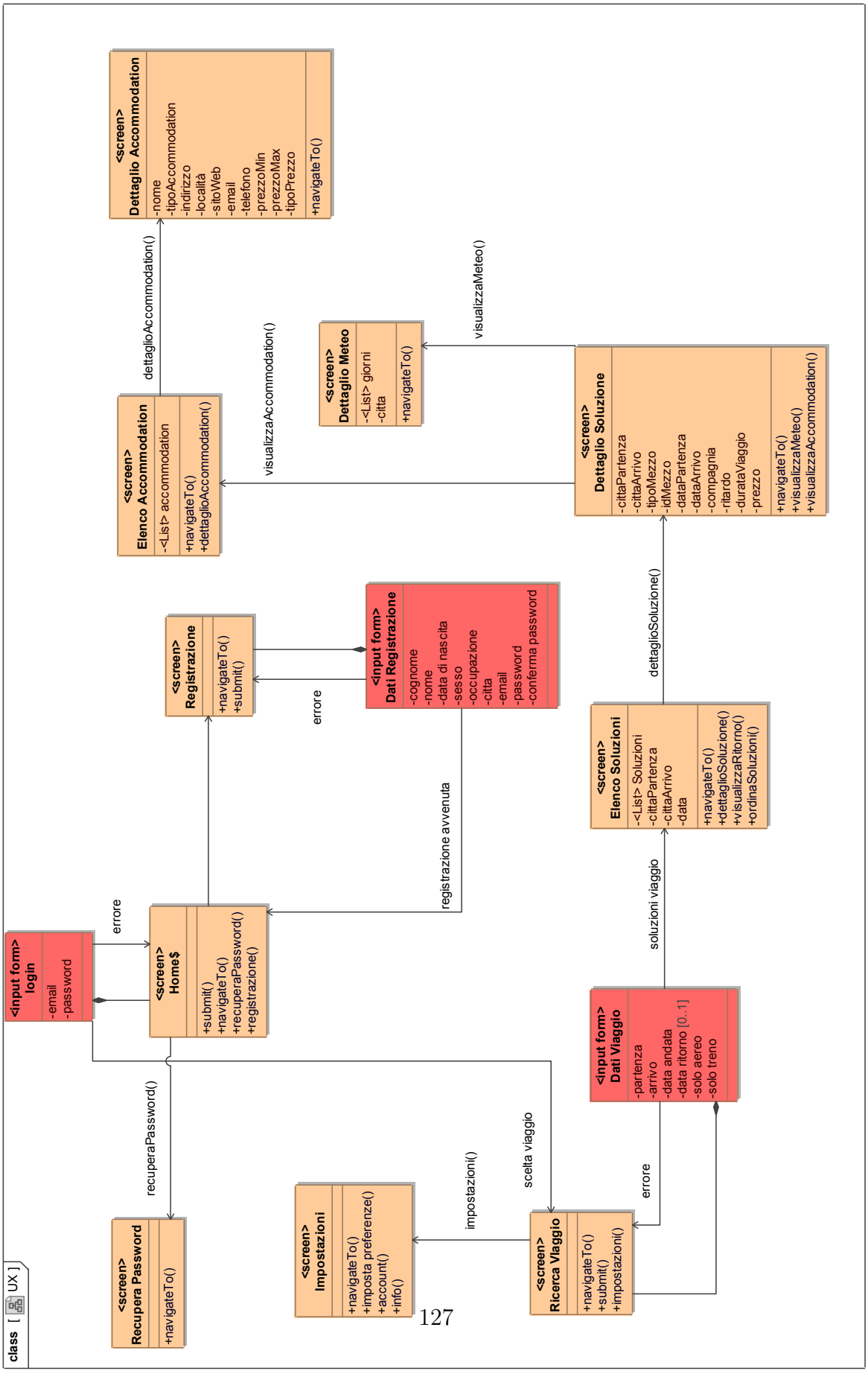


Figura 6.15: Diagramma di navigazione

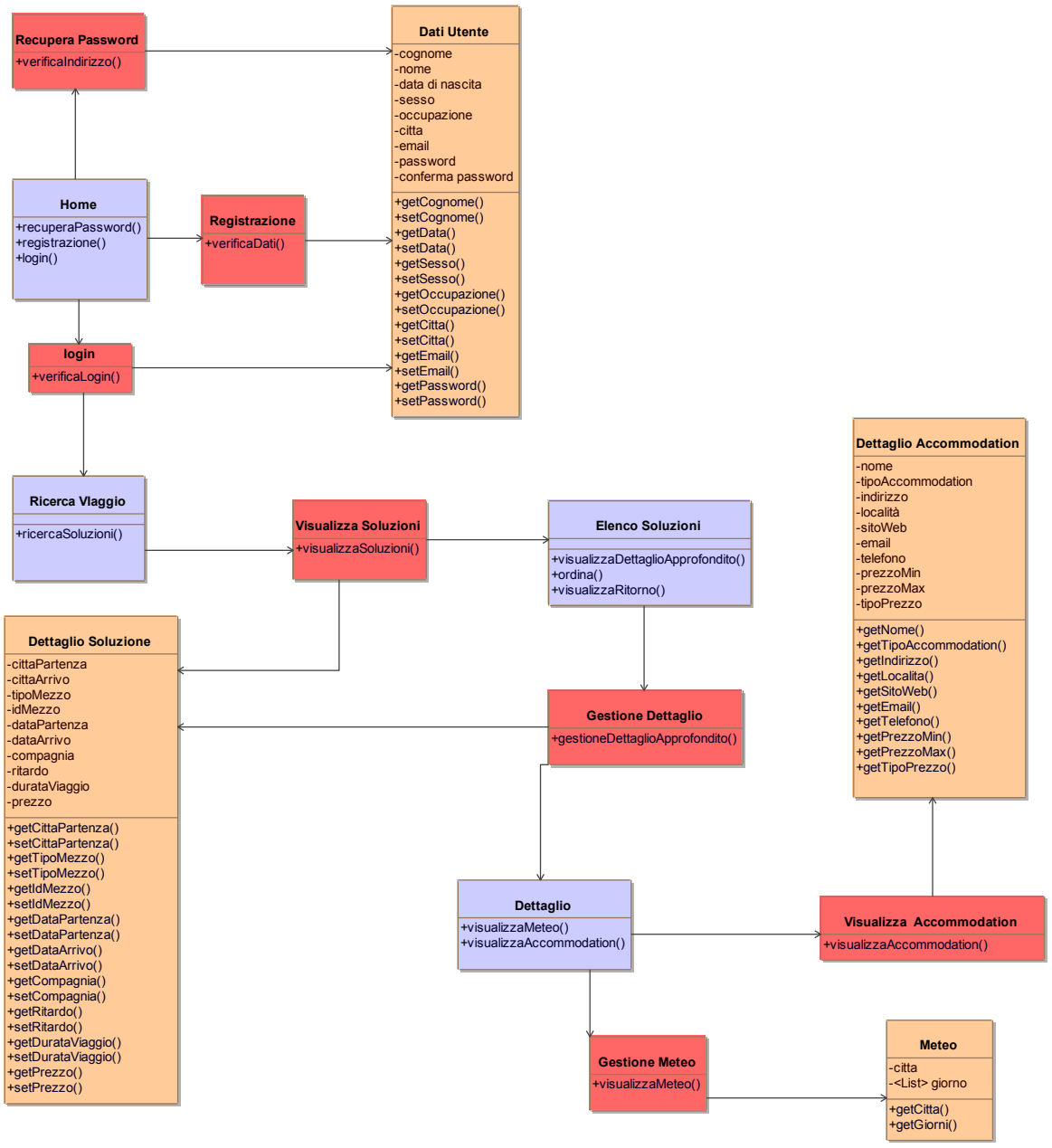


Figura 6.16: Diagramma di analisi

Capitolo 7

Storyboard

In questo capitolo verranno descritte due storyboard che sottolineano le caratteristiche e i punti di forza del progetto CoolTraveling. Il target a cui ci rivolgiamo è mediamente giovane, possessore di uno smartphone e mediamente addicted all'utilizzo di quest'ultimo. CoolTraveling tenta di dare più risposte con una sola richiesta dell'utente, e ciò rappresenta un elemento importante quando si è in mobilità, in quanto il tempo è limitato e le fonti da consultare potrebbero essere numerose e non munite di una adeguata interfaccia mobile. La sezione relativa ai modelli dinamici cercherà di far capire, in maniera esaustiva e visuale, tutte le procedure tecnologiche che si celano durante la descrizione delle attività dell'utente

7.1 Prima storia d'uso

Elio è un agente commerciale in viaggio durante tutta la giornata, spesso si muove con la propria auto ma non disdegna il taxi come ha fatto in mattinata per raggiungere il luogo dell'appuntamento dall'aeroporto. Solitamente i suoi viaggi sono nelle zone limitrofe alla sua sede, nella capitale, ma spesso si reca a Milano per chiudere importanti affari. La società non è molto grande e spesso i costi sui viaggi si fanno sentire. Da oggi Elio, al termine di ogni incontro, avrà un'utile strumento a sua disposizione nella ricerca della stazione e dell'aeroporto più vicini per rientrare verso casa. Sono le 13:30, non ha voglia di pranzare e cerca in CoolTraveling un modo veloce per scegliere il mezzo più economico e veloce con il quale rientrare verso casa.

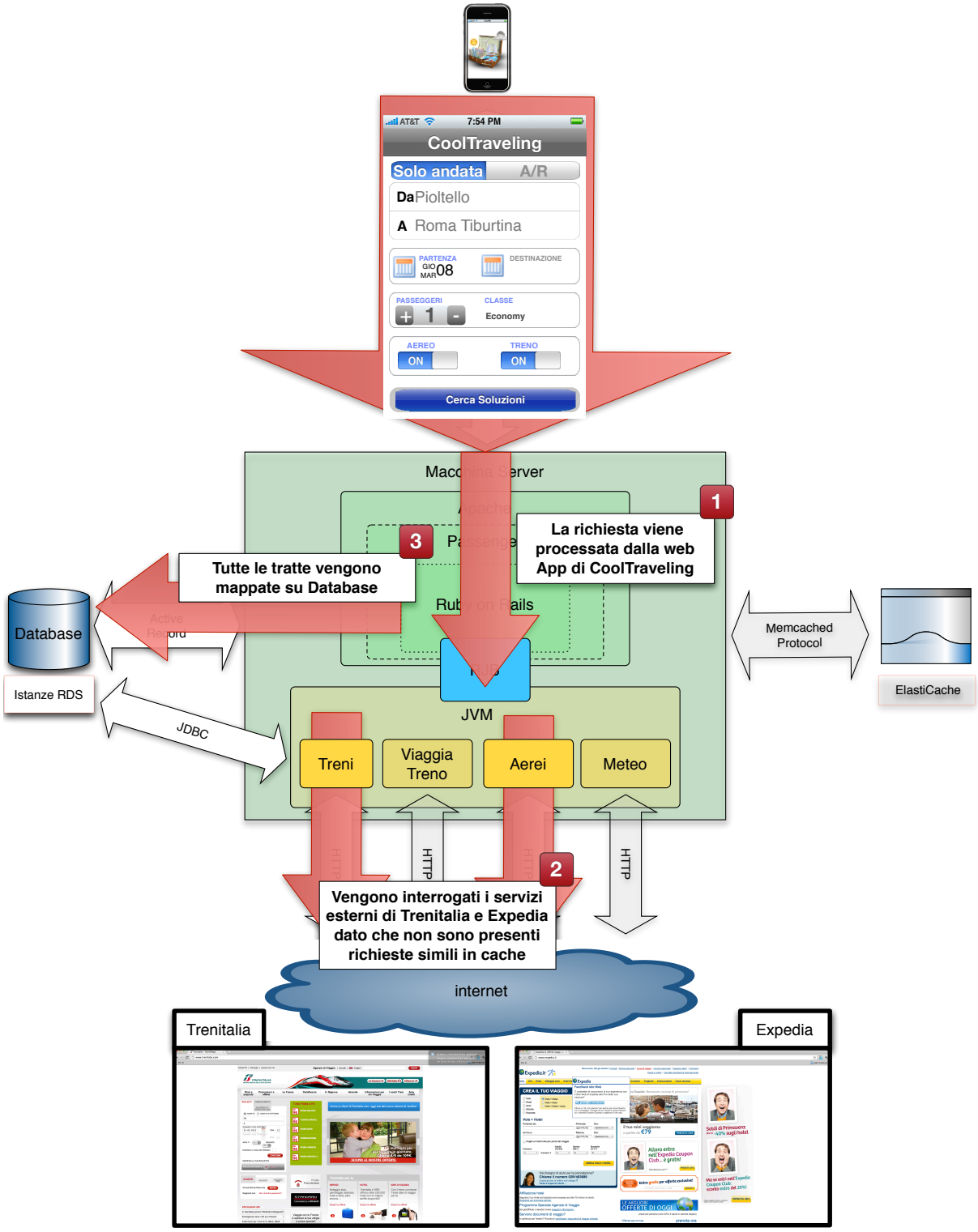
I dati inseriti da Elio sono i seguenti:

- *Città di partenza* **Pioltello**
- *Città di destinazione* **Tiburtin**
- *Data di partenza* **08/03/2012 13:30**

Dai dati inseriti, il sistema di validazione(descritto precedentemente), conferma l'esattezza del comune di Pioltello, mentre suggerisce ad Elio di verificare la città di destinazione inserita tra le seguenti opzioni: **Roma Tiburtina**, **Tiburtin** e **altre città** per immettere nuovamente un'altra città. Accortosi dell'errore, Elio provvede a modificare la città di destinazione scegliendo Roma Tiburtina e consentendo la ricerca con entrambi i mezzi di trasporto. Dalle soluzioni proposte ordinate per prezzo, 12 treni e 15 aerei. Elio esclude sin da subito gli aerei perché grazie al filtro di ordinamento in funzione del prezzo si accorge che gli aerei costano tre volte tanto il prezzo del viaggio in treno. Visto il confronto, circa 300 euro della soluzione aerea contro i 90 euro della soluzione via treno Elio è molto felice di aver consultato CoolTraveling e si stupisce ancora di più quando scopre che Pioltello ha anche anche una stazione ferroviaria molto vicina. Fa un rapido check sulle tempistiche e vede che in circa tre ore di viaggio potrà essere già a casa. Decide quindi di approfondire la soluzione di viaggio identificata proseguendo nei dettagli del viaggio. Dopo aver verificato che sia tutto in linea con le sue aspettative Elio procede all'acquisto del biglietto dirigendosi sul sito Trenitalia direttamente da CoolTraveling. Elio si reca quindi in stazione parte verso casa sicuro di ave trovato un valido assistente di viaggio per il futuro!

Flusso dati

Come descritto nella Figura 7.1 e nei collaboration diagram in Figura 7.12, 7.13 e 7.14 quando l'utente immette dei dati di ricerca per una soluzione di viaggio si avviano molteplici processi. Assumiamo innanzitutto che l'utente sia già registrato e loggato all'interno dell'applicazione. A questo punto inizierà la fase di inserimento dei dati. Nella nostra storia d'uso Elio inserisce correttamente la prima città e sbaglia l'inserimento della seconda troncando l'ultimo carattere, evento molto sovente quando si scrive da un mobile device. Nella nostra architettura il primo modulo che prende in consegna la richiesta, ancora prima che venga processata è il modulo dedicato alla verifica della correttezza delle città inserite. Dopo questa verifica la richiesta prosegue e viene realmente processata, ovvero vengono istanziati i wrapper esterni per la richiesta



WebServices e siti Web Esterni

Figura 7.1: Flusso dati nell'architettura

delle soluzioni di viaggio sui rispettivi servizi. Nel caso specifico verranno istanziati due moduli, il modulo per la ricerca treni e il modulo per la ricerca degli aerei. Poco prima di questo processo per ciascuno dei moduli viene identificato l'aeroporto e la stazione più vicina alle città selezionate dall'utente. In questo modo anche le ricerche con comuni italiani o esteri non dotati di stazioni o aeroporti potranno restituire risultati alle ricerche. Quando i moduli wrapper terminano il loro operato e quindi sono state trovate ed elaborate delle soluzioni di viaggio il controllo torna all'applicazione principale, la quale si preoccuperà di mappare nel modello logico dei dati le soluzioni di viaggio trovate dai singoli moduli esterni mettendole a fattor comune. Completato questo processo i dati sono pronti per essere ordinati in funzione di diversi criteri, quali: partenza, durata e costo della soluzione di viaggi,o e possono quindi essere presentati all'utente. Solamente a questo punto l'utente riceve un feedback visivo da parte dell'applicazione la quale mostrerà a lui le soluzioni di viaggio identificate. L'utente, grazie alla normalizzazione introdotta dall'applicazione sarà in grado di confrontare i dettagli delle soluzioni di viaggio di diversi mezzi di trasporto potendoli ordinare tra loro in funzione dei parametri citati in precedenza. A questo punto Elio è in grado di selezionare il mezzo di trasporto che più si addice alle sue esigenze, nel caso specifico il treno, finalizzando direttamente l'acquisto grazie ad un collegamento diretto con il sito di Trenitalia. La richiesta di viaggio effettuata da Elio viene quindi registrata dal sistema su database in modo che futuri utenti possano beneficiarne.

7.2 Seconda storia d'uso

Fabiana è a pranzo con le sue amiche come spesso è solita fare, è una ragazza solare, ha 18 anni e oggi intorno alle 15:00 prenderà un treno che la porterà a Roma per 2 giorni. Ha deciso di rimanere un giorno in più per godersi le meraviglie della città. Vorrebbe avere conferma dell'ora esatta di partenza del treno, controllare il meteo della settimana e trovare una buona accommodation spendendo poco. Purtroppo il web non è ancora pronto per il mobile e molti siti turistici non sono aggiornati; una sua amica le suggerisce di usare CoolTraveling, in modo da poter velocemente risolvere tutti i suoi quesiti.

I dati inseriti da Fabiana sono i seguenti:

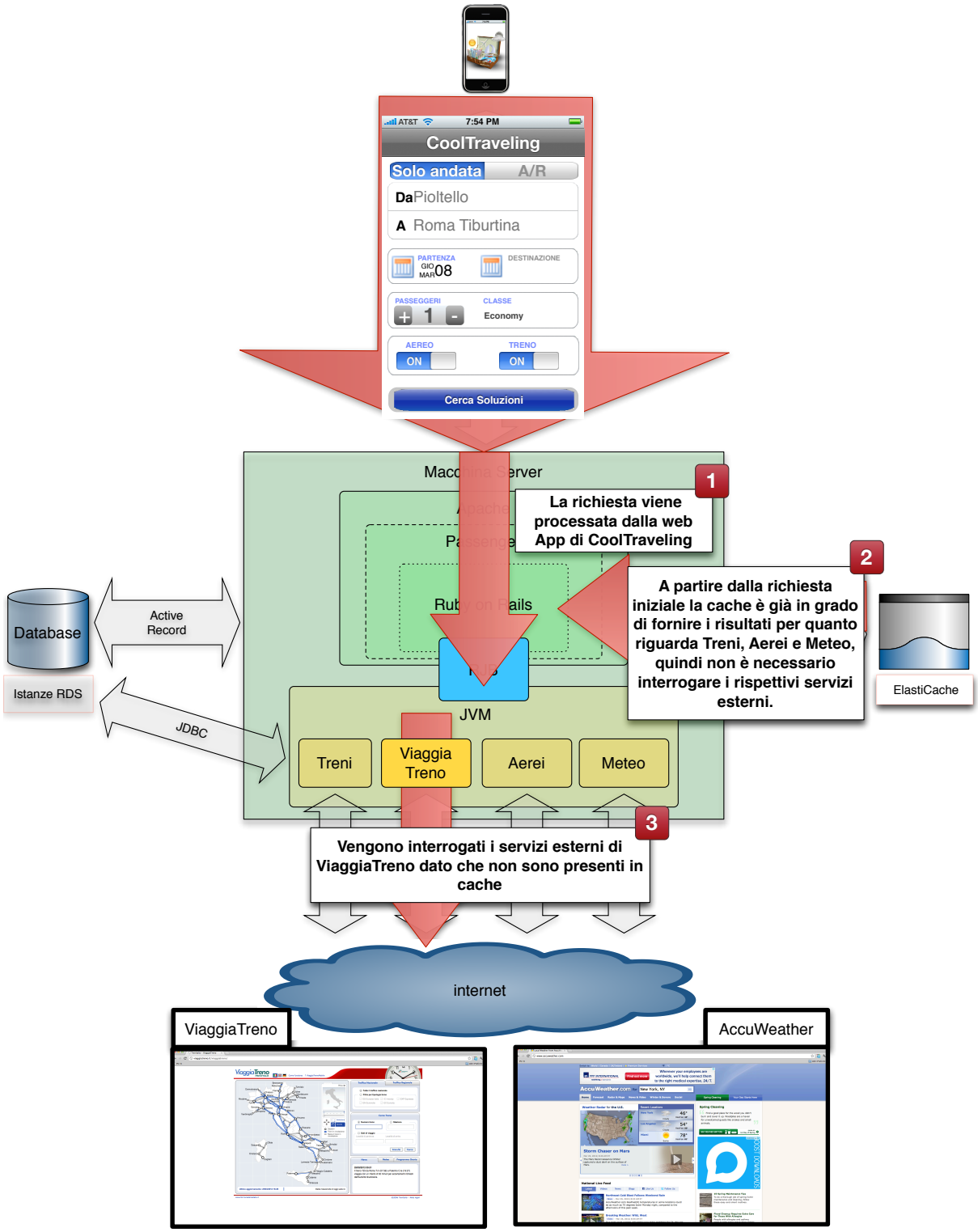
- *Città di partenza* **Pioltello**

- *Città di destinazione* **Roma**
- *Data di partenza* **08/03/2012 14:00**

La richiesta di Fabiana è già stata effettuata mezz'ora prima da Elio, sempre attraverso CoolTraveling, di conseguenza è presente e pronta per essere inviata immediatamente attraverso il sistema di caching implementato. In pochissimo tempo Fabiana è stata in grado di consultare la sezione di dettaglio del suo treno con la possibilità di poter, in seguito, visualizzare eventuali ritardi poco prima della partenza. A partire dal dettaglio, Fabiana ha avuto la possibilità di consultare le condizioni meteorologiche sia diurne che notturne, scoprendo con sua meraviglia che le aspettavano delle splendide giornate in giro per la capitale. Sempre dal dettaglio ha potuto consultare tutte le tipologie di accommodation presenti in città, scegliere quella che più si avvicinava alle sue esigenze ed effettuare una *click to call* in modo da verificare le disponibilità.

Flusso dati

Come descritto nella Figura 7.2 e nei collaboration diagram in Figura 7.12, 7.17, 7.16 quando l'utente immette dei dati di ricerca per una soluzione di viaggio si avviano molteplici processi. In questo caso Fabiana effettua una ricerca molto simile a quella effettuata precedentemente da un'altro utente. Ogni richiesta effettuata viene sempre mappata su database e mantenuta in cache per un certo periodo di validità della stessa. Grazie a questa feature una volta che Fabiana inserisce i dati di ricerca CoolTraveling si rende subito conto della presenza di soluzioni di viaggio pertinenti e non effettua alcuna chiamata esterna ai servizi di treni e aerei ma si preoccupa solamente di recuperare i dati di accommodation, meteo e viaggiatreno per verificare se i treni già precedentemente ricercati siano o meno in ritardo. Se la richiesta di Fabiana fosse invece pervenuta qualche istante dopo il periodo di validità della cache sarebbero state effettuate comunque le richieste esterne. Le risposte ricevute dalle richieste esterne riferite a valori in cache scaduti sono utilizzati per maggiorare o ridurre il tempo di validità della cache stessa, compatibilmente con i relativi lower and upper bound. Operativamente viene quindi istanziato solo un wrapper relativo al viaggiatreno in quanto non è verosimile mantenere in cache i valori dei ritardi dei treni. Le accommodation invece essendo catalogate da un processo batch e archiviate su database non necessitano di alcun wrapper simultaneo. Completato il processo di recupero informazioni, i dati vengono elaborati e presentati a Fabiana la quale rimarrà esterrefatta dalla velocità



WebServices e siti Web Esterni

Figura 7.2: Flusso dati nell'architettura

con la quale tutte queste informazioni sono state cercate ordinate e preparate per lei. Fabiana a questo punto non dovrà far altro che selezionare la soluzione di viaggio a lei più adatta e verificare in un'unica interfaccia tempi di percorrenza, partenza, costo, meteo e accommodation relative.

7.3 Modelli Dinamici

Di seguito mostriamo tre tipologie di *Modelli dinamici*[1] in cui risulta esplicito il comportamento dinamico del sistema. I *Modelli Dinamici* si dividono in due parti:

- Diagramma di Interazione:

  Sono diagrammi di comportamento che modellano le interazioni tra varie entità di un sistema, visualizzano lo scambio di messaggi tra entità nel tempo. Il loro scopo è mostrare come un certo comportamento viene realizzato dalla collaborazione delle entità in gioco.
  Come gli altri diagrammi, possono avere vari livelli di astrazione.
  I Diagrammi di interazione si dividono in:

  - Diagrammi di Sequenza (Sequence Diagram): esprimono l'andamento nel tempo di un sistema dinamico. Più in particolare questo tipo di diagramma serve a rappresentare la sequenza dei messaggi scambiati nel tempo fra due o più oggetti. Come si può notare, si tratta di un diagramma a due dimensioni, quella verticale è il tempo, mentre in orizzontale vengono posti i vari oggetti considerati. Per ognuno di essi viene disegnata una linea verticale tratteggiata che rappresenta la sua linea della vita.
  - Diagrammi di Collaborazione (Collaboration Diagram): descrivono la struttura degli oggetti che si scambiano i messaggi.
    La comunicazione tra i diversi oggetti avviene tramite Messaggi che contengono al loro interno le informazioni relative ad un'azione da compiere.

- Diagramma di Comportamento: descrivono il comportamento dinamico di un gruppo di oggetti che interagiscono per risolvere un problema. A loro volta si dividono in:

[1] I modelli dinamici descrivono il comportamento del sistema in funzione del tempo.

- Diagrammi di Attività (Activity Diagram): definisce le attività da svolgere per realizzare una data funzionalità. Con il diagramma delle Attività si possono definire una serie di attività o di flussi, anche in termini di relazioni tra le attività.
- Diagrammi degli Stati (Statechart Diagram): descrivere il comportamento delle entità o delle classi in termini di stato (macchina a stati).
  Il diagramma mostra gli stati che sono assunti dall'entità o dalla classe in risposta ad eventi esterni. Il concetto di stato è spesso posto in relazione a ciclo di vita; l'insieme completo di stati che un'entità o una classe può assumere, dallo stato iniziale a quello finale, ne rappresenta il ciclo di vita.

7.3.1 Diagramma di Interazione: Diagrammi di Sequenza

I seguenti esempi di *diagrammi di sequenza*[2] illustrano le interazioni di un utente col sistema.

Ricerca dell'utente

Con questo diagramma (Vedi Figura 7.3) si vuole rappresentare l'interazione che avviene tra l'utente dell'applicazione CoolTraveling e il server per effettuare una ricerca di un viaggio e le eventuali richieste di dettaglio.
Vista la complessità nel visualizzare tutto il processo, in questo grafo abbiamo rappresentato dei riferimenti a altri grafi (quali: CercaSoluzioni, Richiesta Dettaglio, Richiesta Accommodation, Richiesta Meteo), per poter spiegare meglio nel dettaglio

Ricerca server-side

Con questo diagramma (Vedi Figura 7.4) vogliamo rappresentare ciò che accade nel WebServer quando si riceve una richiesta esterna relativa alla ricerca di soluzioni per un viaggio.
Con l'aiuto della simbologia UML con l'oggetto par vogliamo sottolineare che le ricerche per i treni e per gli aerei vengono eseguite in parallelo per minimizzare il più possibile il tempo di risposta e quindi di diminuire il tempo di attesa per l'utente.
Anche in questo caso abbiamo la necessità di far riferimento ad altri grafici che vedremo tra poco (RichiestaTreni, RichiestaAerei).

[2]Il diagramma di sequenza ha il compito di mostrare tutte le interazioni dinamiche fra gli oggetti.

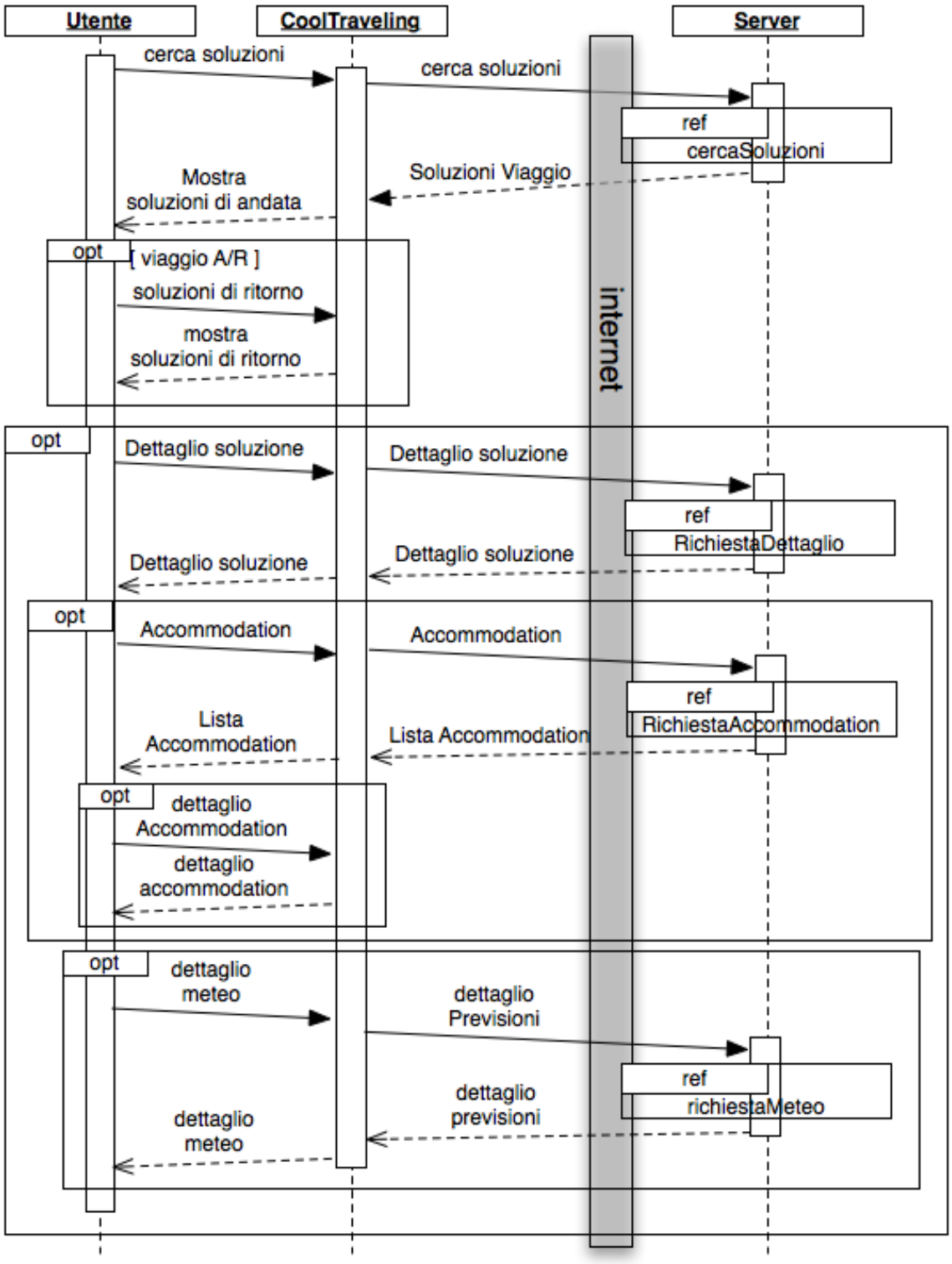


Figura 7.3: Diagramma di Sequenza - Dettaglio principale

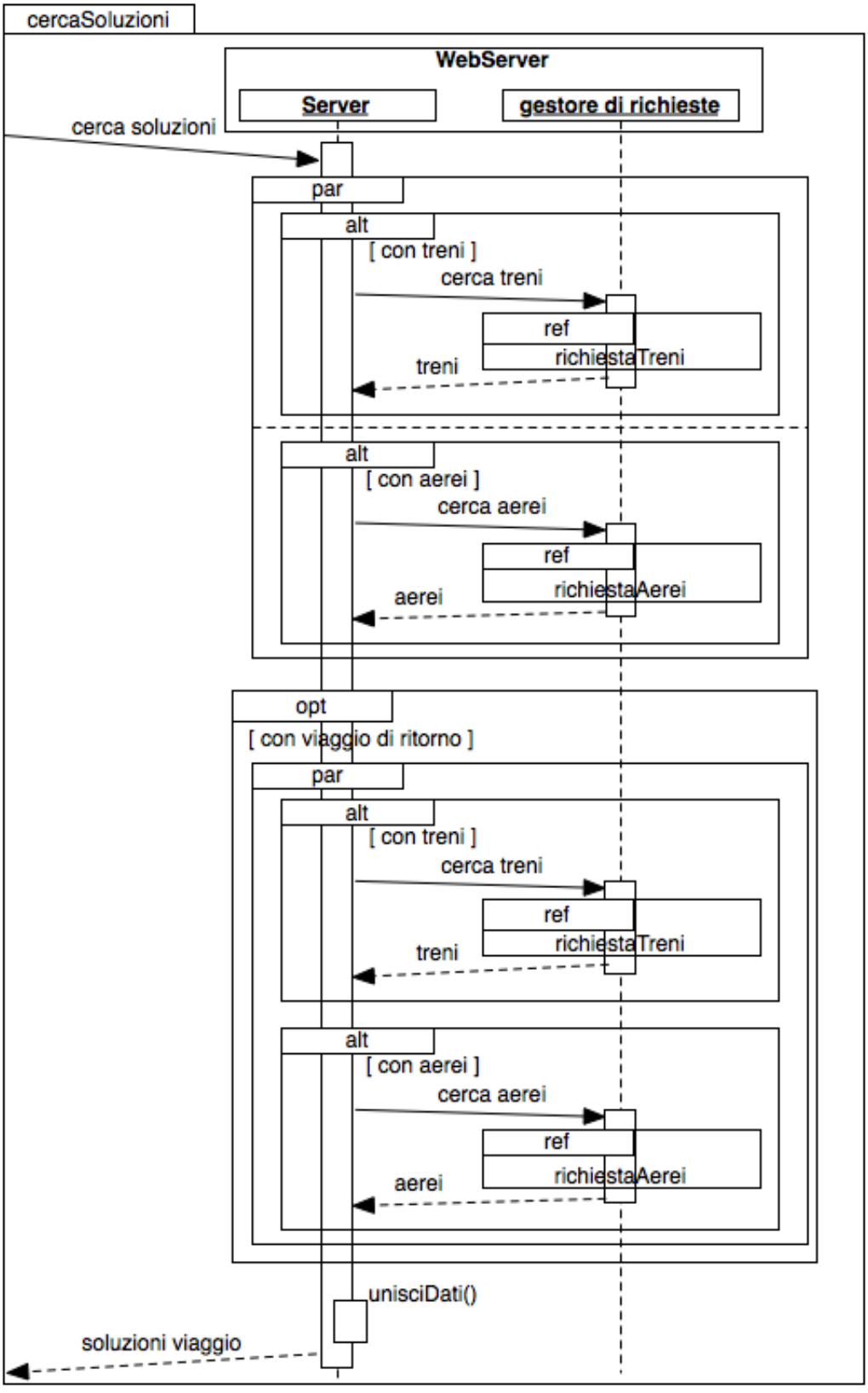


Figura 7.4: Diagramma di Sequenza - Ricerca lato server

Richiesta Treni

In Figura 7.5 si vuole descrivere la sequenza che si verifica alla richiesta di ricerca di viaggi dei treni.
Per questo tipo di ricerca, come primo passo, si verifica che le informazioni richieste siano in già Cache e che abbiano ancora validità. In caso contrario dovremo richiedere al servizio dei treni le informazioni necessarie.
Una volta che il sistema possiede tutte le informazioni necessarie inoltra la risposta al richiedente e salva su DB le informazioni statistiche.

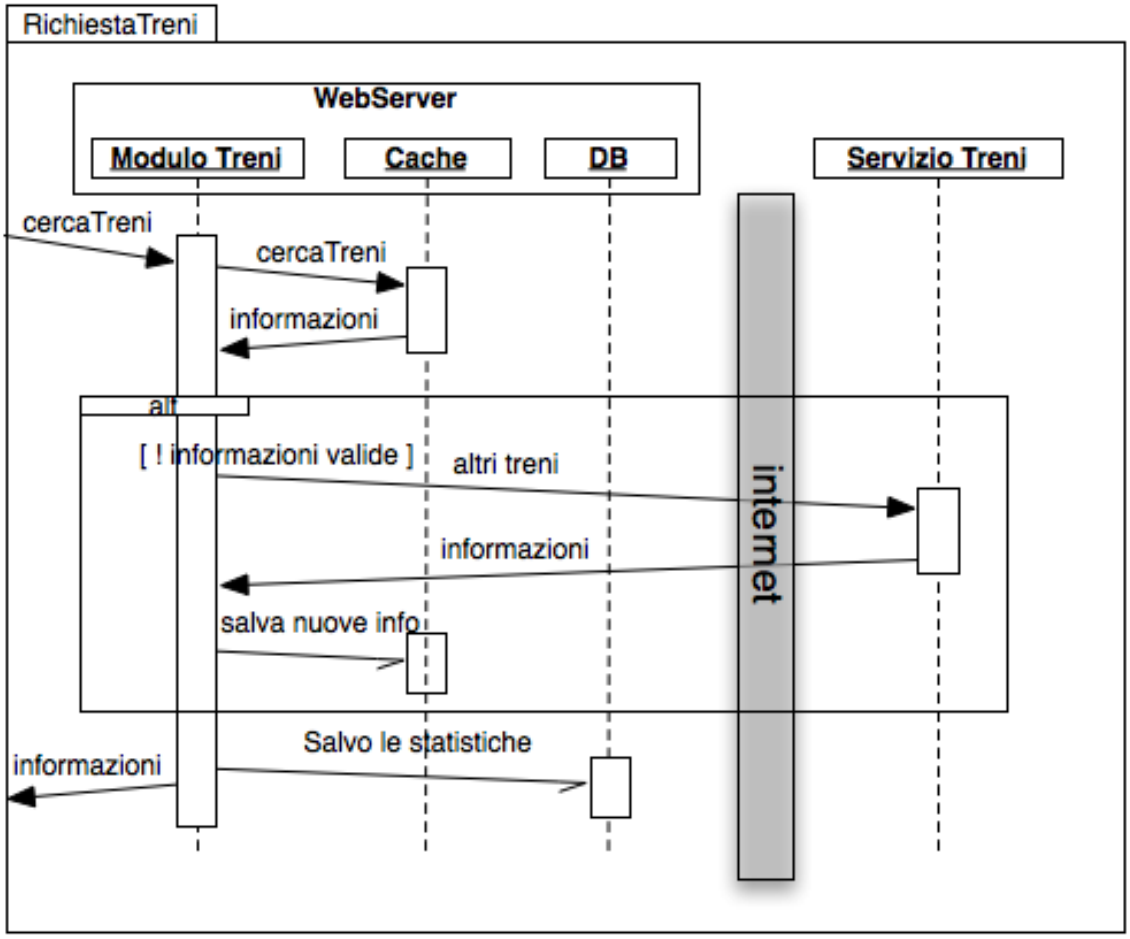


Figura 7.5: Diagramma di Sequenza - Richiesta Treni

Richiesta Aerei

In Figura 7.6 si vuole descrivere la sequenza che si verifica alla richiesta di ricerca di viaggi degli aerei.
Come avviene nel caso dei treni: come primo passo, si verifica che le informazioni richieste siano in già Cache e che abbiamo ancora validità. In caso contrario dovremo richiedere al servizio degli aerei le informazioni necessarie.
Una volta che il sistema possiede tutte le informazioni necessarie inoltra la risposta al richiedente e salva su DB le informazioni statistiche.

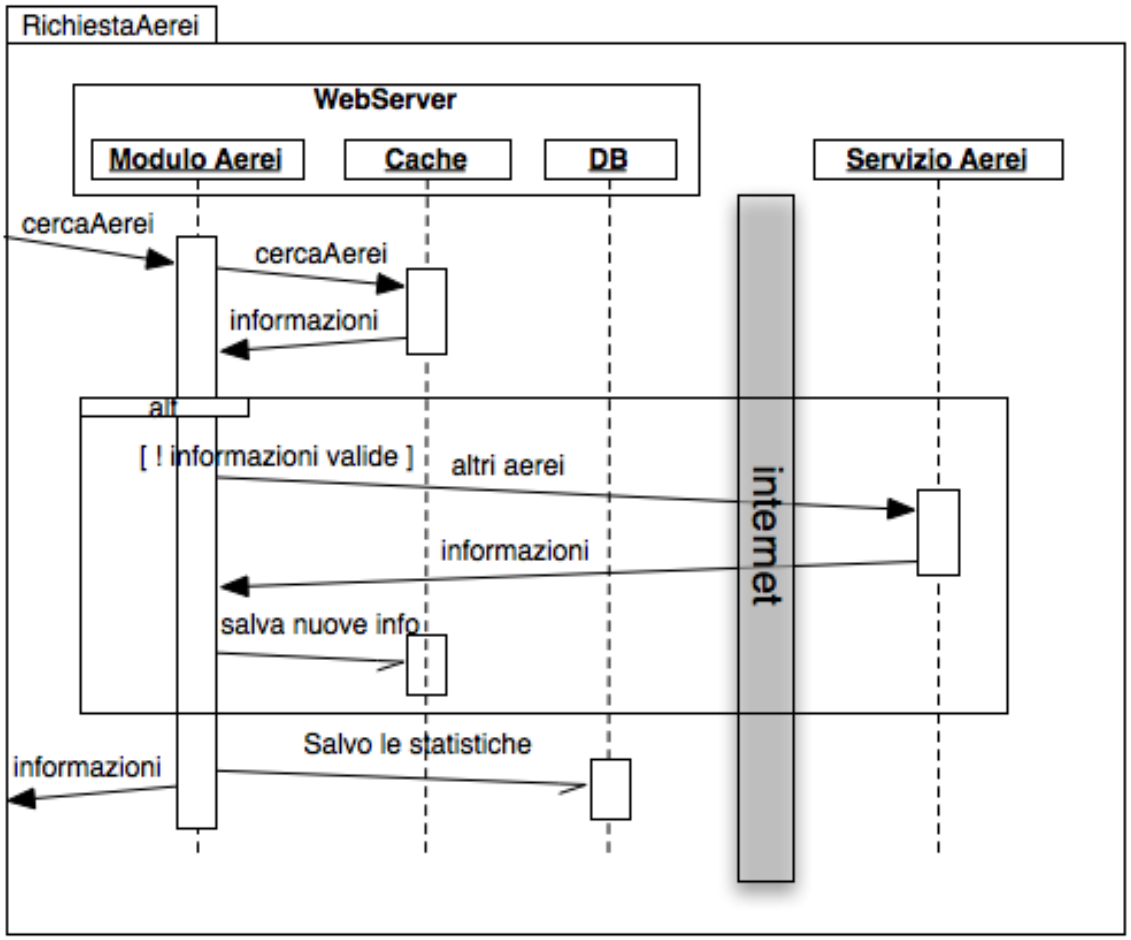


Figura 7.6: Diagramma di Sequenza - Richiesta Aerei

Richiesta Dettaglio

Una volta effettuata la richiesta di viaggio salveremo in Cache tutte le informazioni di dettaglio che l'utente potrebbe chiedere. In Figura 7.7 è possibile vedere il diagramma relativo a questa funzionalità.

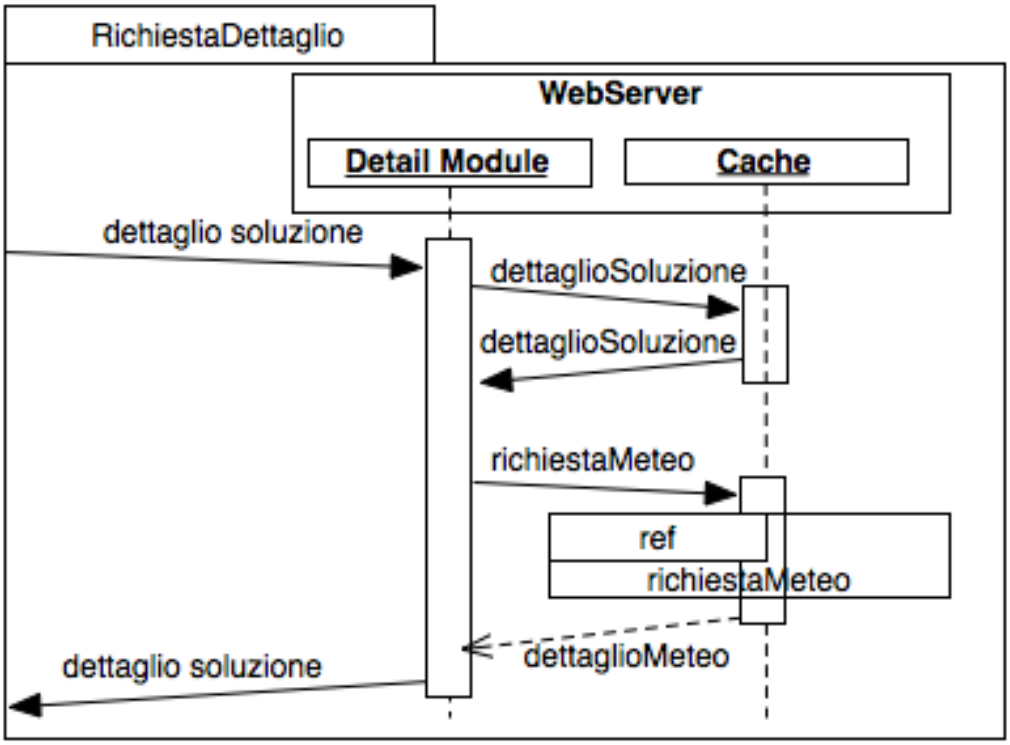


Figura 7.7: Diagramma di Sequenza - Richiesta Dettaglio

Registrazione

Vedi Figura 7.8;

Login

Vedi Figura 7.9;

Richiesta Meteo

Vedi Figura 7.10;

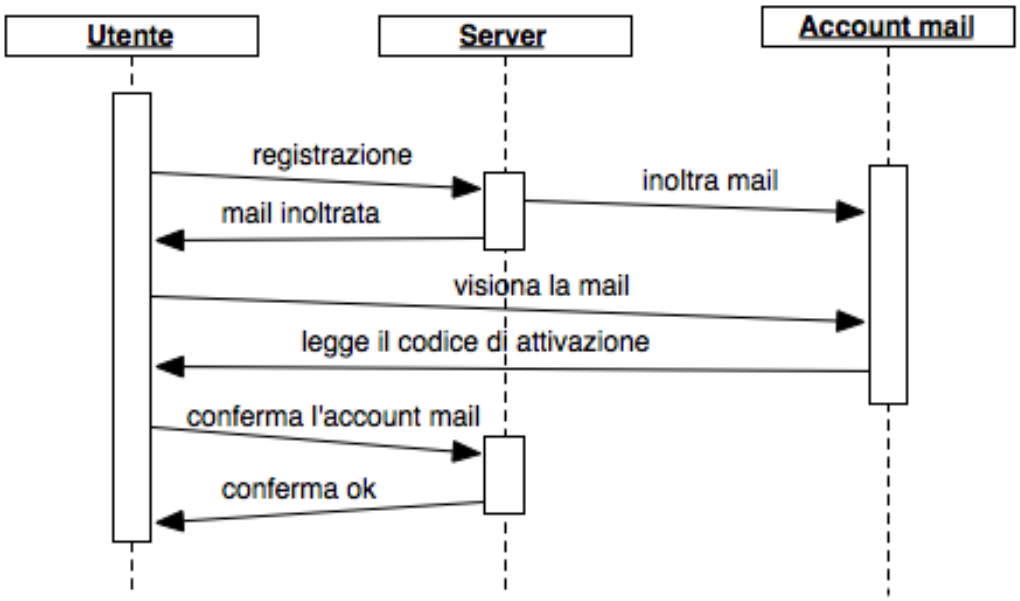


Figura 7.8: Diagramma di Sequenza - Registrazione

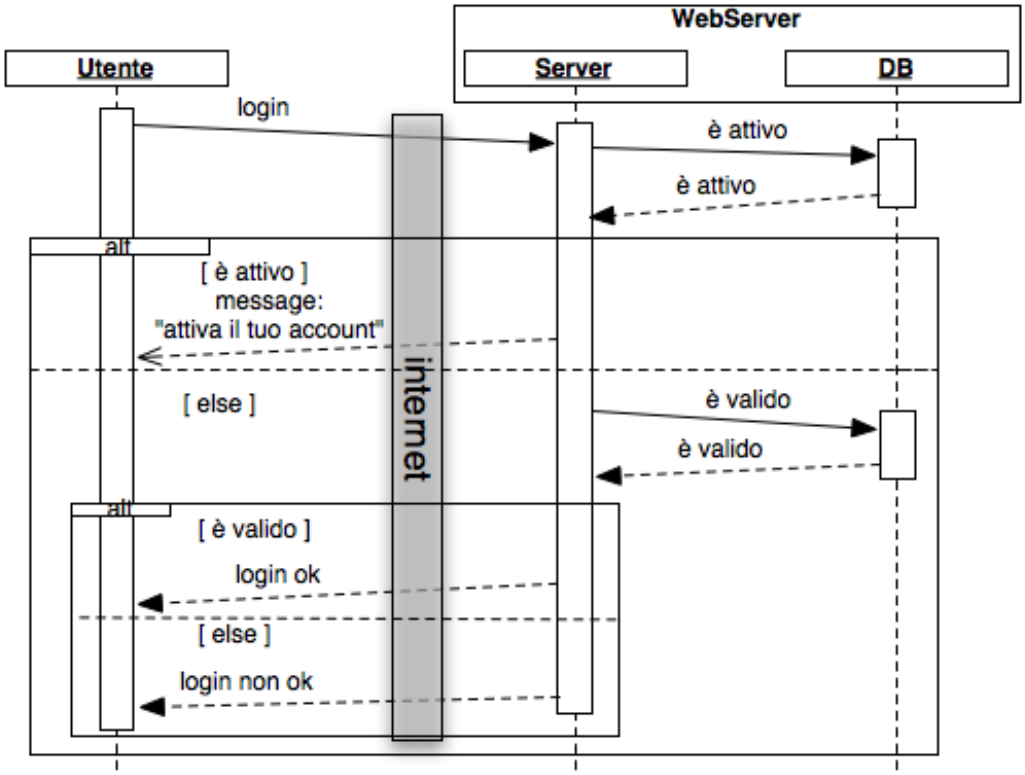


Figura 7.9: Diagramma di Sequenza - Login

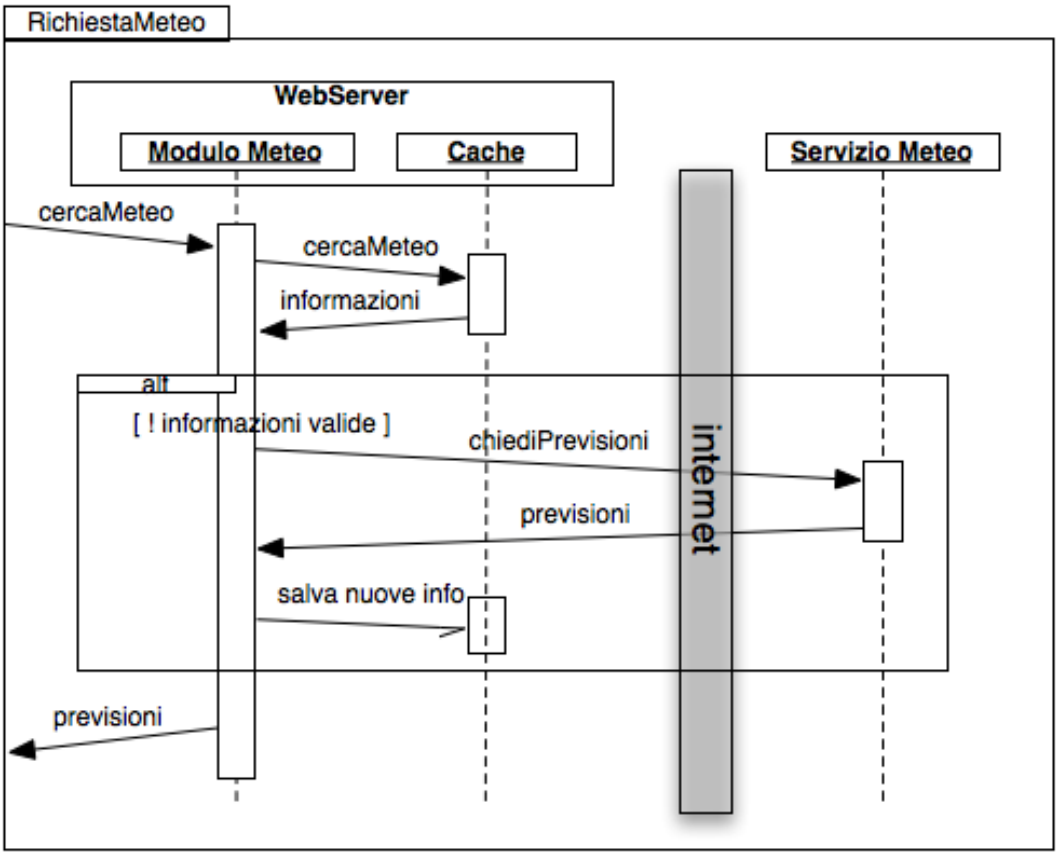


Figura 7.10: Diagramma di Sequenza - Richiesta Meteo

Richiesta Accommodation

Vedi Figura 7.11;

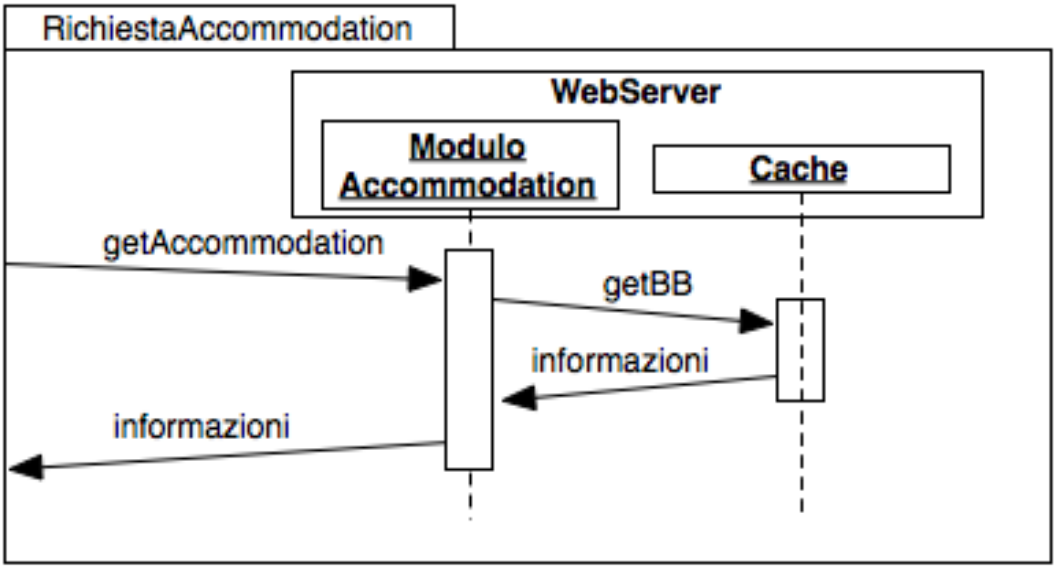


Figura 7.11: Diagramma di Sequenza - Richiesta Accommodation

7.3.2 Diagramma di Interazione: Diagrammi di Collaborazione

I *diagrammi di collaborazione*[3] illustrano le interazioni che avvengono tra gli oggetti che partecipano a una situazione specifica. È più o meno la stessa informazione mostrata nei diagrammi di sequenza, ma nel caso dei diagrammi di sequenza l'enfasi è posta su come le interazioni avvengono nel tempo, mentre i diagrammi di collaborazione mettono in primo piano le relazioni tra gli oggetti e la loro topologia.
Nei diagrammi di collaborazione i messaggi inviati da un oggetto a un altro sono rappresentati da frecce, che mostrano il nome, i parametri e la sequenza del messaggio. I diagrammi di collaborazione sono specialmente adatti a mostrare un particolare flusso o situazione di programma e sono uno dei migliori tipi di diagramma per dimostrare o spiegare rapidamente un processo nella logica del programma.

Ricerca dell'utente

Vedi Figura 7.12;

Richiesta Treni

Vedi Figura 7.13;

Richiesta Aerei

Vedi Figura 7.14;

Ricerca Dettaglio

Vedi Figura 7.15;

Ricerca Meteo

Vedi Figura 7.16;

Richiesta Accommodation

Vedi Figura 7.17;

[3]Il diagramma di collaborazione ha il compito di mostrare tutte le interazioni dinamiche fra gli oggetti.

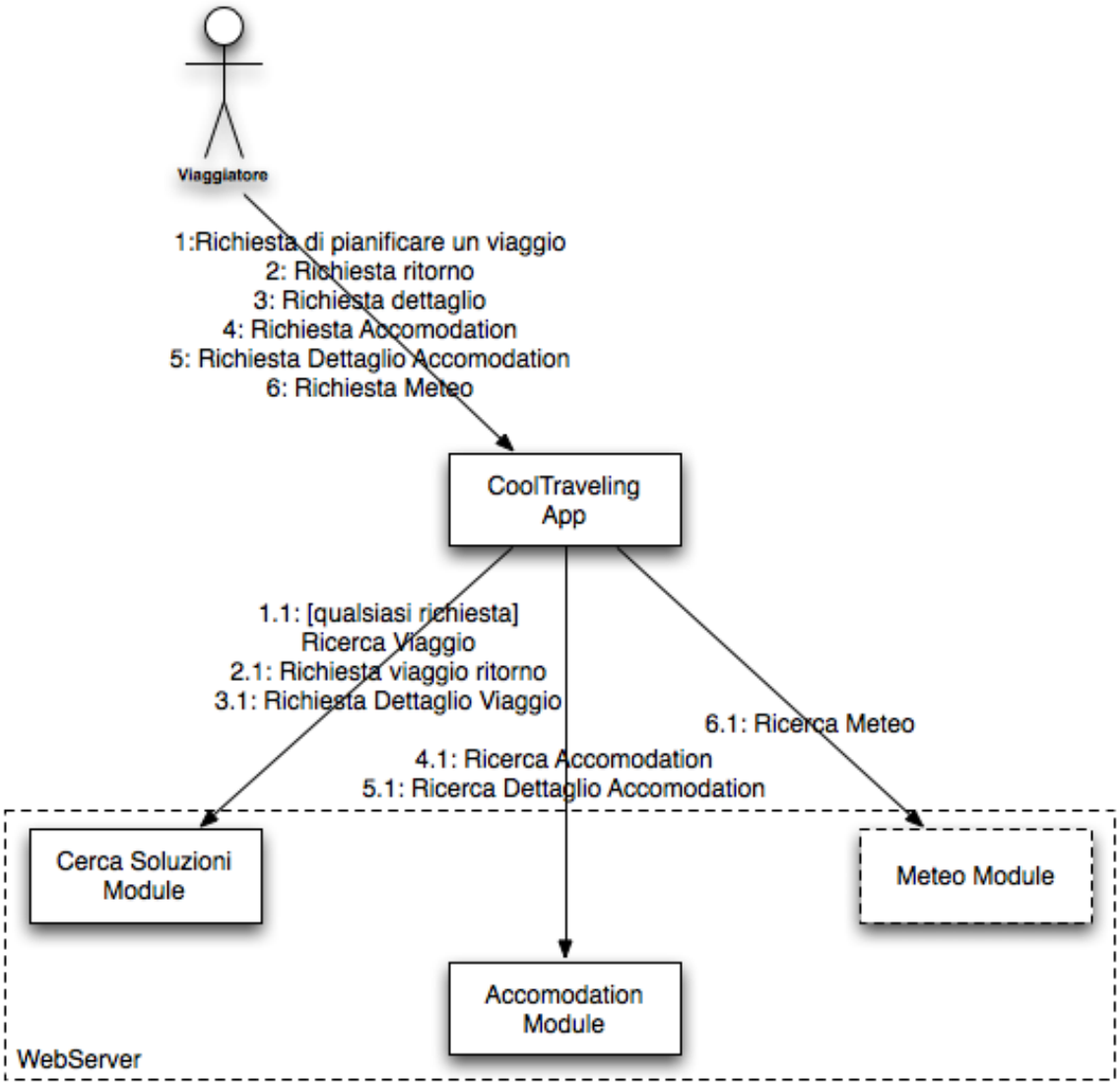


Figura 7.12: Diagramma di Collaborazione - Dettaglio principale

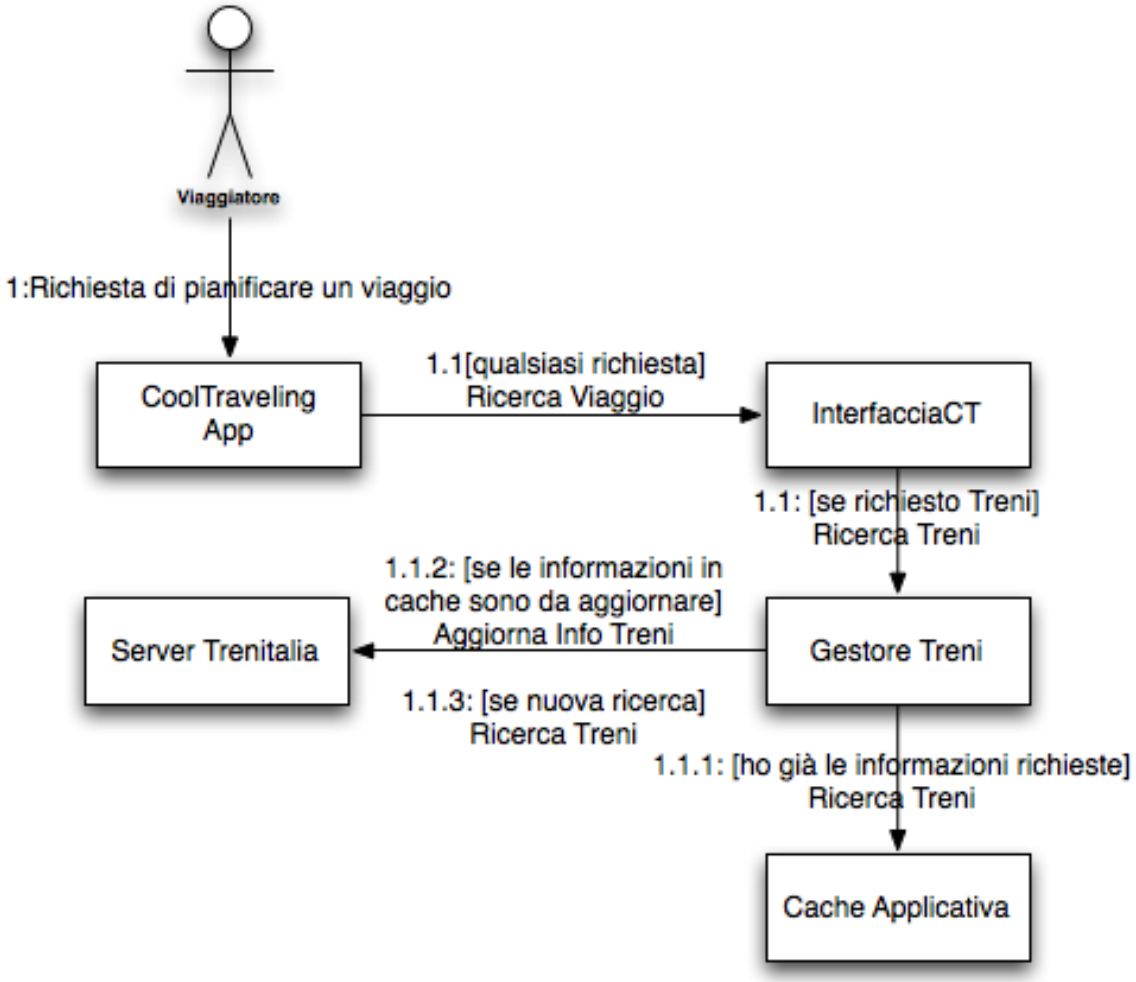


Figura 7.13: Diagramma di Collaborazione - Richiesta Treni

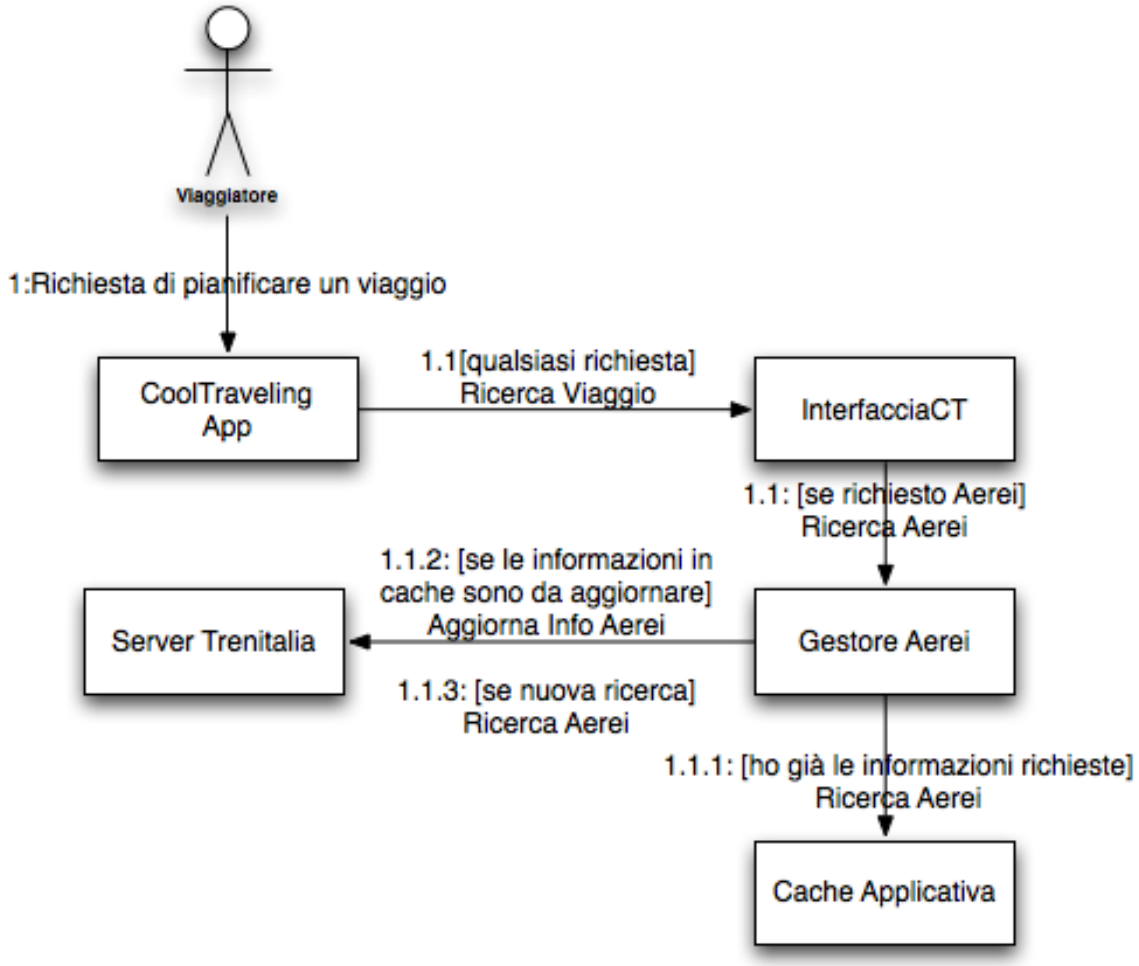


Figura 7.14: Diagramma di Collaborazione - Richiesta Aerei

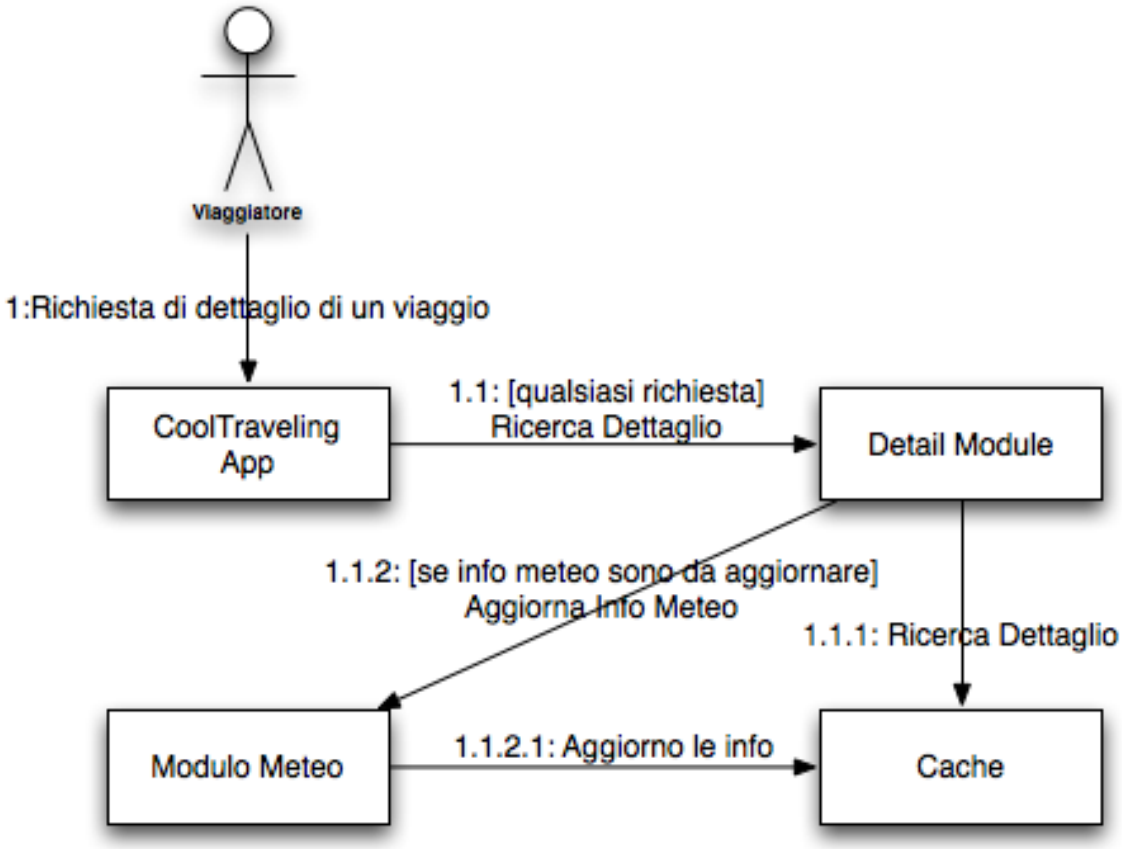


Figura 7.15: Diagramma di Collaborazione - Richiesta Dettaglio

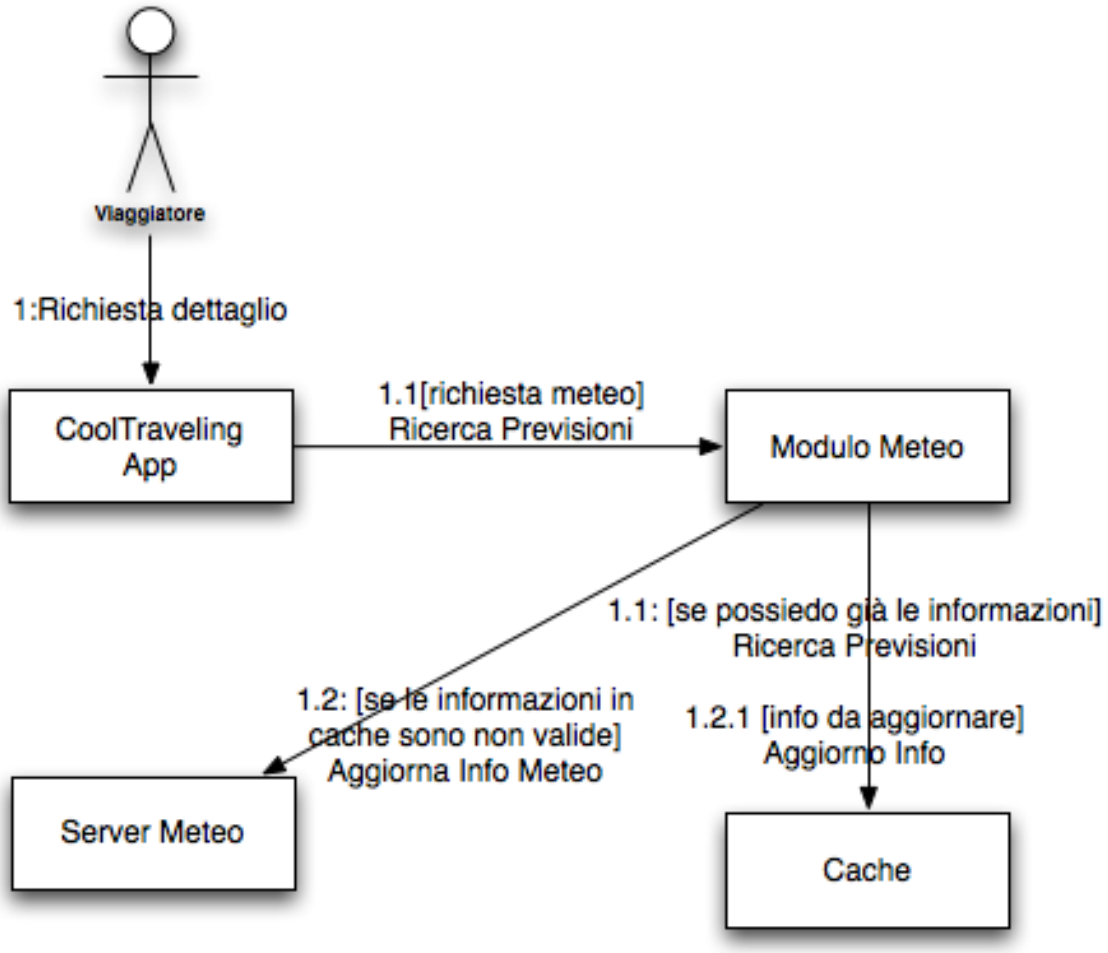


Figura 7.16: Diagramma di Collaborazione - Dettaglio Meteo

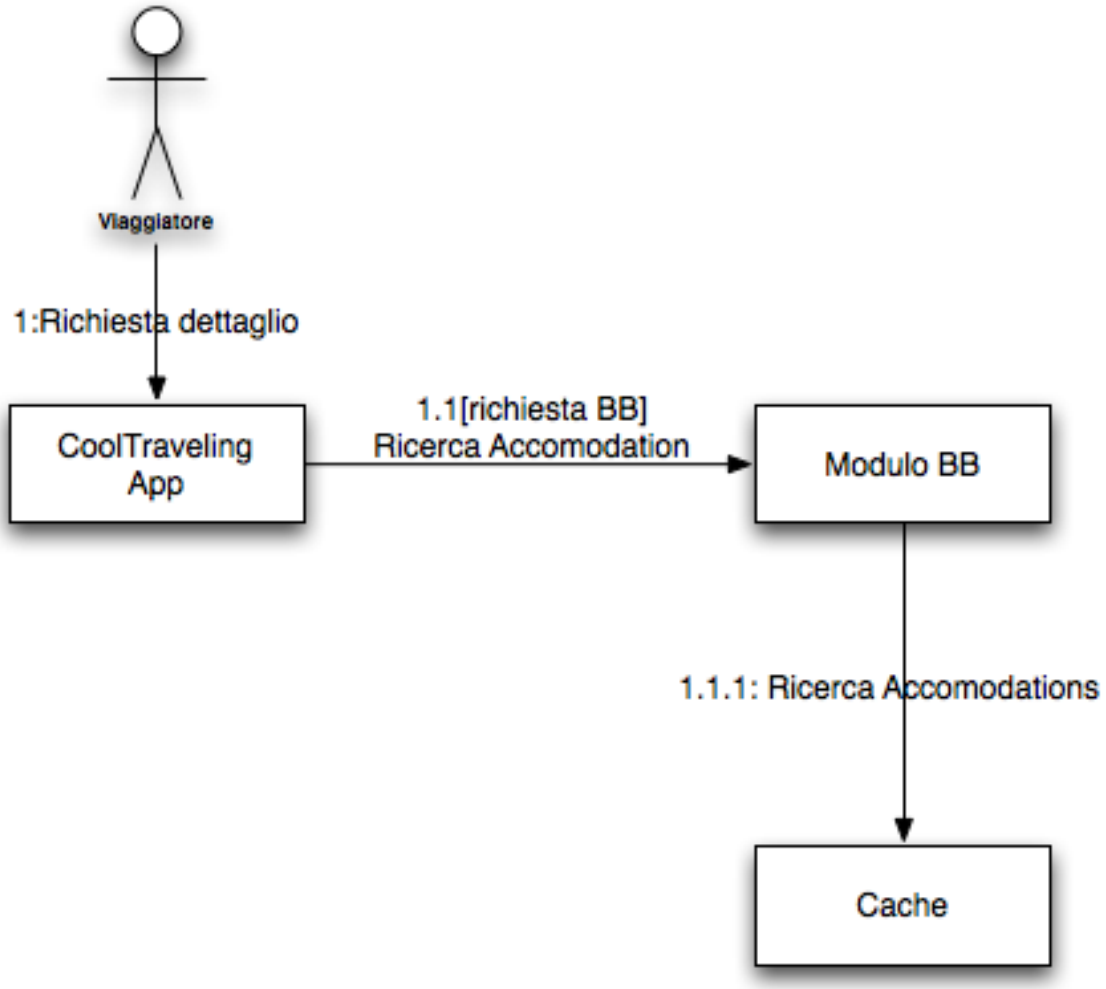


Figura 7.17: Diagramma di Collaborazione - Richiesta Accommodation

7.3.3 Diagrammi di Comportamento: Diagramma di stato

Il seguente *diagramma degli stati*[4] mostrano i vari stati di vita dell'Entità: Utente (Figura 7.18).

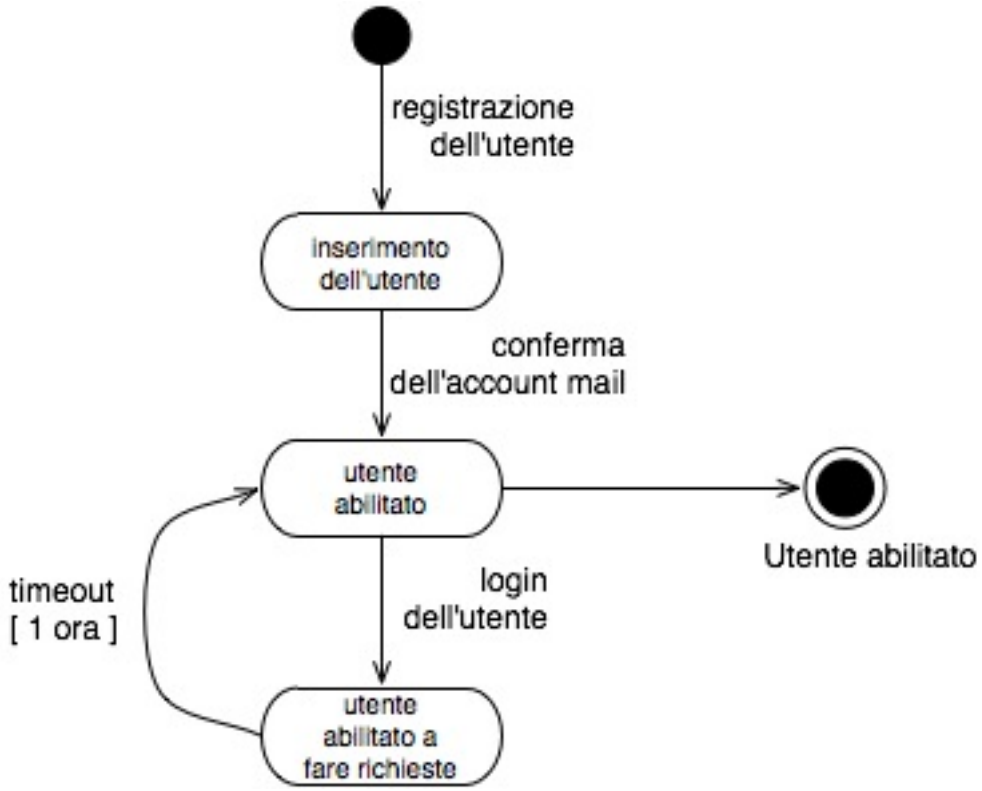


Figura 7.18: Diagramma di stato - Entità Utente

[4]Il *diagramma di stato* ha il compito di relazionare gli stati nei quali si trova un oggetto e iI loro cambiamenti nel tempo.

7.3.4 Diagrammi di Comportamento: Diagramma delle attività

I seguenti *diagrammi delle attività*[5] mostrano i passi per spiegare le funzionalità del sistema.

Cerca Soluzioni

In Figura 7.19 viene mostrato il processo di ricerca di soluzioni di viaggio.

Richiesta Dettaglio

In Figura 7.20 viene mostrato il processo di richiesta dettaglio di un viaggio.

Login

In Figura 7.21 viene mostrato il processo di Login.

Richiesta Accommodation

In Figura 7.22 viene mostrato il processo di richiesta di Accommodation.

[5]Il *diagramma delle attività* ha il compito di rappresentare una sequenza di attività che si riscontrano all'interno di un use case o all'interno del comportamento di un oggetto.

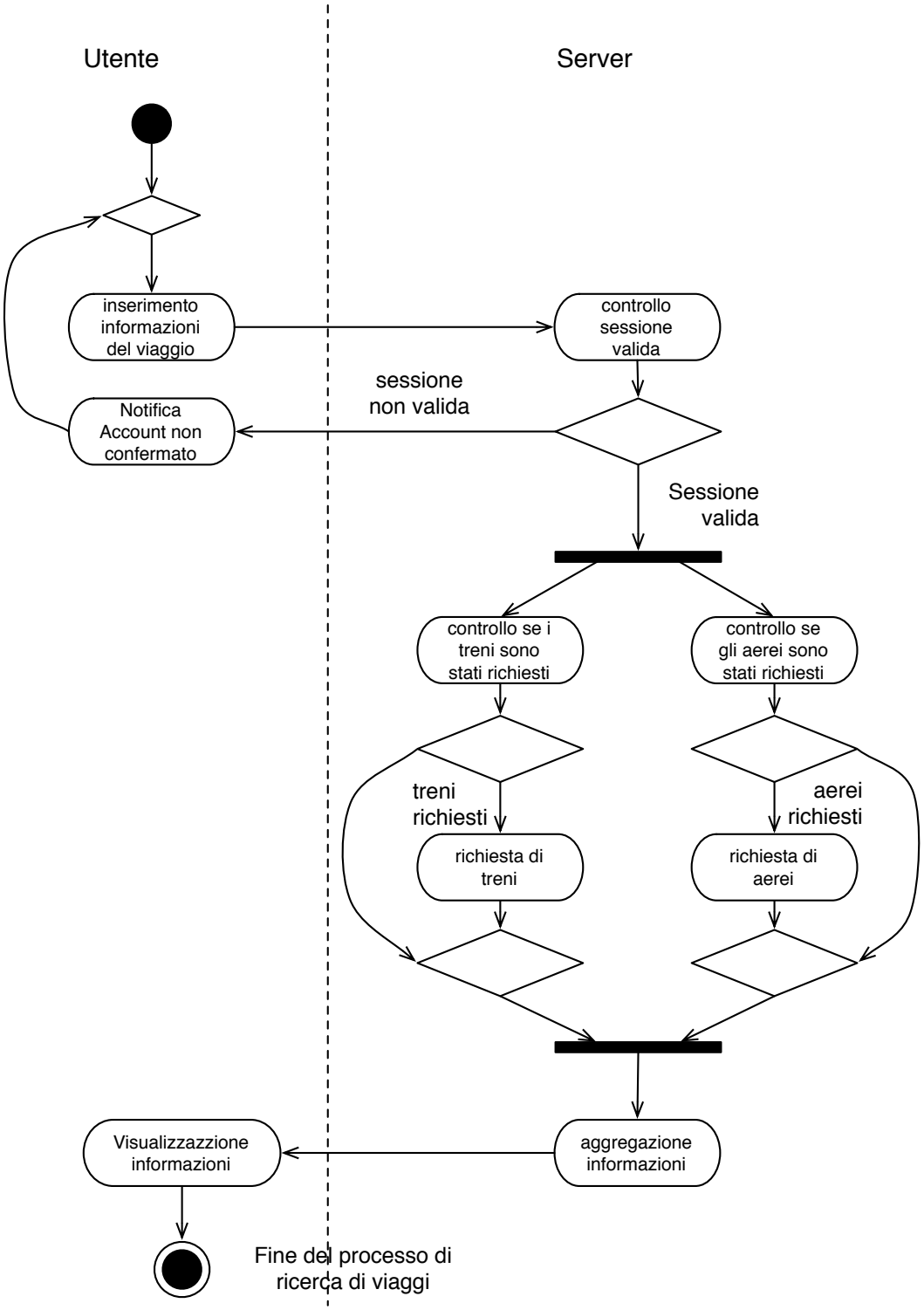


Figura 7.19: Diagramma delle Attività -Cerca Soluzioni

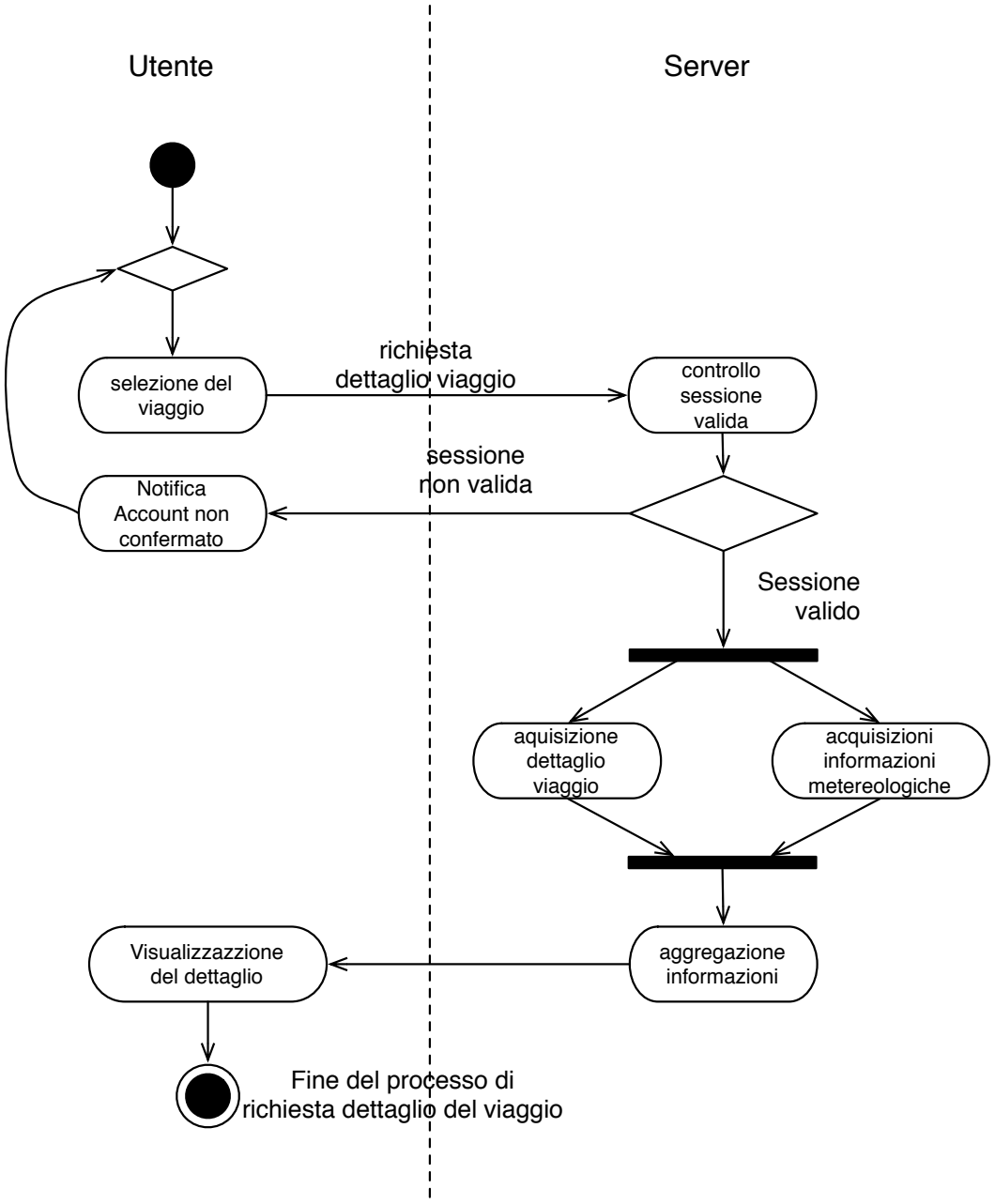


Figura 7.20: Diagramma delle Attività - Richiesta Dettaglio

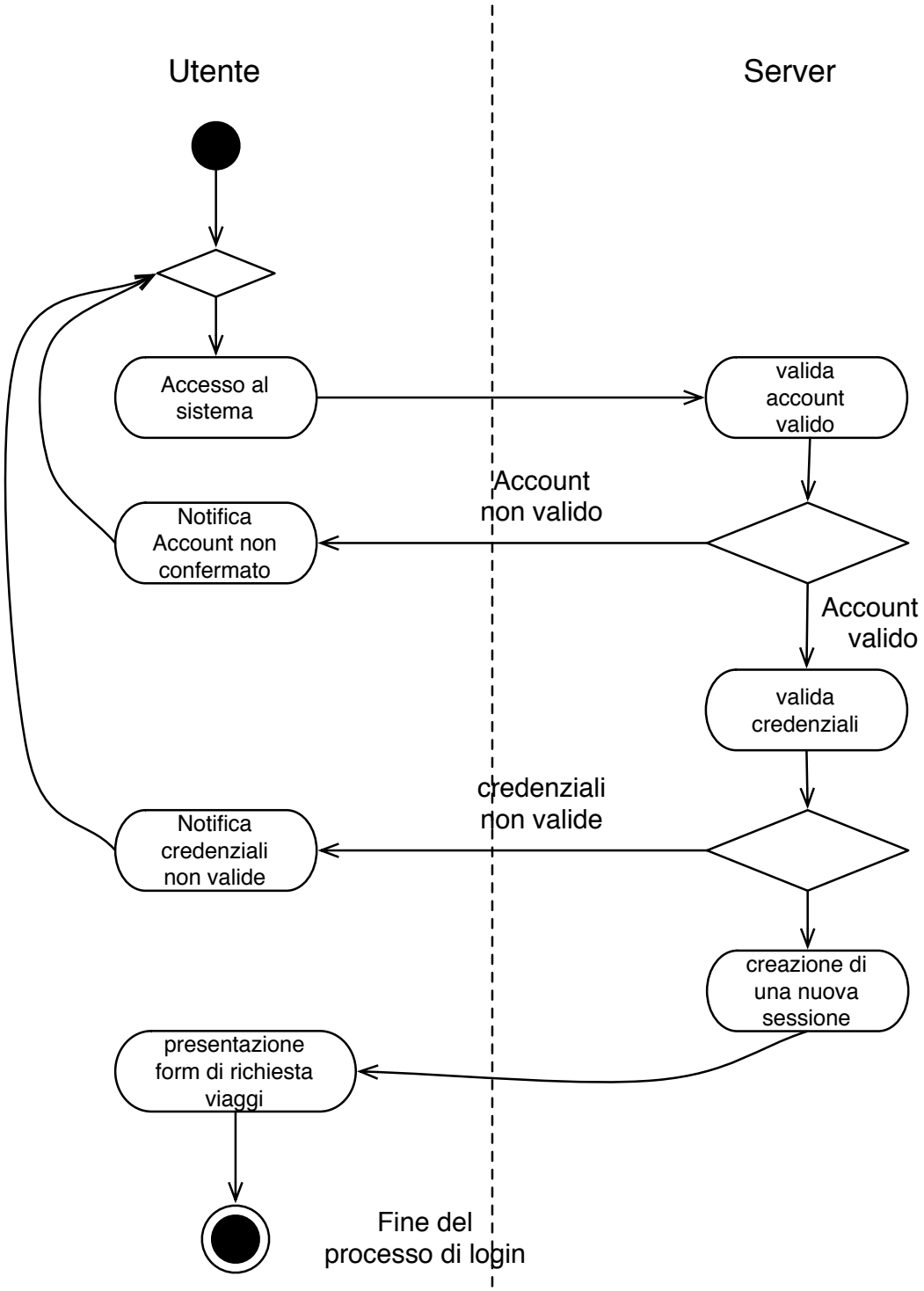


Figura 7.21: Diagramma delle Attività - Login

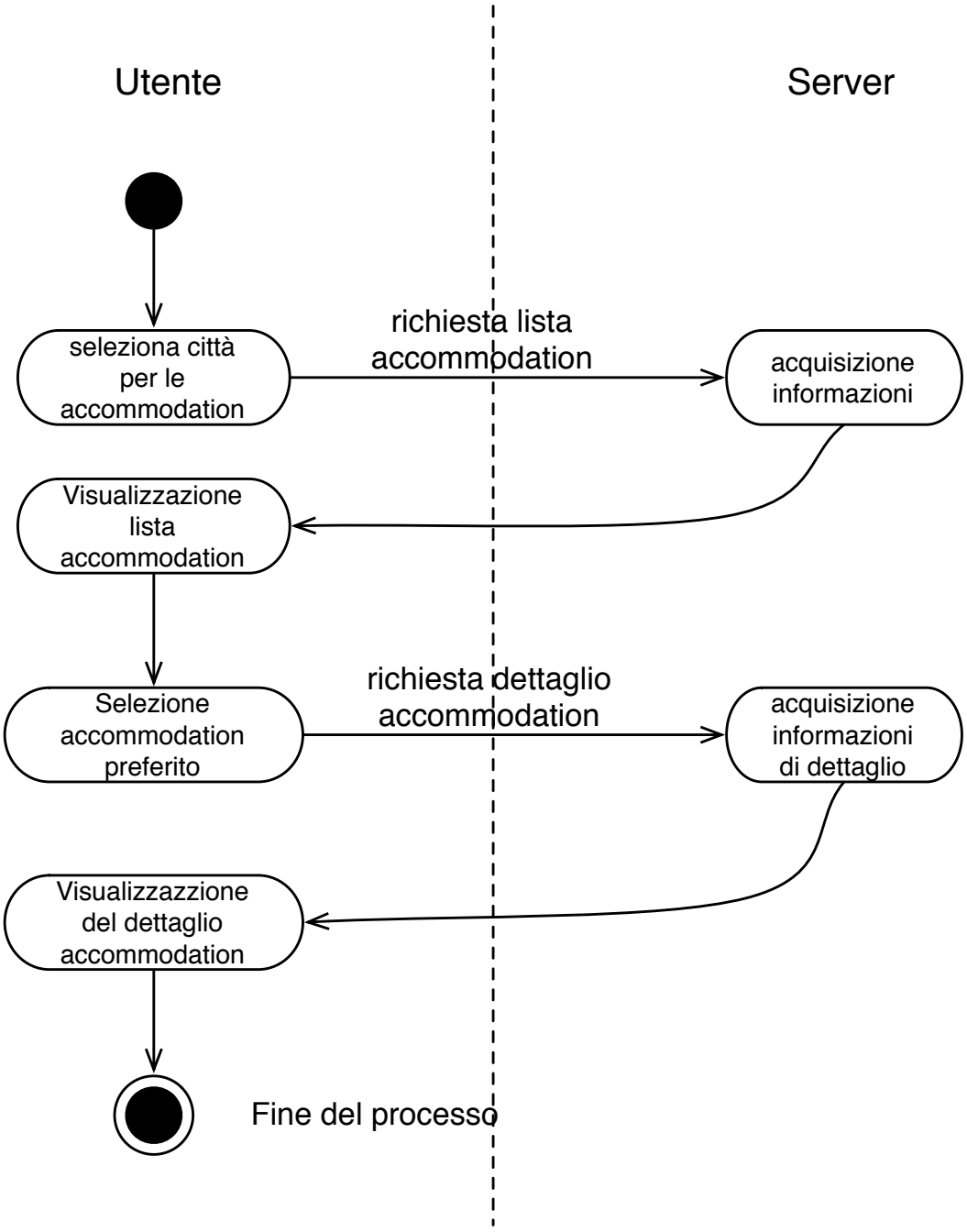


Figura 7.22: Diagramma delle Attività - Richiesta Accommodation

Capitolo 8

Conclusioni e sviluppi futuri

Il progetto CoolTraveling ci ha permesso di entrare in un contesto che attualmente rappresenta la nostra principale passione nonché la fonte del lavoro con il quale quotidianamente portiamo avanti la nostra società. Al di la della vittoria in se, che rappresenta una grossa soddisfazione, ci sono una serie di piccole soddisfazioni impercettibili concretamente ma che ci accompagnano giorno per giorno: il senso di appartenenza ad un team, il sapersi confrontare con persone che hanno background culturali e di formazione molto diversi tra loro, saper gestire nel piccolo situazioni di stress o in cui le tempistiche risultano ridottissime, ... La risonanza che abbiamo avuto grazie a questo progetto ci ha permesso di entrare in contatto con realtà della piccola imprenditorialità italiana e ci ha dato la possibilità di poter decidere il nostro futuro in un momento in cui avevamo bisogno di una guida.

Il mondo dei viaggi ci ha sempre appassionati e in questi ultimi tempi ha visto una seconda nascita nel panorama mobile. La competizione ovviamente è diventata spietata, in misura maggiore rispetto a due anni fa, quando ci siamo avvicinati a questo mondo che poi pian pian abbiamo abbandonato, forse per mancanza di certezze dal punto di vista imprenditoriale del progetto che stavamo creando.
L'esperienza che abbiamo maturato dialogando con i player con i quali abbiamo lavorato ci ha convinti sempre più che il panorama italiano non facilita l'introduzione di nuove realtà in ecosistemi già consolidati che sono appannaggio di pochi player. La conferma è proprio l'approccio che siamo stati costretti a mettere in atto per riuscire nella realizzazione del nostro prototipo. Tutti i contenuti dell'applicazione non provengono da accordi diretti con le relative società che ne posseggono la proprietà, ma sono stati presi dal web simulando i comportamenti degli utenti: una pratica che in gergo

tecnico viene chiama wrapping.

8.1 Sviluppi futuri

Abbiamo in mente numerosi sviluppi futuri per CoolTraveling e adesso ne vedremo con maggior dettaglio alcuni.

- Perfezionamento dell'acquisto sulla pagina della rispettiva compagnia di viaggio
- Estendere il panorama di soluzioni dei treni al mercato europeo, così da introdurre anche in altre nazioni le logiche di comparazione delle offerte dei viaggi.
- Ottimizzazione dell'interfaccia mediante l'utilizzo di HTML5 in modo da creare un prodotto sempre più user friendly ed accattivante possibile per gli utenti.
- Avvalerci di un team in grado di supportarci nella stesura di un business model che renda sostenibile un'idea imprenditoria con CoolTraveling alla base.
- Stringere accordi con le principali compagnie di viaggio per poter usufruire della loro base dati senza dover ricorrere al wrapping delle pagine.
- Estendere la soluzione di ricerca ad ulteriori mezzi di trasporto, quali i taxi, autobus, mezzi pubblici, navi ecc ...
- Poter effettuare ricerche complesse utilizzando combinazioni di diversi mezzi di trasporto per raggiungere la propria meta.
- Aggiungere dinamiche di social engagement tra gli utenti di CoolTraveling
- Costruire interfacce di pagamento in grado di rendere CoolTraveling un punto di riferimento sia per la ricerca che per l'acquisto di soluzioni di viaggio.
- Realizzare API pubbliche per l'interrogazione del nostro servizio, in modo che possa essere integrato nella maggior parte dei portali web e mobile legati all'organizzazione dei viaggi.

Bibliografia

[1] Lab42. *Lab42 WebSite* http://lab42.com

[2] Project Management Institute *A Guide to the Project Management Body of Knowledge*

[3] Eliyahu M. Goldratt. *Critical Chain*

[4] Agile Alliance *http://www.agilealliance.org*

[5] *WSDL*, Web Services Description Language. http://www.w3.org/TR/wsdl

[6] *Rest*, Representational state transfer. http://en.wikipedia.org/wiki/Representational_state_transferl

[7] Mamdouh Ibrahim, Gil Long *Service-Oriented Architecture and Enterprise Architecture*. IBM

[8] Jeff Davis, *Open Source SOA*

[9] *iTravel* brevetto Apple. http://www.patentlyapple.com/patently-apple/2010/04/itravel-apples-future-travel-centric-app-for-the-iphone.html

[10] W3C. *Extensible Markup Language (XML)* http://www.w3.org/XML/

[11] Expedia. *Expedia WebSite* http://www.expedia.it

[12] AccuWeather. *AccuWeather WebSite* http://www.accuweather.com

[13] ANTLR. *ANTLR WebSite* http://www.antlr.org

[14] Terence Parr. *The Definive ANTLR Reference, Building Domain-Specific Languages* http://www.pragprog.com/titles/tpantlr

[15] Wikipedia.org *EBNF (Extended Backus-Naur form)* `http://it.wikipedia.org/wiki/EBNF`

[16] *Popping holes in a few mobile myths* `http://venturebeat.com/2011/04/05/popping-holes-in-a-few-mobile-myths/`

Elenco delle figure