Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE
Corso di Laurea in Informatica Magistrale

TrainSpotter: una applicazione mobile collaborativa per il monitoraggio e l'analisi dei viaggi ferroviari

Tesi di Laurea in Sistemi Mobili

Relatore:
Chiar.mo Prof.
Vittorio Ghini

Presentata da:
Carlo Donzelli

II Sessione
Anno Accademico 2012-2013

Introduzione

Negli ultimi anni l'utilizzo quotidiano di dispositivi mobili di ultima generazione, più semplicemente chiamati smartphone, ha registrato un grande aumento che non accenna ad arrestarsi, anche grazie all'enorme varietà di modelli differenziati per caratteristiche e prezzo. Di pari passo all'evoluzione delle caratteristiche hardware, anche i sistemi operativi dei dispositivi mobili sono divenuti sempre più avanzati, abbinando le funzionalità di telefono cellulare a quelle di gestione di dati personali, produttività e svago, grazie alla presenza di applicazioni specifiche. Analizzando la situazione tecnologica attuale possiamo notare che la maggior parte del mercato mobile è occupato da sistemi tecnicamente avanzati che, oltre ad un set di applicazioni pre-installate, offrono pressoché in tutti i casi la possibilità di svilupparne di proprie. Inoltre, analizzando il traffico internet proveniente da rete mobile, notiamo un aumento del 70% durante il 2012, con un incremento stimato del 1300% nei 5 anni successivi[1]; segno evidente che il traffico di rete sempre più spesso passa per i dispositivi mobili.

Una crescita così marcata della mobilità ha influenzato lo sviluppo di un nuovo modo di fruizione dei contenuti, caratterizzato da una maggiore flessibilità riguardo a luogo, orario e strumenti a disposizione; flessibilità resa possibile anche da una sempre più elevata capacità computazionale, dalle moderne tecnologie di rete, e dalla recente introduzione del cloud computing. Si è fatto largo anche un nuovo modello di comunicazione efficace, basato sulla collaborazione e condivisione di informazioni da parte degli utilizzatori del sistema. La semplice collaborazione prevede -di per sè- uno scambio

di informazioni per il supporto e l'aiuto reciproco, ma quando più persone concorrono insieme al conseguimento di un obiettivo comune, otteniamo una forma più avanzata di collaborazione: la cooperazione.

Con la diffusione di smartphone e cloud computing è nato il problema di come connettere queste due tecnologie e di come rendere questa connessione quanto più trasparente possibile allo sviluppatore e all'utente finale. Il procedimento tipico sarebbe quello di creare, configurare ed utilizzare sistemi di backend in grado di immagazzinare dati ed eseguire business logic. Tuttavia, le tre classiche tipologie pre-esistenti di servizi cloud, Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS), non rispecchiano pienamente tutti i requisiti richiesti dalle mobile app; ecco perché è stata introdotta una nuova categoria chiamata Backend as a Service (BaaS). Il BaaS permette agli sviluppatori di impostare e gestire un backend per le loro applicazioni mobile in maniera semplice e veloce, potendo utilizzare il tempo risparmiato in questa fase per focalizzarsi sull'implementazione dell'applicazione vera e propria.

Questo progetto di tesi ha come obiettivo la creazione di una applicazione mobile collaborativa per il monitoraggio e l'analisi dei viaggi ferroviari. Collaborativa perché per poter adempiere alla sua funzione è necessario che i suoi utilizzatori partecipino attivamente all'accrescimento del database di dati, supponendo che con più dati raccolti sia maggiore l'accuratezza di quest'ultimi. Sarà un'applicazione di monitoraggio nel senso che di ogni singolo treno verranno presi in esame una serie di aspetti considerati utili ai fini della valutazione della qualità del servizio offerto. Analisi perché, una volta raccolti, questi dati possono essere utilizzati sia da coloro che viaggiano in treno -per decidere se prendere un treno piuttosto che un altro- sia da coloro che gestiscono la rete in modo da stilare delle statistiche o per eseguire degli interventi mirati su di una specifica tratta o anche per potenziare il servizio clienti a bordo del treno.

Il documento di tesi è suddiviso in 6 capitoli. Nel primo capitolo effettuerò una panoramica per illustrare lo scenario di riferimento su cui andrò

ad operare ovvero il trasporto ferroviario, e poi una introduzione sulle più rilevanti piattaforme smartphone presenti sul mercato oggi e sulle tecnologie relative al cloud computing. Nel secondo capitolo presenterò e descriverò gli strumenti a disposizione per lo sviluppo e a supporto di esso, mentre nel terzo capitolo descriverò la fase di progettazione, le scelte effettuate e la specifica dei requisiti. Nel quarto capitolo analizzerò tutte le funzionalità implementate dall'applicazione comprensive di quelle in via sviluppo. Nel quinto capitolo definirò i criteri su cui basarsi per poter valutare la bontà dell'applicazione e come misurare l'aderenza agli stessi. Infine nell'ultimo capitolo esporrò le conclusioni e gli sviluppi futuri.

Indice

Elenco delle figure

Capitolo 1

Scenario

In questo capitolo effettuerò una breve panoramica prima sullo scenario su cui si è deciso di operare per questo lavoro di tesi, ovvero la mobilità su treno, per poi passare ad una analisi delle tecnologie disponibili legate all'ambiente mobile.

1.1 Mobilità sui treni

Al giorno d'oggi gli spostamenti tramite treno costituiscono una parte rilevante della nostra mobilità, soprattutto per chi, per esigenze di lavoro o di studio, si trova a doversi spostare quotidianamente dalla città in cui risiede ad un'altra. Osserviamo che la rete ferroviaria italiana (RFI) si sviluppa abbastanza uniformemente su tutto il territorio e costituisce una fitta rete di collegamento tra piccoli e grandi centri urbani[2]. Ogni anno vengono stimati milioni di viaggiatori e milioni di tonnellate di merci che transitano su rotaia lungo tutto il paese.

Le linee ferroviarie che compongono la rete possono essere suddivise principalmente in tre categorie in base alle loro caratteristiche in:

- **linee fondamentali**: sono quelle soggette ad un'alta densità di traffico e presentano una elevata qualità dell'infrastruttura; ne fanno parte i collegamenti tra le più importanti città italiane;

- **linee complementari**: sono quelle che presentano livelli minori per quanto riguarda la densità di traffico, costituiscono le reti di collegamento regionali e connettono fittamente tra di loro le direttrici principali;

- **linee di nodo**: sono quelle che si sviluppano all'interno di grandi zone di scambio e collegamento tra linee fondamentali e complementari, situate in ambito metropolitano.

Le linee ferroviarie contano 16.742 km di lunghezza totale, sono suddivise in 6.444 km di linee fondamentali, 9359 km di linee complementari e 939 km di linee di nodo; 7.536 km sono a binario doppio mentre 9.206 km a binario semplice; infine, 11.959 km sono elettrificate, 4.783 km sono a diesel. Il numero totale di stazioni ferroviarie è 2.260.

Figura 1.1: Grafico della Rete Ferroviaria Italiana (RFI).

1.2 Qualità del servizio ferroviario

Su tutto il territorio italiano e con particolare riferimento alla regione Emilia Romagna[3], possiamo notare che sono già previste alcune attività di monitoraggio sulla qualità erogata sui treni e nelle stazioni svolte da personale specializzato, attività costantemente integrate dal rilevamento dello stato della rete e degli impianti ferroviari.

Queste attività comprendono le seguenti operazioni:

- verifica di alcuni parametri di qualità basilari come puntualità, affidabilità e affollamento;
- analisi della circolazione dei treni;
- verifica e analisi delle segnalazioni degli utenti;
- indagini sul campo per la verifica della qualità erogata sul treno e nelle stazioni ferroviarie.

Inoltre in passato è stato costituito il Comitato consultivo regionale degli utenti ferroviari dell'Emilia-Romagna (Crufer) che mira ad avvicinare gli utenti al fornitore di servizio. Tra le funzioni principali di questo organo troviamo:

- espressione di pareri sulle tematiche proposte dalla regione Emilia-Romagna;
- assunzione di iniziative di proposta alla regione su aspetti del servizio ferroviario;
- acquisizione di informazioni;
- esecuzione di studi, analisi e ricerche.

1.3 Esigenza di verificare la qualità di viaggio

Come già detto in precedenza sono migliaia le persone che ogni giorno scelgono il treno come mezzo di trasporto, sia perché in un certo senso non hanno alternativa (ad esempio non posseggono un'automobile) sia perché può risultare vantaggioso rispetto ad altri mezzi di trasporto. Ad esempio per gli spostamenti su medie distanze viaggiare in treno costa meno che viaggiare in aereo, e i prezzi del biglietto sono sostanzialmente stabili nel tempo. Sempre rispetto ai voli, non è necessario fare check-in, imbarco del bagaglio e non c'è bisogno di arrivare con largo anticipo in stazione. Non ci sono limitazioni per quanto riguarda il peso, la dimensione o il numero di bagagli, se non la buona educazione. Se si perde un treno per un ritardo o una mancata coincidenza, si può ripiegare su uno dei treni che transiteranno successivamente. Le stazioni sono quasi tutte collocate nel centro città quindi facilmente raggiungibili. Inoltre sono un mezzo di trasporto molto più sensibile e rispettoso dell'ambiente rispetto ad automobile o aereo. Rispetto all'auto invece, a discapito di una minore flessibilità degli orari degli spostamenti, si ha però il vantaggio di evitare il traffico e di non avere il problema del posteggio.

Normalmente la verifica delle condizioni di qualità del viaggio viene fatta da personale specializzato che certifica in prima persona se vengono rispettate una serie di regole e normative, ma come si può immaginare questo è un processo molto lento e macchinoso. Il personale addetto non è sufficiente a coprire la totalità delle linee ferroviarie, e anche nel caso di una segnalazione, l'iter per far si che essa vada a buona fine richiede del tempo. Nonostante gli sforzi delle società che gestiscono questo mezzo di trasporto, troppo spesso il viaggio risulta insoddisfacente, in modo particolare nei viaggi a bordo dei treni di categoria più bassa. Ritardi, sovraffollamento, scarsa pulizia, carrozze con mancanze funzionali sono alcuni dei problemi più diffusi.

1.4 Mobile computing

La costante evoluzione dei dispositivi mobili sta rendendo sempre più popolare l'uso di servizi di varia applicazione al di fuori della propria abitazione o del proprio posto di lavoro, eliminando di fatto i vincoli sulla posizione fisica dell'utente. Il mobile computing può essere definito come una modalità di interazione persona-computer dove il computer non è legato ad una posizione fisica. Fanno parte di questa categoria dispositivi come dispositivi indossabili, smartphone, tablet, computer ultra-mobile.

Oltre all'eliminazione di questi vincoli, il mobile computing introduce nuovi concetti da tenere in considerazione in fase di progettazione di un prodotto:

- localizzazione: possibilità di determinare in maniera automatica la posizione geografica di un dispositivo;
- navigazione: possibilità di creare un percorso tra due località;
- tracciamento: controllo e memorizzazione degli spostamenti del dispositivo;
- timing: determinazione del momento temporale del verificarsi di un evento in una certa località.

Tuttavia, fanno la loro comparsa anche alcune problematiche, tra cui:

- connettività con copertura e banda limitata;
- problematiche di sicurezza poiché molti utenti si collegano alla stessa rete;
- tenere conto del consumo energetico delle batterie, perché la rete energetica non è utilizzabile;
- la trasmissione è soggetta ad interferenze dovute alle condizioni meteo, ostacoli o alla conformazione del territorio;
- dimensioni schermo limitate, che complicano l'interazione.

1.5 Smartphone

Uno smartphone è una tipologia di dispositivo mobile che possiede funzioni avanzate di computazione e connettività, che si avvale di un sistema operativo completo con complessità paragonabile a quelli destinati a sistemi desktop. Una delle caratteristiche più interessanti è che essi (come i tablet) permettono l'installazione di applicazioni di terze parti che aggiungono nuove funzionalità al dispositivo.

Tra i produttori che detengono le maggiori quote di mercato troviamo Google con Android, Apple con iOS, RIM con Blackberry OS e Microsoft con Windows Phone. La crescente domanda di dispositivi mobili ne ha accelerato notevolmente lo sviluppo dal punto di vista hardware, dotandoli di processori evoluti, capacità di memoria sempre maggiore, schermi multitouch ad alta risoluzione, rendendo così possible anche lo sviluppo di applicazioni evolute. Inoltre, con la sempre maggiore crescita economica di questo settore, si è scatenata una forte competizione tra i maggiori produttori, alla ricerca della fetta di mercato più ampia.

Vediamo ora una breve descrizione per ogni sistema operativo mobile.

1.5.1 Android

Android è un sistema operativo basato su Linux destinato a dispositivi mobili touchscreen come smartphone e tablet, sviluppato inizialmente da Android, Inc. Acquisito da Google nel 2005, venne presentato la prima volta nel 2007. Il codice del sistema operativo è open source rilasciato sotto licenza Apache[1] permettendo ad esempio ai produttori, ai carriers o ad appassionati di personalizzare a proprio piacimento il sistema. E' la prima piattaforma software al mondo per diffusione, anche grazie alla sua grande flessibilità, al basso costo e funzionalità implementate. Oltre a smartphone e tablet, lo troviamo

[1]Licenza di software libero non copyleft che obbliga gli utenti a preservare l'informativa di diritto d'autore e d'esclusione di responsabilità nelle versioni modificate.

installato anche su console (M.O.J.O.[2] e OUYA[3]), media player, fotocamere digitali e altri dispositivi elettronici. Nel maggio 2013 le applicazioni scaricate dal Google Play Store erano circa 48 miliardi, con circa 1 miliardo di dispositivi Android attivati.

Figura 1.2: Alcuni dispositivi Android

Da un punto di vista tecnico, Android è costituito da un Kernel basato sul kernel Linux, con middleware, librerie e API scritte in C (o C++) e software in esecuzione su un framework di applicazioni che include librerie Java. La piattaforma hardware principale di Android è l'architettura ARM. Le applicazioni Android sono basate su Java.

1.5.2 iOS

E' un sistema operativo proprietario mobile basato su UNIX e sviluppato da Apple per i suoi dispositivi mobili. Fin dalle sue prime incarnazioni una caratteristica funzionale è stata il fondarsi dell'interfaccia utente sul concetto di manipolazione diretta e multitouch. L'interazione col sistema comprende

[2]M.O.J.O.: http://madcatz.com/pressroom/m-o-j-o-android-micro-console/
[3]OUYA: http://www.ouya.tv/

le cosiddette “gestures”, ovvero movimenti da compiere con le dita come swipe, tap, pinch. Il sistema è composto da quattro livelli di astrazione: Core OS, Core Services, Media, Cocoa Touch.

Figura 1.3: Alcuni dispositivi iOS

Ad agosto 2013 le applicazioni disponibili nello store ufficiale erano circa 900.000, scaricate più di 50 miliardi di volte.

1.5.3 BlackBerry OS

BlackBerry OS è un sistema operativo mobile proprietario sviluppato da BlackBerry Ltd. che equipaggia gli smartphone dello stesso brand. La caratteristica principale che ha contribuito al successo di questa piattaforma è la gestione delle mail attraverso Mobile Information Device Profile (MIDP) grazie alla quale le mail vengono consegnate sul dispositivo da appositi server attraverso un servizio di push, assieme alla sincronizzazione di contatti, calendari, attività e note con Microsoft Exchange. Anche in questo caso gli sviluppatori, tramite le API disponibili, possono scrivere software di terze parti.

Figura 1.4: Alcuni dispositivi BlackBerry

A gennaio 2013 è stato presentato BlackBerry 10, nuova versione del sistema operativo compatibile con smartphone e tablet di ultima generazione, basato sul sistema operativo progettato come sistema microkernel[4] QNX. Con questo sistema si compie la transizione verso dispositivi completamente touchscreen, seguendo il trend del momento e abbandonando la classica tastiera fisica e trackball caratteristici dei vecchi device.

1.5.4 Windows Phone

Anche Windows Phone è un sistema operativo mobile proprietario sviluppato da Microsoft, visto come discendente naturale di Windows Mobile, che si pone però come obiettivo il mercato consumer piuttosto che quello professionale. Una caratteristica distintiva è l'interfaccia utente offerta, nome in codice Metro, composta da riquadri animati che possono costituire dei link per l'apertura di determinate applicazioni o funzionalità.

[4]L'idea base è quella di costruire il sistema operativo come un insieme di piccole parti che offrono uno specifico servizio

Figura 1.5: Alcuni dispositivi Windows Phone

Anche questo caso è possibile sviluppare applicazioni di terze parti tramite il linguaggio C/C++.

1.6 Cloud computing

Una tecnologia che sta prendendo molto piede negli ultimi anni e su cui sempre più utenti stanno facendo affidamento è quella del cloud computing, che molto sinteticamente può essere descritto come un insieme di tecnologie che consentono l'elaborazione di dati e operazioni e facoltativamente in seguito di memorizzarli, tramite l'utilizzo di risorse sia hardware che software distribuite su di una rete con tipica architettura client-server. Si tratta sostanzialmente di una modalità di fruizione di servizi come memorizzazione ed elaborazione di dati tramite risorse hardware e software delocalizzate.

Tra le caratteristiche principali, elenchiamo[4]:

- **servizio self-service on-demand**: l'utente può disporre autonomamente sulla base delle proprie esigenze di capacità di calcolo e storage, senza quindi richiedere il supporto "umano" del fornitore del servizio;

- **accesso tramite rete**: è possibile accedere ai servizi cloud da qualsiasi dispositivo (fisso o mobile) dotato di connessione, attraverso meccanismi di accesso standard;

- **condivisione delle risorse**: le risorse di calcolo del provider sono messe in comune per poter servire più utenti usando un modello condiviso, dove differenti risorse sia fisiche che virtuali sono dinamicamente allocate sulla base dei bisogni del singolo cliente. L'utente nella maggior parte dei casi non ha controllo o non sa l'esatta ubicazione delle risorse fornite. Esempi di risorse includono memoria, elaborazione e larghezza di banda della rete;

- **elasticità**: le risorse sono facilmente acquisibili e rilasciabili per scalare sia verso l'alto che verso il basso sulla base della domanda. Lato utente le risorse disponibili appaiono illimitate e possono essere allocate in qualsiasi momento e quantità;

- **servizi su misura**: tutte le risorse sono controllate e monitorate sia dal provider che dall'utente, in modo trasparente. Il sistema cloud controlla e ottimizza automaticamente l'utilizzo delle risorse sulla base del tipo di servizio utilizzato (storage, banda, computazione).

Tutto ciò porta ad una serie di vantaggi per l'utilizzatore finale del servizio cloud. Innanzitutto vi è un risparmio dal punto di vista economico sia per quanto riguarda l'hardware poiché non sarà più necessario possedere computer di fascia alta per eseguire le applicazioni cloud, sia per il software poiché si pagherà solo per il reale utilizzo del servizio. Inoltre, si ottiene una maggiore flessibilità nel caso in cui sia necessario una aumento delle risorse utilizzabili e maggiore scalabilità poiché è possibile allocare nuove risorse per fare fronte a nuove richieste. I dati e le applicazioni in questo contesto sono accessibili in mobilità, consultabili online in un qualsiasi luogo in cui vi sia una connessione internet e fruibili da una moltitudine di dispositivi (smartphone, tablet, laptop).

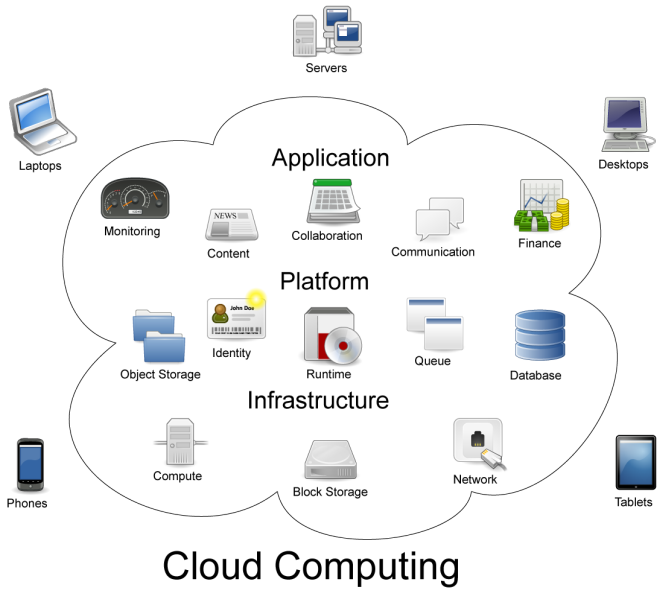


Figura 1.6: Panoramica servizi Cloud Computing

Dall'altro lato però bisogna anche tenere in considerazione i rischi e le problematiche a cui vengono esposti gli utenti[5]. Tra le più rilevanti abbiamo:

- **sicurezza e privacy**: i dati personali o sensibili vengono spesso memorizzati in strutture server che hanno sede in uno stato diverso da quello dell'utilizzatore, con potenziali rischi di violazione della privacy, come utilizzo degli stessi per operazioni di marketing o profilazione degli utenti[6]. Inoltre di frequente la richiesta di cancellazione dei dati non coincide con l'eliminazione effettiva;

- **continuità del servizio**: un eventuale malfunzionamento del sistema potrebbe compromettere l'operabilità di un gran numero di utenti (problema risolto in parte dalla rindondanza dei componenti critici del sistema cloud). Inoltre, visto che tutte le operazioni vengono svolte

tramite l'accesso ad internet, anche i problemi di connessione possono influire sull'operabilità;

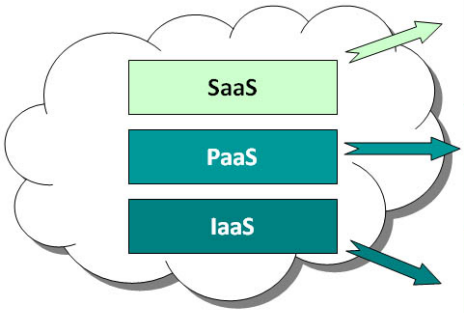
- **vendor lock-in**: poiché questo modello è relativamente nuovo, gli standard devono ancora essere pienamente definiti, col risultato che molti provider usano piattaforme e servizi proprietari. Perciò la migrazione da un provider all'altro è un'operazione spesso complessa e costosa da effettuare.

I fornitori di servizi di cloud computing rendono disponibili i loro prodotti secondo alcuni modelli basilari[7]. Possiamo distinguere tre varianti fondamentali:

- **Infrastructure as a Service (IaaS)**: rappresenta il livello di servizi basati sul cloud di più basso livello, in cui i provider forniscono elaboratori (fisici o virtualizzati), server e reti di interconnessione. Modello rivolto prevalentemente ad architetti di infrastrutture e sistemisti;

- **Platform as a Service (PaaS)**: rappresenta una piattaforma di computing che tipicamente include il sistema operativo, un ambiente di esecuzione, database e web server. Modello rivolto prevalentemente a sviluppatori di applicazioni tradizionali; mentre per gli sviluppatori di applicazioni web e mobile è consigliabile un **Backend as a Service (BaaS)**: modello che fornisce un backend basato su cloud storage in congiunzione con altri servizi come gestione utente, notifiche push, integrazione con i social network[8];

- **Software as a Service (SaaS)**: modello di distribuzione software, dove il software ed i dati ad esso associati sono ospitati nel cloud. Modello rivolto prevalentemente all'utente finale che tipicamente vi accede tramite web browser.



Who Uses It	What Services are available	Why use it?
Business Users	EMail, Office Automation, CRM, Website Testing, Wiki, Blog, Virtual Desktop ...	To complete business tasks
Developers and Deployers	Service and application test, development, integration and deployment	Create or deploy applications and services for users
System Managers	Virtual machines, operating systems, message queues, networks, storage, CPU, memory, backup services	Create platforms for service and application test, development, integration and deployment

Figura 1.7: Vista a layer del Cloud Computing

Analizziamo ora i singoli modelli di cloud computing, per capire quale soddisfa meglio i requisiti richiesti dal progetto oggetto di questa tesi.

1.6.1 Infrastructure as a Service (IaaS)

Modello che rappresenta l'accesso al livello più basso del cloud computing. Può essere rappresentato da una infrastruttura hardware remota che offre diverse risorse come ad esempio cicli di cpu, memoria, storage per permettere l'esecuzione di virtual machines tramite assegnazione di risorse on demand[9]. L'utilizzatore può sfruttare risorse di elaborazione, memoria, rete, sistemi operativi e applicazioni ma non gestisce l'infrastruttura sottostante. Gli utenti cloud possono virtualizzare immagini di sistemi operativi e il software per la creazione di applicazioni sull'infrastruttura cloud.

Nel modell IaaS è l'utente che si occupa del mantenimento del sistema ed il costo richiesto riflette l'utilizzo delle risorse effettivamente allocate e consumate[10].

1.6.2 Platform as a Service (PaaS)

Questo modello comprende i servizi che mettono a disposizione una piattaforma pronta all'utilizzo per il deployment di applicazioni o elaborazione di dati. L'utilizzatore può distribuire sull'infrastruttura cloud le applicazioni che ha creato o acquisito da soggetti terzi, utilizzando linguaggi di programmazione, librerie e servizi: esso non gestisce l'infrastruttura sottostante e nemmeno sistemi operativi e memoria ma ha il controllo sulle applicazioni e sulla configurazione dell'ambiente che le ospita[11].

Come nel caso dei servizi di web hosting, l'utente finale non ha la necessità di preoccuparsi della gestione software o hardware, ma ha disposizione delle librerie e framework per essere subito operativo sul deployment della logica di business e del layer di presentazione. Permettono di sviluppare, testare e implementare applicazioni senza i costi e le complessità legate all'acquisto e configurazione e ottimizzazione dell'hardware e del software di base.

Tra le caratteristiche più frequenti troviamo strumenti per la creazione di interfaccia utente basata su web, utilizzo di applicazioni da parte di più utenti simultaneamente, integrazione con servizi web e database (SOAP e REST). In questo modello l'utente crea il software usando strumenti e librerie messe a disposizione dal fornitore, ne segue lo sviluppo e provvede alla sua configurazione. Dall'altra parte il provider fornisce rete, server, storage e altri servizi. il vantaggio che si ottiene durante lo sviluppo sta nel non dover sopportare i costi e la complessità dell'acquisto e manutenzione dell'hardware e software sottostante[12].

I servizi offrono vari livelli di scalabilità e mantenimento. Le funzionalità rese disponibili dai PaaS possono includere anche strumenti per il design dell'applicazione, per lo sviluppo, testing e distribuzione così come servizi di collaborazione in team, integrazione di web service, di database, sicurezza, persistenza, application versioning.

1.6.3 Backend as a Service (BaaS)

Con il boom della diffusione degli smartphone e delle applicazioni mobile sta emergendo un trend che gli addetti ai lavori hanno formalmente chiamato Backend as a Service (BaaS), anche rinominato come mobile Backend as a Service (mBaaS): è un modello relativamente recente (un paio d'anni circa) che rappresenta una modalità di connessione tra mobile app e servizi cloud, eliminando il bisogno di costruirsi il proprio backend[13]. Può essere considerata una tecnologia relativamente recente nel settore del cloud computing poiché le aziende fornitrici di questo servizio sono datate a partire dal 2011 in poi, con un mercato globale destinato a crescere dai 216.5 milioni di dollari del 2012 fino a circa 7.7 miliardi di dollari nel 2017[14].

La nascita di questo modello è dovuta alla grande diffusione di applicazioni mobili, che richiedono tutte più o meno tutte le stesse caratteristiche per quanto riguarda il backend, ovvero notifiche push, integrazione con i social network e cloud storage. Però ognuna di queste funzionalità richiede l'incorporazione delle proprie API per poter essere sfruttata, operazione che può risultare complicata e richiedere molto tempo. E qui intervongono i BaaS, che fanno da ponte tra il frontend di una applicazione e il backend cloud-based, fornendo delle API e un SDK unificati. Tutto ciò risulta nel fatto che gli sviluppatori non devono più concentrarsi anche sulla parte di backend. Il tempo risparmiato quindi può essere reinvestito nella parte frontend come ad esempio nel design dell'interfaccia utente. La scalabilità, fattore fondamentale per lo sviluppatore di app mobile, è gestita intrinsecamente dai BaaS: le applicazioni "scalano" automaticamente, nel cloud, in base alla variazioni del volume di utenti e traffico generato.

Una problematica può essere quella del lock-in del fornitore del servizio, che si verifica quando un utente si trova ad utilizzare un prodotto o servizio che non può essere facilmente trasferito su una piattaforma simile fornita da un altro competitor, a causa ad esempio di incompatibilità tra le tecnologie (a volte proprietarie) utilizzate. Data la complessità nella migrazione del servizio web, è probabile che molti utenti decidano di rimanere affiliati ad un

provider che non li soddisfa appieno, solo per evitare il difficile processo di transizione. Per spostare dati da un provider all'altro spesso è necessario prima trasferire i dati in locale e poi ritrasferirli nel cloud. Inoltre può capitare che i dati siano stati modificati preventivamente per ragioni di compatibilità con il provider originario, così che si renda necessario riportarli allo stadio originale prima di muoverli ancora. Quindi per evitare questa situazione è bene, prima di decidere l'adozione di un determinato servizio, informarsi se sia possibile la migrazione, se ci sono degli strumenti di migrazione o servizi che facilitino lo spostamento di grandi quantità di dati, scegliere dei provider che supportino degli standard come il Cloud Data Management Interface (CDMI). Anche se spesso i fornitori assicurano flessibilità di sviluppo e di migrazione in qualsiasi momento, rimane comunque una operazione molto complicata.

Tra i servizi più comuni forniti abbiamo come già detto notifiche push, file storage e sharing, integrazione con i social network, ma anche servizi di geolocalizzazione, messaggistica, gestione dell'utente, business logic e strumenti di analisi dell'utilizzo. Le API e SDK forniti sono compatibili per lo sviluppo di applicazioni multi-piattaforma tra cui iOS, Android, Windows Phone, BlackBerry, HTML5.

Le finalità principali dei BaaS si possono riassumere nella facilitazione ed alleggerimento dello sforzo dello sviluppatore. Si guadagna in efficienza riducendo l'overhead nelle fasi dello sviluppo di applicazioni mobili, raggiungendo così più velocemente il mercato. I dati e la rete sono ottimizzati per l'integrazione con smartphone e tablet, forniscono una infrastruttura scalabile e sicura. Insomma, tutto lo sviluppo e il mantenimento dell'infrastruttura che di solito era associata allo sviluppo di applicazioni, ora viene astratto e incapsulato in una singola piattaforma di backend.

Attualmente ci sono più di 50 aziende che forniscono questo tipo di servizi, ognuna delle quali fornisce il proprio approccio alla fornitura del servizio anche se più in generale tentano tutte di emulare coloro che sono nate prima. Ovviamente era impossibile analizzare tutti questi BaaS uno ad uno per

capire quale meglio si adattava e soddisfava le mie esigenze per questo lavoro di tesi, quindi per prendere la mia decisione mi sono basato su opinioni di utilizzatori, recensioni e compatibilità con la piattaforma su cui ho scelto di sviluppare, ovvero iOS sviluppato dalla Apple.

Figura 1.8: Alcuni tra i principali fornitori di mBaaS

1.6.4 Software as a Service (SaaS)

Rappresentano i servizi, principalmente legati al mondo commerciale che mettono a disposizione dell'utente finale un software personabilizzabile per le proprie esigenze ma completamente funzionante, che va a soddisfare determinate richieste. L'utilizzatore può utilizzare le applicazioni messe a disposizione dal provider che girano sull'infrastruttura cloud e le applicazioni sono accessibili sia tramite browser oppure da applicazioni native. L'utilizzatore non controlla l'infrastruttura cloud, così come sistemi operativi o memoria, così come le capacità delle singole applicazioni, ma si riduce ad una limitata configurazione delle stesse[13].

Gli esempi più comuni sono le applicazioni di webmail, oppure le suite di produttività come Google Docs. E' un modello di distribuzione del software applicativo dove un produttore sviluppa, esegue e gestisce una applicazione web che mette poi a disposizione dei propri clienti tramite Internet. Il software e i dati ad esso associati sono hostati nel cloud. L'utente solitamente vi accede tramite un thin-client, ovvero una macchina con poca o nessuna logica, via browser. In questo caso, l'utilizzatore non paga per ottenere

il possesso del software, ma solo per utilizzarlo. L'utilizzo avviene tramite specifiche API accessibili via web, ad esempio come web service o REST.

La maggior parte delle soluzioni SaaS sono basate su di una architettura multi-localizzata, dove una singola istanza dell'applicativo è usata da più utenti. Per assicurarne la scalabilità, l'applicazione è installata su diverse macchine. Ciò si contrappone alla distribuzione software tradizionale dove ci sono molteplici copie fisiche del software che girano su diverse configurazioni. Esempi di questo tipo di applicativo possono essere software di gestione delle relazioni coi clienti (CRM), video conferenza, webmail, gestione delle risorse umane (HRM), gestione dei contenuti (CM). Quindi SaaS può essere interpretato come un insieme di mezzi, servizi e competenze che consente alle aziende di esternalizzare completamente alcuni aspetti del loro sistema informativo in cambio di un corrispettivo in denaro piuttosto che in un investimento vero e proprio.

Alcune della maggiori caratteristiche sono: un'alta personalizzazione e configurazione specifica per utilizzatore, update e bugfix rilasciati più frequentemente rispetto al software tradizionale, implementazione di protocolli quali HTTP, REST, SOAP, JSON, possibilità di implementazione di caratteristiche collaborative e di condivisione delle informazioni. Tra i vantaggi ve ne è certamente uno finanziario, in quanto le soluzioni SaaS sono flessibili e on-demand e perché è necessario solo una macchina entry-level con un browser, abbassando di fatto i costi di acquisto hardware e del consumo di elettricità.

Dall'altro lato c'è la questione della sicurezza e privacy di dati poiché risiedono nel server del fornitore del servizio, rendendo necessario di fatto l'implementazione di politiche riguardo il trattamento dei suddetti dati. Sono necessari dei piani aziendali alternativi nel caso sorgano delle divergenze con il provider; inoltre, visto che il lavoro in questo caso è svolto su due macchine differenti, si possono verificare notevoli difficoltà nel caso in cui cessi la comunicazone (problemi con l'Internet service provider). E' possibile inoltre che la comunicazione soffra di latenza perché applicativi hostati nel cloud

sono fisicamente lontani dall'utilizzatore.

Tra i più famosi fornitori di SaaS troviamo Amazon Web Service di Amazon, Google Apps di Google, iCloud di Apple, Microsoft, Oracle.

Capitolo 2

Strumenti

In questo capitolo descriverò gli strumenti hardware e software di cui mi sono servito per la realizzazione di questo lavoro di tesi.

2.1 Strumenti hardware

2.1.1 iPhone

Con il termine iPhone si vuole indicare una linea di smartphone progettata e distribuita interamente da Apple, che esegue il sistema operativo iOS anch'esso proprietario. La prima generazione di iPhone fu messa in commercio a partire da giugno 2007 mentre l'ultima, la settima, ha fatto la sua comparsa sul mercato il 20 Settembre 2013, meno di due mesi fa.

Una delle caratteristiche che ha contraddistinto l'iPhone fin dal primo modello è la sua interfaccia utente, in forte contrasto con gli standard che dominavano il mercato, costruita ed adattata allo schermo multi-touch che aveva in dotazione. Possiede una connessione Wi-Fi, Bluetooth e una connessione cellulare sulle bande 2G, 3G, 4G e LTE. E' in grado di scattare foto, fare riprese video, riprodurre musica, inviare e ricevere mail, navigare su internet, fare chiamate, inviare sms e altro ancora.

Lo schermo è di tipo capacitivo, a cristalli liquidi, resistente ai graffi e abilitato al multitouch. Tra i sensori sono presenti quello di prossimità, di

luce ambientale, accelerometro, giroscopio e magnetometro. Infine troviamo un GPS di tipo assistito.

Figura 2.1: Visuale anteriore, laterale e posteriore di un iPhone 4

2.1.2 Utente

Classificare l'utente come strumento fisico può sembrare riduttivo ed in effetti lo è, ma ai fini di questo documento di tesi può risultare tale. L'applicazione sviluppata, usando un approccio di tipo collaborativo, pone l'utente non solo come fruitore passivo di informazioni ma prevede anche la sua partecipazione attiva, poiché può eseguire inserimento di dati, una delle condizioni fondamentali per permetterne l'utilizzo. I dati generati dagli utenti hanno il vantaggio di essere aggiornati e raccolti a costo zero ma, data la natura umana, non sono attendibili al 100%.

2.2 Strumenti software

2.2.1 iOS

iOS (chiamato in precedenza iPhone OS) è il nome del sistema operativo per dispositivi mobile sviluppato e distribuito da Apple. Giunto alla sua settima incarnazione, è presente oltre che ovviamente su iPhone, anche su iPod Touch, iPad, iPad Mini e Apple TV. A differenza di altri sistemi operativi mobile come Android o Windows Phone, iOS non può essere installato su dispositivi che non siano Apple. Uno dei punti di forza di questo sistema operativo è, oltre alla grande facilità d'uso, la disponibilità di più di 900'000 applicazioni presenti sull'App Store.

L'interfaccia utente di iOS è basata sulla manipolazione diretta da parte dell'utente degli oggetti contenuti nel display multi-touch tramite le cosiddette "gestures", definibili come le azioni che l'utente può compiere con una o più dita in movimento sullo schermo. L'SDK di iOS fornisce tutte le risorse necessarie per lo sviluppo di applicazioni native e avere un'idea del funzionamento delle tecnologie e degli strumenti in esso contenuto è cruciale per fare poi le scelte giuste riguardo il design e l'implementazione della propria app.

Figura 2.2: Schermata iniziale di iOS 6 a sinistra e iOS 7 a destra

L'architettura di iOS è rappresentabile per livelli. Al livello più alto, iOS agisce da intermediario tra l'hardware e le applicazioni: infatti esse comunicano con l'hardware attraverso una serie di interfacce di sistema ben definite. Invece i livelli più bassi contengono i servizi e le tecnologie fondamentali. Possiamo distinguere 4 diversi livelli:

- **Core OS**: livello che contiene le caratteristiche di basso livello su cui fanno affidamento la maggior parte delle altre tecnologie integrate. Anche se queste tecnologie non sono utilizzate direttamente nell'app, molto probabilmente sono utilizzate da altri frameworks. Mentre per quanto riguarda le situazioni in cui c'è bisogno di interfacciarsi esplicitamente con questioni di sicurezza o comunicazione con accessori hardware esterni, è necessario usare i framework contenuti in questo layer;

- **Core Services**: livello che contiene servizi di sistema fondamentali per le applicazioni, come i frameworks Core Foundation e Foundation che definiscono le primitive utilizzate da tutte le app. Questo livello contiene anche delle tecnologie per il supporto di funzioni come ad esempio la localizzazione, iCloud, i social media e le funzionalità di rete;

- **Media**: livello che contiene la grafica, l'audio, e le tecnologie video impiegate per implementare funzionalità multimediali all'interno dell'applicazione;

- **Cocoa Touch**: livello che contiene i framework fondamentali per costruire applicazioni iOS. Questi frameworks definiscono l'aspetto dell'applicazione, forniscono una infrastruttura base e il supporto a funzionalità come il multitasking, l'input touch-based, le notifiche push e altri servizi di sistema di alto livello. Durante la fase di design dell'applicazione, è consigliabile approfondire le tecnologie di questo livello per vedere quali sono potenzialmente utilizzabili.

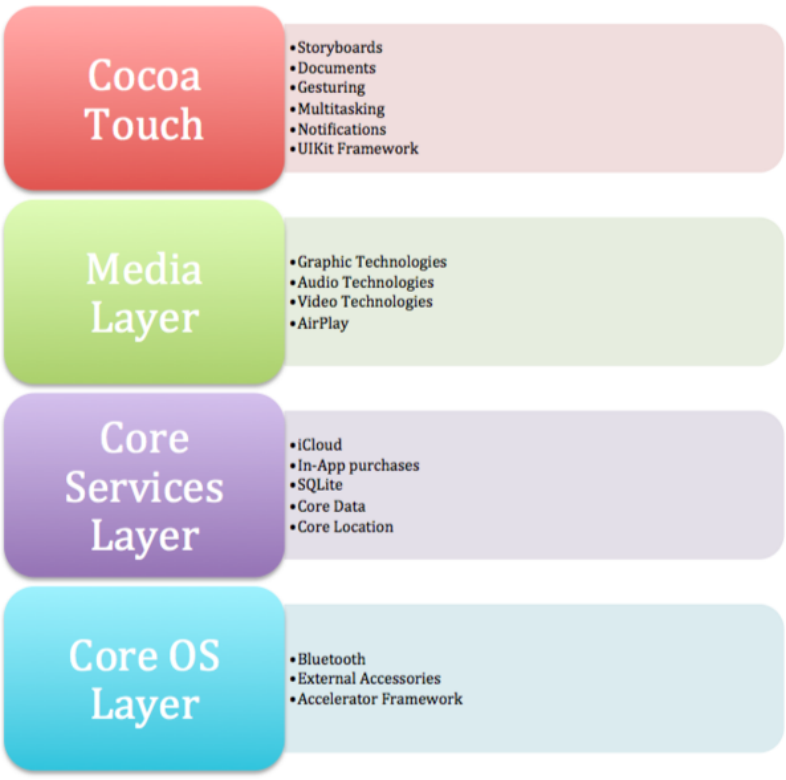


Figura 2.3: Livelli architetturali di iOS

2.2.2 Xcode

Xcode è un ambiente di sviluppo integrato (IDE) sviluppato e distribuito gratuitamente da Apple, per agevolare lo sviluppo di software sia per Mac OS X che per iOS. E' affiancato dall'Interface Builder, uno strumento grafico che facilita la strutturazione delle interfacce grafiche. Include il compilatore Gnu Compiler Collection (GCC) per la compilazione di codice C, C++, Objective-C e Java.

Una volta avviato l'applicativo e creato un progetto, ci si ritrova nella finestra principale del workspace che racchiude tutti gli strumenti e le opzioni principali che si andranno ad usare più frequentemente.

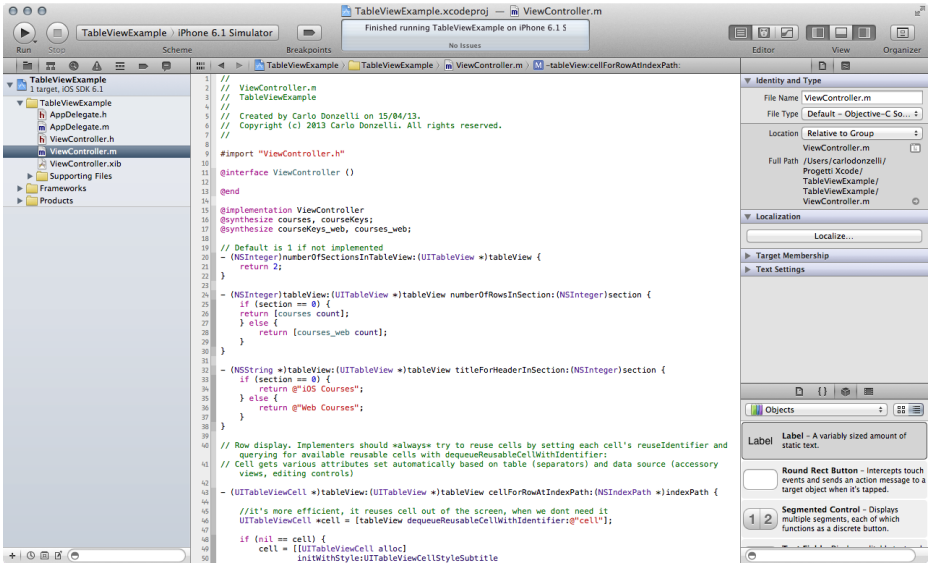


Figura 2.4: Finestra principale di Xcode

Nella parte superiore della schermata è presente la barra degli strumenti. A sinistra sono posizionati i comandi per eseguire ed arrestare l'esecuzione del progetto su cui si sta lavorando; proseguendo verso destra troviamo il selettore dello schema da eseguire, a seconda che si voglia eseguire l'applicativo sul simulatore o se sul dispositivo fisico; poco più in la l'opzione per attivare o disattivare i breakpoint, utilizzati in fase di debug. Al centro troviamo l'activity view, un piccolo pannello che comunica all'utente cioè che sta succedendo in quel preciso istante. Infine sulla parte destra sono presenti dei selettori: il primo permette di cambiare la visuale di editing del codice da standard alla modalità assistita; il secondo attiva o disattiva la visualizzazione di determinate aree dell'editor. Infine troviamo una opzione per visualizzare l'organizer.

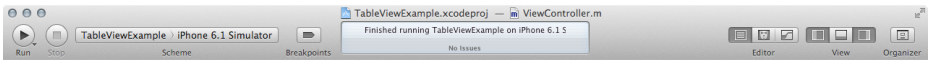


Figura 2.5: Barra degli strumenti di Xcode

Immediatamente sotto la barra degli strumenti, posizionata lungo la colonna sinistra di Xcode, è presente la vista di navigazione. Essa è composta internamente da sette diverse sezioni che forniscono una vista dettagliata ciascuna su una specifica parte di interesse. Nello specifico troviamo le opzioni: project, symbol, search, issues, debug, breakpoint, log.

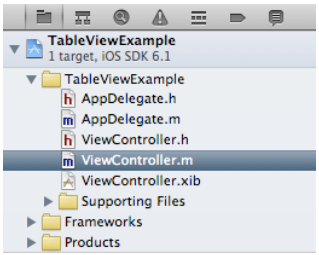


Figura 2.6: Vista navigator di Xcode

La jump bar è una sorta di menu strutturato nel classico stile breadcrumb, che permette quindi di "saltare" direttamente ad uno specifico elemento nella gerarchia di file con cui si sta lavorando.

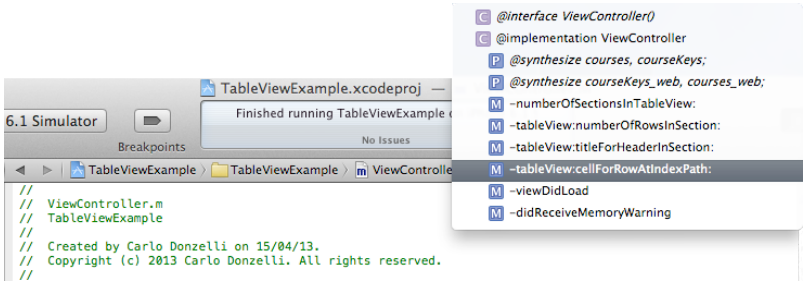


Figura 2.7: Jump bar di Xcode

L'utility panel risiede nella colonna destra dell'interfaccia e anche questo è suddiviso in 6 sotto-menù (chiamati inspector) che mostrano determinate opzioni a seconda dell'elemento su cui si sta correntemente lavorando: file, quick help, identity, attributes, size, connections.

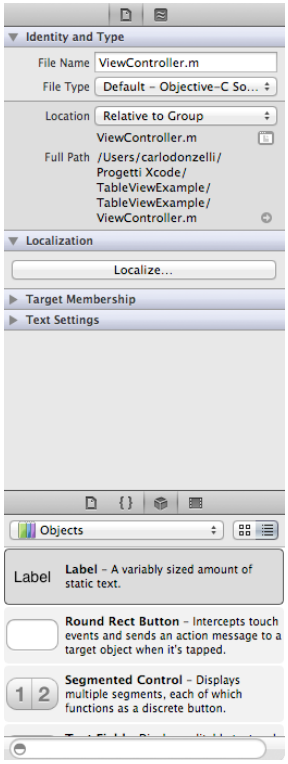


Figura 2.8: Utility panel di Xcode

Infine l'interface builder è la sezione di Xcode dove si procede alla progettazione e design dell'interfaccia utente. E' possibile trascinare direttamente sulla view gli elementi grafici di iOS come pulsanti o label e creare così l'interfaccia, per poi personalizzarla tramite l'utility panel senza scrivere una riga di codice.

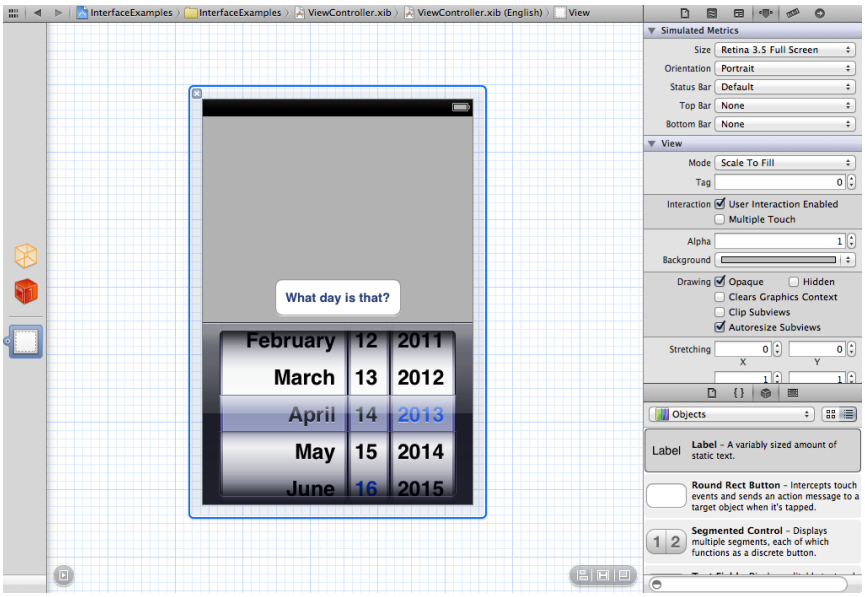


Figura 2.9: Interface Builder di Xcode

Il simulatore iOS è un potente strumento che permette rapidamente di eseguire e testare un'applicazione durante il suo processo di sviluppo. Permette di simulare una moltitudine di dispositivi iOS con diverse versioni del sistema operativo. Ogni versione del software simulata è considerata un ambiente separato dagli altri, indipendente, con i propri file e impostazioni. L'utilizzo del simulatore permette di trovare problematiche nella applicazione durante la fase di design o di testing iniziale e di prendere confidenza con l'ambiente di sviluppo iOS prima di diventare ufficialmente uno sviluppatore affiliato.

E' possibile riprodurre alcune funzionalità del dispositivo hardware come lo "shake", rotazione dello schermo, simulare la locazione GPS; d'altro canto non sono disponibili accelerometro, giroscopio, fotocamera e sensori di prossimità.

Figura 2.10: Simulatore iOS

2.2.3 Objective-C

L'Objective-C è un linguaggio orientato agli oggetti che aggiunge lo stile di passaggio dei messaggi tipico del linguaggio Smalltalk al linguaggio C. Questo linguaggio è un sovra-insieme del linguaggio C, fornendo costrutti che permettono di definire classi ed oggetti. Una volta acquisita la sintassi stile Smalltalk, se si ha già dell'esperienza con linguaggio orientati agli oggetti, Objective-C risulterà abbastanza familiare, anche se ci sono alcune differenze. Una delle più grandi (specialmente per chi come me proviene da Java) sta nel modo di gestione della memoria, sebbene questo sia stato notevolmente migliorato dalla versione 5 di iOS in poi.

Dichiarazione e definizione delle classi. Come nel caso di quasi tutti gli altri linguaggi orientati agli oggetti, in Objective-C le classi forniscono i blocchi costruttivi che permettono l'incapsulamento di dati e metodi che agiscono sui quei dati. Gli oggetti sono una istanza specifica di una classe, e contengono la loro istanza di dati e puntatori ai metodi implementati dalla classe. Le classi vengono specificate in due parti distinte: l'interfaccia e l'implementazione. L'interfaccia contiene la dichiarazione della classe e di

solito è contenuta in file con estensione .h. Invece l'implementazione contiene l'attuale definizione ed è solitamente contenuta in un file con estensione .m.

Gestione della memoria. Il modo in cui la memoria è gestita in Objective-C su iPhone ed iPad non è lo stesso che viene usato da linguaggi quali Java, Perl, Ruby. Se si sta scrivendo una applicazione in Objective-C per il Mac si ha la possibilità di abilitare il garbage collector, mentre su iPhone e iPad si è obbligati ad utilizzare il reference counting, divenuto oggi Automatic Reference Counting (ARC). Un oggetto può essere creato in due modi: allocando manualmente la memoria per l'oggetto tramite il comando alloc ed inizializzarlo tramite il comando init, oppure usando un metodo costruttore. Ma in entrambi i casi l'utente è responsabile del successivo rilascio della memoria precedentemente allocata, tramite il comando release. Tuttavia, nel secondo caso l'oggetto sarò autoreleased alla fine della funzione corrente se esso non è stato esplicitamente preservato. Bisogna fare attenzione però all'utilizzo dell'autorelease perché potrebbe estendere senza che ce ne sia necessità il tempo per il quale l'oggetto viene istanziato, andando ad occupare perciò memoria preziosa. Quando si decide di gestire la memoria manualmente, bisogna fare attenzione a non rilasciare gli oggetti che non si possiedono, essendo sempre sicuri che le chiamate di retain siano sempre bilanciare a quelle di release. Inviare un messaggio di release libererà immediatamente la memoria usata dall'oggetto se quella release porta il retain count dell'oggetto a zero, mentre inviando un messaggio di autorelease si aggiungerà l'oggetto al recipiente di autorelease. L'oggetto sarà quindi rilasciato quando il recipiente è distrutto, tipicamente alla fine della funzione corrente.

Automatic Reference Counting. Sebbene sia molto importante capire come funziona la gestione della memoria in background per la costruzione di applicazioni in Objective-C, l'arrivo dell'Automatic Reference Counting con il rilascio della versione 5 di iOS ha semplificato di molto la gestione della memoria. Per esempio, invece di doversi ricordare manualmente quando usare i vari comandi di retain, release e autorelease, ARC valuta automaticamente

i requisiti degli oggetti a inserisce automaticamente le chiamate ai metodi appropriate quando viene eseguita la compilazione del codice. E' importante sottolineare che ARC è un passo eseguito subito prima della compilazione che aggiunge al posto dell'utente i comandi retain, release, autorelease; quindi in sostanza il reference counting non è scomparso, semplicemente è stato automatizzato.

Schema architetturale. Lo schema architetturale può essere visto come una soluzione implementativa, un template di approccio ai problemi comuni che possono occorrere. Non si tratta di codice ma invece descrive come si dovrebbe modellare l'applicazione in termini delle classi che sono usate. Il framework Cocoa Touch che sta alla base delle applicazioni iOS è basato su uno dei più "vecchi" design pattern, il Model-View-Controller (MVC). Questo pattern è usato per separare la logica del programma dall'interfaccia grafica ed è generalmente il modo più diffuso di costruire le applicazioni iOS. E' utilizzato estensivamente anche dai framework Apple (come ad esempio l'UIKit) e risulterebbe molto difficile scrivere una applicazione iOS senza utilizzare questo pattern nella propria implementazione, oltre che a risultare una scelta senza una valida motivazione.

Il pattern MVC prevede la divisione della applicazione in tre parti funzionali:

- **Model**: formato dalle classi che possiedono i dati dell'applicazione. Gestisce lo stato dell'applicazione (e i dati associati) e solitamente è persistente. E' interamente disaccoppiato dall'interfaccia utente;

- **View**: composta da finestre, oggetti e altri elementi che l'utente può vedere e con cui può interagire. Rappresenta quello che l'utente vede e si occupa della visualizzazione del modello all'utente. E' manipolabile dall'utente e risponde agli eventi generati. Nelle applicazioni iOS la view è generalmente costruita tramite l'interface builder piuttosto che tramite la stesura di righe di codice;

- **Controller**: codice che lega il model e la view. Contiene la logica dell'applicazione che decide come gestire l'input dell'utente. Coordina gli aggiornamenti della view e del model quando l'interazione utente con la view genera cambiamenti al model e viceversa. Solitamente è la parte un cui viene implementata la logica della applicazione.

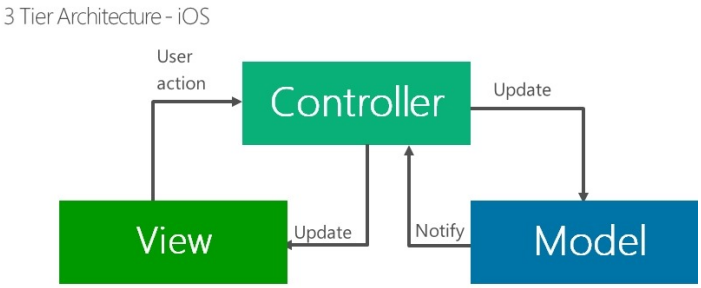


Figura 2.11: Diagramma Model View Controller applicato ad iOS

Solitamente si procede con il creare una view tramite l'interface builder e la classe ViewController ad essa associata che gestisce la view. Può capitare che in base alla complessità, il model sia incorporato all'interno della classe ViewController se la complessità e bassa, altrimenti è necessaria una nuova classe (sottoclasse di NSObject) con una serie di variabili di istanza, metodi di accesso, e metodi personalizzati.

L'obiettivo con MVC è di rendere gli oggetti che implementano queste 3 diverse tipologie di codice distinti e indipendenti il più possibile l'uno dall'altro. Ogni oggetto creato dovrebbe essere chiaramente distinguibile ed identificabile poiché appartenente a una delle tre categorie, con nessuna o poche funzionalità che possono essere classificate come appartenenti alle altre due. Così facendo MVC aiuta ad assicurare la massima riusabilità

View e ViewControllers. Nelle prime versioni dell'SDK, la view veniva implementata esclusivamente tramite la stesura di righe di codice, ma l'introduzione dell'Interface Builder ha semplificato notevolmente le cose. Utilizzando questo strumento, si agisce su un file .nib, che sostanzialmente è un file XML contenente tutti gli elementi che compongono la view. Usando

Interface Builder per creare questi oggetti e definire le relazioni tra di essi e il proprio codice, permette di risparmiare la scrittura di una grande quantità di codice per la gestione della view. E' possibile anche utilizzare una combinazione dei due approcci, ovvero definire lo stato iniziale dell'interfaccia tramite Interface Builder e poi a tempo di runtime modificarla via codice.

I pattern Delegates e DataSource. Un oggetto che implementa un protocollo delegate viene definito come colui che agisce per conto di un altro oggetto. Per poter ricevere notifica di un evento al quale esso dovrà poi rispondere, la classe delegate deve implementare il metodo di notifica dichiarato come protocollo di delegate associato con quell'evento. Solitamente il protocollo specifica un certo numero di metodi che la classe delegate deve implementare. I DataSource sono simili ai delegate, ma invece di delegare il controllo, un oggetto che implementa un protocollo DataSource deve implementare uno o più metodi che forniscano i dati all'oggetto richiedente. Un oggetto "delegante" chiederà al data source quali dati esso dovrebbe mostrare.

Storyboard. A partire dalla versione 5 di iOS ha introdotto un nuovo sistema chiamato Storyboard, che mira a risolvere alcune delle problematiche più comuni che si presentano durante la realizzazione dell'interfaccia grafica. A differenza di prima, ora è possibile lavorare con più view allo stesso tempo, ognuna collegata al proprio view controller, in un workspace visivo unico. E' possibile configurare come le transizioni tra le varie schermate debbano avvenire, attraverso l'invocazione del comando "segue". In sostanza è possibile utilizzare una singola storyboard per rappresentare la maggior parte delle view dell'applicazione, con tutte le interfacce e le loro interconnessioni rappresentati sotto forma di grafico. Tutto ciò porta ad ottenere due vantaggi principali: da un lato è possibile vedere e modificare l'intero flusso di esecuzione dell'applicazione in un solo posto rendendo più facile lavorare osservando interamente l'insieme dei vari componenti; dall'altro evita l'implementazione di una parte di codice dalle classi ViewController.

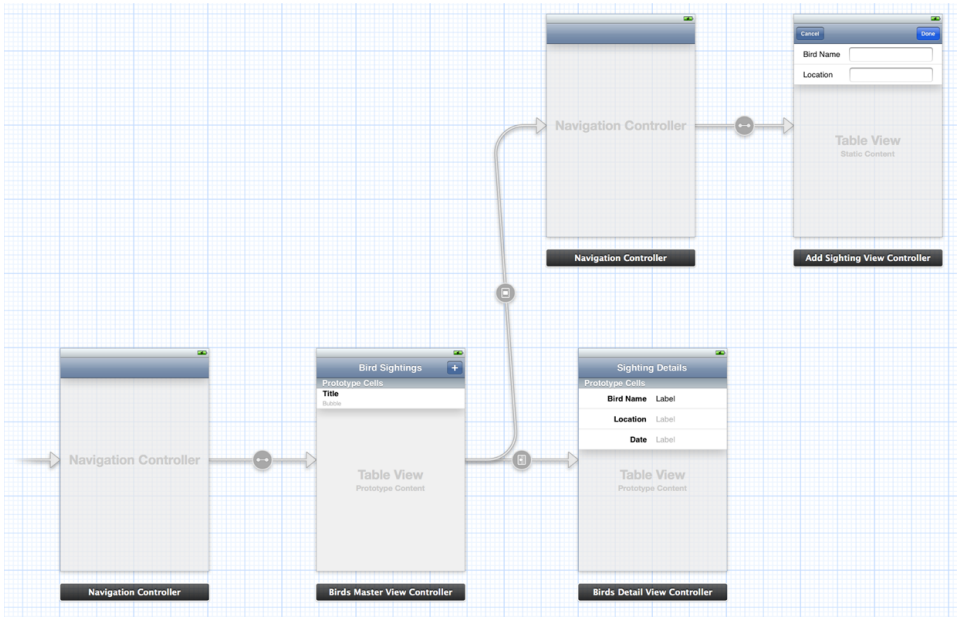


Figura 2.12: Un esempio di Storyboard

2.2.4 Parse

Parse[1] Inc., startup fondata nel 2011 a San Francisco e acquisita da qualche mese da Facebook Inc., è un mobile Backend as a Service (mBaaS) che fornisce un backend completo da affiancare allo sviluppo di applicazione mobile. Consente quindi di focalizzarsi maggiormente sulla creazione dell'esperienza utente e delle funzionalità da offrire senza doversi preoccupare della realizzazione e manutenzione del server e della infrastruttura ad essa associata, eliminando la necessità di scrittura del codice per il lato server.

Usando Parse lo sviluppatore può impostare ed utilizzare il backend molto velocemente, ottenendo funzionalità quali la gestione del database, autenticazione utente, notifiche push e salvataggio di dati. Con la sottoscrizione al servizio gratuita si ottengono, mensilmente, 1 milione di richieste API, 1 milione di notifiche push e 1 GB di spazio, quindi più che sufficienti per

[1]Parse: http://www.parse.com/

svolgere questo progetto di tesi; ovviamente per chi ha esigenze professionali, sono disponibili anche piani di sottoscrizione a pagamento.

Capitolo 3

Progettazione

Il nome che ho deciso di dare alla mia applicazione è TrainSpotter (letteralmente dall'inglese osservatore di treni) in modo che comunichi immediatamente all'utilizzatore l'idea di che cosa offre l'applicazione. Si tratta di una applicazione sviluppata per la piattaforma iOS, specificatamente per iPhone 4S (il dispositivo in mio possesso) che intende misurare attraverso le segnalazioni degli utenti la qualità del viaggio in treno, rendendola disponibile per la consultazione a chi deve ancora salire in carrozza. Questo progetto è nato da un idea del sottoscritto e del Prof. Vittorio Ghini, entrambi frequentatori assidui di treni ed entrambi troppo spesso scontenti del servizio che ci viene offerto, che quindi ci ha portato ad elaborare una possibile soluzione a questo problema. In questo capitolo descriverò il sistema da realizzare, presentando prima gli obiettivi da raggiungere, poi le scelte implementative adottate, i requisiti da rispettare ed infine le metodologie utilizzate.

3.1 Obiettivi: cosa deve fare l'applicazione?

L'obiettivo principale dell'applicazione sviluppata è quello del monitoraggio e della valutazione della qualità dei viaggi ferroviari. Ciò equivale alla raccolta di una serie di dati e parametri definiti in precedenza, grazie al contributo diretto degli utenti. Questi dati dovranno essere in un primo momento visibili

a tutti gli utilizzatori dell'applicazione e dovranno consentire di avere una serie di informazioni aggiornate in tempo reale sui treni che, ad esempio, metteranno in condizione di decidere di prendere un certo treno o meno sulla base della valutazione ricevuta da quel specifico treno. Inoltre in un secondo momento questi dati potranno essere raccolti e analizzati per stilare analisi più dettagliate, utili anche per individuare le tratte che necessitano di interventi più urgenti rispetto ad altre.

Una caratteristica fondamentale che dovrà essere alla base del funzionamento dell'applicazione è quindi il crowdsourcing, ovvero saranno gli utenti, gli stessi utilizzatori che forniranno i dati in prima persona, dati essenziali al funzionamento della stessa. L'esempio forse più famoso e calzante di questa tipologia di modello di raccolta dati è Wikipedia, la più famosa e completa enciclopedia online; mentre per quanto riguarda prettamente le applicazioni mobile, un'applicazione che mi ha ispirato nella realizzazione di TrainSpotter è sicuramente Waze. Waze è una applicazione mobile distribuita gratuitamente adibita alla navigazione satellitare, utilizzabile su tutte le principali piattaforme mobile del momento.

Figura 3.1: Alcune schermate dell'applicazione Waze

Si differenzia dai sistemi di navigazione GPS tradizionali in quanto, oltre ad essere distribuita gratuitamente, invece di avere al suo interno mappe precaricate e precedentemente raccolte da enti professionali, grazie al contributo dei suoi utilizzatori è in grado di fornire mappe dettagliate e costantemente

aggiornate, unite a servizi aggiuntivi quali segnalazioni in tempo reale su traffico o incidenti o altri eventi.

Tornando a TrainSpotter, l'applicazione sarà divisa in tre parti, ognuna con una funzione principale associata:

- una prima parte servirà all'inserimento di dati da parte dell'utilizzatore che andranno poi a formare quello che sarà il database di valutazione disponibile a tutti gli utenti;
- la seconda parte invece servirà alla semplice fruizione di contenuti, sia generati dagli utenti, o da servizi web pre-esistenti;
- la terza e ultima parte sarà quella dedicata all'interfacciamento con i sensori di movimento e alla disponibilità di servizi aggiuntivi.

L'obiettivo finale dovrà essere quindi quello di poter usufruire del contributo degli utenti che saranno direttamente coinvolti nell'azione (che in questo caso è il viaggio in treno) per ottenere informazioni aggiornate, dettagliate e possibilmente imparziali (oggettive). Tutti i dati e le informazioni raccolte sono in un primo momento utili e usufruibili da tutti gli utilizzatori dell'applicazione e in un secondo momento, potranno essere utilizzati per stilare delle statistiche che aiuteranno ad esempio ad indirizzare gli interventi da effettuare per migliorare la mobilità su treno ed offrire una esperienza di viaggio migliore.

3.2 Scelta del dispositivo mobile: iPhone

La mia scelta della piattaforma mobile su cui sviluppare il mio progetto di tesi è ricaduta su iOS e di conseguenza su iPhone. Oltre ad una mia preferenza personale, ritengo che iOS sia al momento la piattaforma di sviluppo mobile di riferimento. E' uno dei sistemi operativi mobile più diffuso e utilizzati al mondo, che conta centinaia di migliaia di dispositivi tra iPhone, iPod Touch e iPad.

L'ambiente di sviluppo offerto da Xcode è molto potente, supportando e agevolando il lavoro dello sviluppatore. Integra efficacemente lo sviluppo

del codice da un lato, e il design di interfacce dall'altro, tutto all'interno dello stesso strumento. Include inoltre strumenti aggiuntivi per l'esecuzione di test, analisi e debug del codice e monitoraggio delle perfomance dell'applicazione. Offre un meccanismo di source control offrendo la possibilità di creare, cambiare o unire branch di sviluppo differenti. Infine rende disponibile un simulatore su cui compilare, eseguire e fare debug delle proprie applicazioni.

Per quanto riguarda il sistema operativo mobile, Apple con il rilascio della versione 7 di iOS, ha fatto un ulteriore balzo in avanti, migliorando il già eccellente sistema versione 6. Anche se per alcuni le decisioni prese in relazione al design possono essere discutibili, iOS 7 risulta piacevole all'utilizzo e fluido rispetto ai competitors. Uno dei motivi della sua enorme diffusione è la qualità del sistema operativo, apprezzabile se ad esempio ci soffermiamo su 4 criteri di valutazione:

- il carico cognitivo può essere definito come la somma degli elementi a cui l'utente si deve abituare per poter usare il dispositivo senza problemi, uno degli aspetti chiave dell'esperienza per utenti non esperti. Il carico cognitivo viene calcolato tramite il numero della applicazioni e i widget, le icone e altri elementi dell'interfaccia presenti nell'istallazione di default del sistema. Nel caso di iOS richiede uno sforzo minimo;

- l'esperienza utente degli smartphone dipende in gran parte sulla facilità d'uso e efficienza d'integrazione delle caratteristiche e servizi fondamentali, in cui ad esempio sono presi in considerazione l'accesso alle impostazioni, integrazione con le notifiche, multitasking, accesso alla fotocamera. iOS offre efficienza a integrazione completi, con un'area personalizzabile per le notifiche (Control Center), un multitasking e switch tra applicazioni ben implementato, e scorciatoie per le funzioni principali.

- per quanto riguarda la personalizzazione da parte dell'utente, iOS offre il minimo indispensabile per l'interfaccia, come personalizzione sfondo

o supporto dinamico dei caratteri ma buone funzionalità per quanto riguarda opzioni per l'accessibilità;

- le problematiche nell'esperienza utente comprendo tutto quelle situazioni in cui un dispositivo non fa quello che noi ci aspettiamo, oppure manca di funzioni basilari che dovrebbero essere disponibili. Anche qui iOS risulta tendenzialmente superiore ai concorrenti, grazie soprattutto al suo carico cognitivo molto basso;

Infine, un'altra caratteristica degna di nota visto il contesto in cui andrò ad operare: con il rilascio della versione 7 di iOS, Apple ha introdotto nel suo sistema operativo il supporto al protocollo di rete Multipath TCP[15] (MPTCP), permettendo al device di utilizzare simultaneamente interfacce di rete multiple (ad esempio 3G e WiFi) per il trasferimento di dati. Uno dei benefici rispetto al protocollo TCP tradizionale è la possibilità di inviare dati attraverso la rete che in quel momento opera in maniera più efficiente, riducendo al minimo la perdita di pacchetti. E quando un canale non sarà più disponibile, l'altro continuerà a funzionare rendendo il canale di comunicazione più robusto, risultando in una utilizzazione della rete migliorata e throughput maggiore.

3.3 Scelta del backend: Parse

Parse è uno dei tanti mobile Backend as a Service (mBaaS) che fornisce un backend completo da affiancare allo sviluppo di applicazione mobile, consentendo quindi di focalizzarsi maggiormente sulla creazione dell'applicazione client.

Tra le funzionalità offerte come già accennato, troviamo l'invio di notifiche push, salvataggio di dati nel cloud, integrazione coi social network Twitter e Facebook. L'SDK nativo di Parse è disponibile per le principali piattaforme desktop e mobile come iOS, OS X, Android, Windows 8, Windows Phone

8, JavaScript, .NET[1], Xamarin[2] e Unity[3]. Parse gestisce automaticamente il salvataggio di dati nel cloud e l'interrogazione del database tramite query. Tramite il data browser è possibile gestire, cercare e aggiornare i contenuti memorizzati.

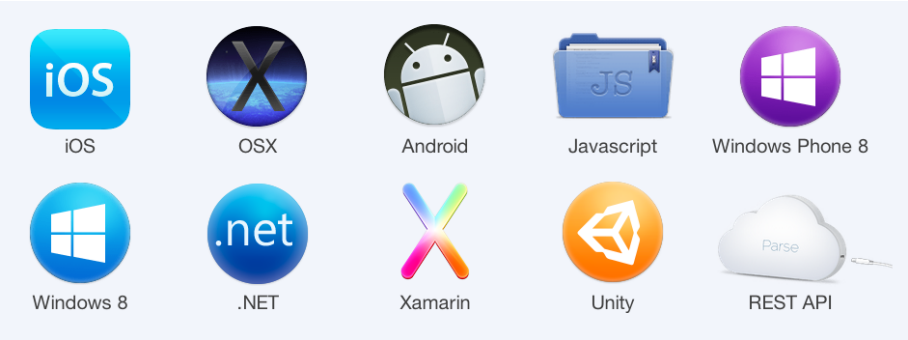


Figura 3.2: Piattaforme supportate da Parse

Nello specifico, Parse offre le seguenti funzionalità:

- **salvataggio e recupero dei dati**: funzione fondamentale nel caso in cui, specialmente per applicazioni mobili, c'è necessità di condivisione dei dati tra più utenti;
- **API e SDK**: i servizi BaaS forniscono un certo numero di SDK, ognuno specifico ad una piattaforma client, che offrono un livello di astrazione per poter lavorare con il servizio. Inoltre forniscono anche API REST che permettono ad altre piattaforme di inter-operare con esso. Esse permettono l'interazione con Parse da tutto ciò che può inoltrare delle richieste HTTP, come ad esempio Javascript;

[1]Framework di sviluppo per applicativi .NET: http://www.microsoft.com/net
[2]Piattaforma di sviluppo cross-platform: http://xamarin.com/
[3]Ambiente di sviluppo videogiochi 3D: http://unity3d.com/

- **query di dati**: laddove il semplice recupero di dati non è sufficiente, è possibile effettuare delle query più dettagliate;
- **storage di dati**: dato che gli smartphone spesso sono equipaggiati con dispositivi di acquisizione digitale di dati come fotocamere o microfono, Parse consente l'upload di questi dati;
- **security**: offre vari metodi per la protezione di dati come l'impostazione di lettura/scrittura dati in base allo status o il ruolo dell'utente loggato;
- **autenticazione**: offre la possibilità di gestire gli utenti tramite registrazione, login, recupero password e integrazione con i social network (Twitter e Facebook);
- **notifiche push**: diventate ormai fondamentali strumenti per la notifica dell'utente, Parse consente di inviare delle notifiche push anche quando l'applicazione sta venendo eseguita in background.

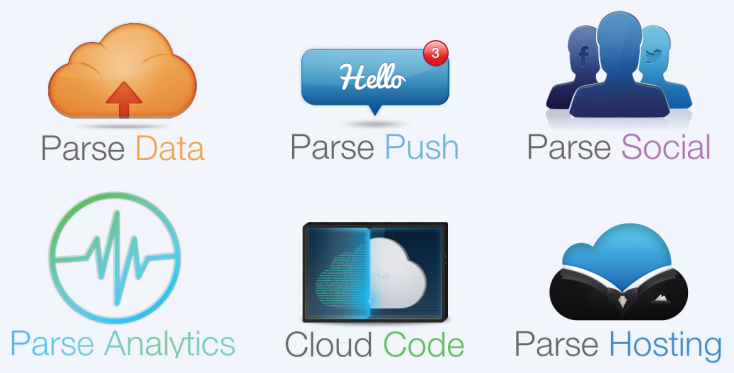


Figura 3.3: Funzionalità principali offerte da Parse

Il primo passo nell'utilizzo è la creazione di una app di Parse relativa alla applicazione mobile che si sta sviluppando: ogni app ha un suo id e client key che la identificano e vengono associati all'installazione dell'SDK. Per quanto

riguarda la sicurezza vengono utilizzate delle Access Control List (ACL); per ogni oggetto creato è possibile specificare quale utente ha permessi di lettura e quali hanno permessi di scrittura.

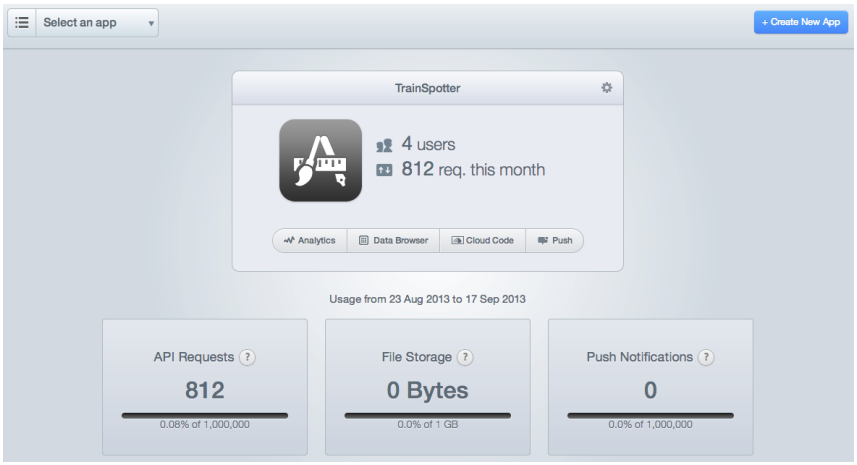


Figura 3.4: Dashboard della piattaforma Parse

La mia scelta è caduta su Parse per una serie di motivi. Essendo oltre 50 i fornitori di questo tipo di servizi era impossibile per motivi di tempo esaminarli uno ad uno per capire quello che meglio si adattasse alle mie necessità. Quindi, dopo qualche ricerca online, ho deciso di utilizzare Parse perché era uno dei BaaS più utilizzati (circa 100.000 applicazioni utilizzano questa piattaforma), è stato uno dei precursori, offriva un piano gratuito ed offriva tutte le funzionalità di cui avevo bisogno. Una dei punti di forza è la semplicità, non dovendosi preoccupare di aspetti quali mantenimento del server, della sicurezza e della scalabilità. Dietro c'è una comunità attiva di sviluppatori ed il supporto dal team di Parse è valido. Infine la documentazione fornita è ottima.

3.4 Requisiti funzionali

I requisiti funzionali vanno a definire che cosa il sistema da sviluppare dovrà fare, le condizioni necessarie al funzionamento del sistema e le funzioni che esso deve fornire ai suoi futuri utilizzatori. L'obiettivo della definzione di tali requisiti è la specifica dei processi che devono essere supportati dall'applicazione, ovvero l'insieme delle attività eseguite dagli utenti quando interagiscono con l'applicazione.

Il sistema in questione, dovrà essere sviluppato cercando di soddisfare i seguenti requisiti funzionali:

- per essere utilizzabile, l'applicazione richiede obbligatoriamente una connessione internet (indifferentemente WiFi o cellulare);
- per utilizzare l'applicazione, è obbligatorio creare un account utente personale;
- per inviare una valutazione di un treno, prima dovrà essere effettuato il check-in sullo stesso;
- l'applicazione dovrà, tramite l'input fornito dall'utente, permettere di creare una valutazione di qualità del viaggio ferroviario;
- i dati inviati dovranno essere salvati in un database e resi accessibili agli altri utenti;
- l'applicazione dovrà offrire servizi di mapping e posizionamento.

3.5 Requisiti non funzionali

I requisiti non funzionali vanno a definire il modo in cui il sistema dovrà operare, definiscono cioè proprietà e vincoli sul sistema. Essi riguardano alcuni aspetti che sono rilevanti per il raggiungimento degli obiettivi preposti ma non sono correlati in modo specifico alle funzionalità del sistema.

I requisiti non funzionali comprendono problematiche, aspetti tecnici e di comunicazione.

Il sistema in questione, dovrà essere sviluppato cercando di soddisfare i seguenti requisiti non funzionali, in modo da assicurare affidabilità ed efficienza:

- usabilità: l'applicazione dovrà essere facile ed intuitiva da usare, le interfacce dovranno essere semplici e contenere oggetti di interazione (menu, pulsanti) che aderiscono a standard consolidati, fornire meccanismi di orientamento ed aiuto;
- prestazioni: l'applicazione dovrà sfruttare le risorse a disposizione in maniera efficiente e senza sprechi;
- scalabilità: le prestazioni dell'applicazione non dovranno essere condizionate dall'incremento del volume delle richieste effettuate;
- sicurezza: l'applicazione dovrà implementare meccanismi di protezione dell'integrità, confidenzialità e privatezza dei dati, di autenticazione degli utenti e di protezione del flusso di scambio di informazioni tra utenti e applicazione;
- manutenibilità: il processo di sviluppo adottato dovrà essere organizzato e implementato in modo da garantire la correzione degli errori e l'aggiunta di nuove funzionalità, ad esempio adottanto una architettura software modulare che permetta allo sviluppatore di intervenire separatamente sulle diverse parti che compongono il sistema.

3.6 Architettura del sistema

Dal punto di vista dell'architettura di rete, il sistema che dovrà essere implementato adotterà una classica configurazione client-server, dove nello specifico, il dispositivo mobile (client) si connette al backend (server) per poter usufruire di certi servizi che esso mette a disposizione, come ad esempio la

richiesta di dati o la condivisione di una certa risorsa. L'esistenza del server permette ad una moltitudine di client di condividere tra di loro le risorse disponibili, lasciando che esso gestisca gli accessi per evitare dei conflitti.

In questo caso il client è l'applicazione iOS da realizzare mentre il server è costituito dal mobile Backend as a Service scelto, ovvero la piattaforma Parse. L'applicazione mobile sarà composta da tre macro aree, ognuna con delle funzioni specifiche. La prima area sarà quella adibita alla raccolta dei dati sulla qualità dei viaggi ferroviari da parte dell'utente. La seconda area sarà quella adibita alla fruizione dei contenuti e risorse salvate sul server, mentre la terza e ultima area raggrupperà alcune funzioni e servizi di utilità. Per quanto riguarda il server, come già detto in precedenza nella descrizione di Parse, esso servirà al salvataggio dei dati, alla condivisione degli stessi tra tutti gli utilizzatori del sistema, offrendo anche funzionalità aggiuntive come invio di notifiche push, integrazione coi social e gestione degli utenti.

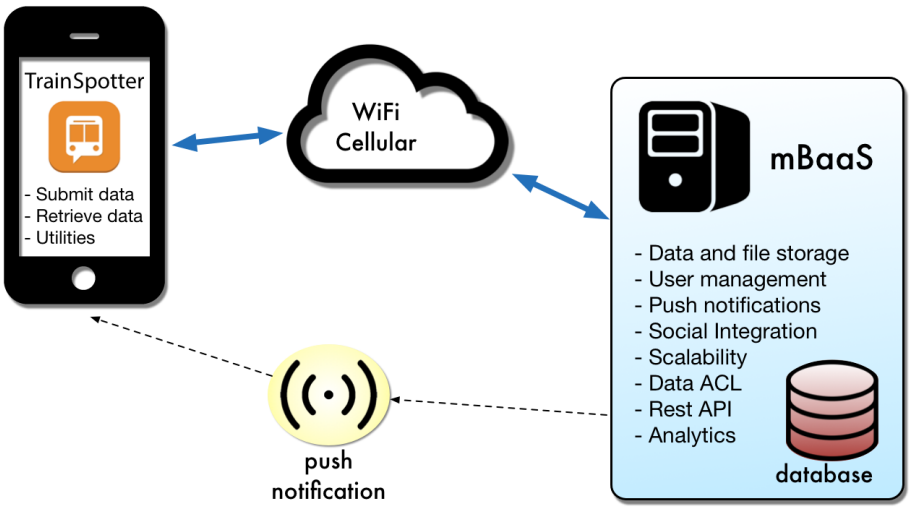


Figura 3.5: Scherma architetturale del sistema progettato

3.7 Problematiche software

Nel progettare un applicazione per il sistema operativo iOS, è necessario valutare e tenere presente una serie di limitazioni software che il sistema target ci impone:

- **una sola applicazione attiva**: solo una applicazione alla volta può essere attiva e visualizzata sullo schermo del dispositivo in un singolo istante di tempo. Anche se dall'arrivo di iOS 4 in poi, è stata implementata una modalità di esecuzione delle applicazioni in background, ciò è limitato ad una serie di funzionalità come la riproduzione audio e la geolocalizzazione;

- **una sola finestra**: mentre i sistemi operativi desktop permettono l'esecuzione di programmi che comprendono e gestiscono finestre di controllo multiple, iOS mette a disposizione dell'applicazione una sola finestra con cui poter interagire, implicando quindi che tutte le operazioni svolte dall'utente devono avvenire all'interno dell'unica schermata presente, che ha dimensioni pari alla grandezza dello schermo del dispositivo;

- **accesso limitato al sistema**: mentre ad esempio sul personal computer i programmi hanno più o meno accesso a tutte le funzionalità del sistema, su iOS sono presenti molte restrizioni. Ad esempio è possibile leggere o scrivere dati solo nella sezione del file system che è stata creata e dedicata per l'applicazione (chiamata application sandbox); in questo spazio possono essere salvati documenti, preferenze dell'utente o altri tipi di dato. Le restrizioni riguardano anche il tipo di accesso, negando tipicamente l'esecuzione di tutte quelle operazioni che richiedono privilegi di admin o root;

- **tempo di risposta limitato**: alla base dell'utilizzo del sistema iOS c'è una elevata velocità di interazione e presentazione e lo stesso deve valere per le applicazioni di terze parti. Quando una applicazione viene

lanciata devono essere caricate immediatamente le preferenze, i dati e la view principale. Una volta lanciata, se un utente preme il tasto home (che ha il compito di chiudere l'applicazione correntemente in uso e ritornare alla schermata principale), tutto ciò che era in esecuzione in quel momento deve essere salvato prima che l'applicazione venga terminata e nel caso il tempo impiegato sia maggiore di 5 secondi, il processo viene chiuso forzatamente con la conseguente perdita di dati. Questo però non è del tutto vero perché a partire da iOS 4 la situazione è migliorata con l'introduzione di nuove API che permettono all'applicazione di "chiedere" tempo aggiuntivo per terminare la propria esecuzione;

- **dimensione dello schermo limitata**: sebbene sia uno dei migliori display disponibili attualmente per smartphone, ovviamente esso non possiede le dimensioni di un normale schermo di un sistema desktop. Con una risoluzione di 320x480 per i primi modelli, di 640x960 per iPhone 4 e 4S e di 640x1136 dell'iPhone 5, 5C e 5S, bisogna comunque tenere in considerazione che tutti questi pixel sono "incastrati" in un display di dimensioni molto contenute (rispettivamente 3.5" e 4"), seppure esso risulti molto definito;

- **risorse di sistema limitate**: i dispositivi iOS disponibili ora in commercio posseggono un quantitativo di rispettivamente 512/1024 Mb di memoria RAM, ma una parte di essa (circa la metà) è costantemente utilizzata dal sistema operativo e da altre applicazioni eventualmente aperte in background. Inoltre a differenza dei sistemi operativi desktop che in casi estremi fanno uso del cosiddetto file di swap[4], ciò non avviene su iOS e quindi si deve fare affidamento esclusivamente sulla memoria attualmente non occupata. E' disponibile inoltre per l'applicazione correntemente in uso un meccanismo integrato di notifica che

[4]Con il termine swap si intende l'aumento di capacità della memoria volatile oltre il limite della quantità di RAM, attraverso l'utilizzo di spazio fisico di memorizzazione aggiuntivo.

avvisa che la memoria disponibile sta per terminare e quindi suggerisce di provvedere a liberarne un po' per evitare il rischio di terminazione improvvisa;

- **niente garbage collector**: il sistema iOS non prevede il supporto al classico garbage collector, lasciando così in mano dell'utente la gestione della memoria nelle prime versioni del sistema, ed implementando il meccanismo di Automatic Reference Counting (ARC) sulle le versioni più recenti;

- **approccio differente all'interazione**: i dispositivi mobile moderni sono sprovvisti di tastiera fisica e mouse, implicando perciò una differente modalità di interazione con l'utente. Di conseguenza anche le interfacce vanno adattate all'utilizzo senza l'ausilio di periferiche hardware.

3.8 Applicazione web vs. applicazione nativa

Una applicazione web può essere vista come un sito web particolare, ottimizzato specificatamente per iPhone. L'interfaccia utente è costruita tramite le tecnologie web standard, è raggiungibile ad un determinato URL (che può essere pubblico, privato o accessibile tramite login) ed è ottimizzata sulla base delle specifiche dell'iPhone. Infine, una applicazione web non si installa sul telefono, non è disponibile sull'App Store e non è scritta con il linguaggio Objective-C.

Contrariamente, le applicazioni native sono installate sul telefono, hanno accesso all'hardware integrato (accelerometro, fotocamera), sono scritte in Objective-C e sono disponibili tramite App Store, una caratteristica che negli ultimi anni ha coinvolto un gran numero di sviluppatori a livello mondiale.

A seconda della loro finalità le applicazioni posseggono diversi requisiti, per cui risulta che alcune sono maggiormente indicate per essere sviluppate sotto forma di applicazioni web piuttosto che altre; è quindi un passo fonda-

mentale analizzare i pro e contro di ciascuna variante per capire quale sia la più adatta al proprio ambito.

Con riguardo alla applicazioni native, possiamo individuare i seguenti vantaggi:

- Xcode, Interface Builder e il framework Cocoa Touch sono un insieme di strumenti molto potenti e ben integrati tra di loro;
- tramite le API fornite da Apple è possibile accedere a tutta una serie di caratteristiche e funzionalità hardware del dispositivo;
- la pubblicazione in App Store e la conseguente visibilità rendono semplice ed immediato l'installazione da parte degli utenti.

e i seguenti svantaggi:

- per diventare uno sviluppatore Apple è necessario pagare una quota annua di 80 Euro. E' prevista anche una iscrizione gratuita ma è limitata al solo testing sul simulatore;
- il processo di approvazione delle applicazioni può risultare lungo e complicato, poiché ci sono una serie di direttive dettate da Apple da dover rispettare;
- è necessario possedere un Mac per usufruire dell'ambiente di sviluppo;
- è necessario sviluppare tramite Objective-C;
- il ciclo di sviluppo e la correzione di eventuali bug sono processi che richiedono tempo poiché occorre inviare l'applicazione corretta ad Apple ed aspettare che essa la verifichi e se l'esito sarà positivo, la pubblichi.

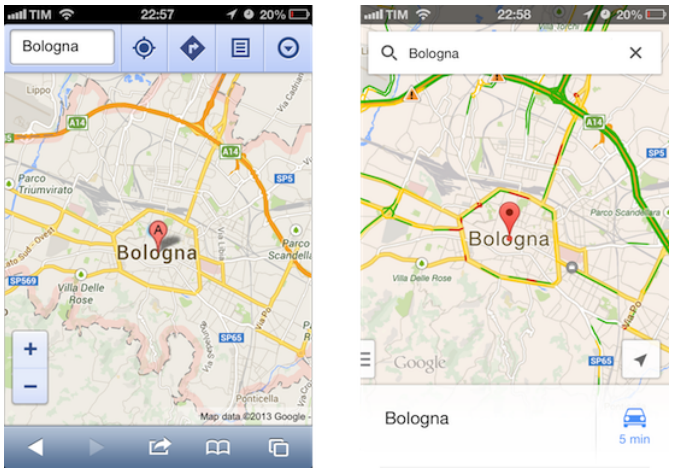


Figura 3.6: Screenshots Google Maps: a sinistra l'applicazione web, a destra quella nativa.

Riguardo le applicazioni web invece, possiamo individuare i seguenti vantaggi:

- gli sviluppatori web possono utilizzare linguaggi e tecnologie che già conoscono;
- l'ambiente di sviluppo è indipendente dalla piattaforma che si utilizza, non è necessario possedere un Mac;
- l'applicazione potrà essere eseguita su qualsiasi dispositivo dotato di un browser web;
- il ciclo di sviluppo e la correzione di eventuali bug richiedono poco tempo poiché basta semplicemente caricare la versione aggiornata senza dover aspettare controlli e verifiche.

Svantaggi applicazioni web:

- non è possibile accedere a tutte le caratteristiche hardware del dispositivo;

- bisogna implementare una personale procedura di pagamento se si vuole monetizzare l'applicazione;

- risulta difficile implementare interfacce grafiche sofisticate.

Per la realizzazione di questo progetto di tesi, ho scelto la strada dell'applicazione nativa. La motivazione ovvia per utilizzare l'SDK nativo è per fare cose che non possono essere fatte usando esclusivamente le tecnologie web. Ad esempio le applicazioni di realtà aumentata hanno bisogno di una stretta integrazione coi sensori presenti sull'iPhone (GPS, accelerometro, fotocamera) e non sarebbe possibile realizzarle senza accedervi. Lo stesso discorso vale per questo caso: avendo una forte componente di interazione con i sensori di movimento (accelerometro, giroscopio e magnetometro) e periferiche hardware (microfono, fotocamera, GPS) l'applicazione dovrà essere forzatamente una applicazione nativa, guadagnandone anche in efficienza e prestazioni.

Questa però non vuole essere una critica a linguaggi come HTML, CSS e JavaScript. Queste tecnologie web sono solide e in costante miglioramento e l'attrazione di avere del codice cross-platform è molto forte dal punto di vista dello sviluppatore. Tuttavia per gli utenti, non è così ovvio. Molte delle motivazioni che supportano l'uso di tecnologie legate al web appartengono alla prospettiva dello sviluppatore. Anche se non è tutto quello che conta, bisognerebbe tenere in considerazione il punto di vista degli utenti, perché ad essi interessa veramente la questione principale: l'usabilità.

I potenziali utilizzatori e clienti non compreranno una applicazione solo perché essa supporta anche altre piattaforme; invece vorranno una applicazione che sia stilisticamente conforme al resto della loro piattaforma in uso, che segua gli stessi paradigmi per l'interfaccia come fa il resto delle altre applicazioni. Quindi se l'applicazione è integrata nell'ecosistema iOS, se fa uso delle possibilità che l'hardware offre, e si assicura che l'interfaccia sia ottimizzata per il dispositivo, l'esperienza finale dell'utente ne risulterà molto migliore. Tutto questo è possibile anche usando le tecnologie web nominate precedentemente, ma il più delle volte è veramente difficile scrivere applica-

zioni non native che funzionino bene su più piattaforme. Ed è ancora più difficile farle sembrare native su una moltitudine di diverse piattaforme.

Anche se il browser Safari su iOS supporta le funzionalità di geolocalizzazione (grazie al supporto dell'HTML 5) ciò non risolve completamente il problema. Non è sicuro che l'hardware di ogni piattaforma sarà gestito nella stessa maniera tramite HTML 5, ad esempio.

Qualche volta non si tratta tanto del fatto che le cose non possano essere fatte o meno, ma quanto al fatto di farle in maniera più veloce ed efficiente. Ad esempio le applicazioni iTunes ed App Store che sono presenti su iOS sono in realtà delle applicazioni web "racchiuse" da applicazioni native. Ciò significa che anche se le applicazioni non possono fare granché senza una connessione internet, almeno possono essere avviate senza difficoltà.

Ma questi sono esempi in un certo senso "estremi". Un sacco di applicazioni presenti nell'App Store combinano dati recuperati da remoto ed una interfaccia nativa. Senza l'accesso alla rete, alcune delle interfacce grafiche sono disabilitate. Tuttavia le applicazioni native possono essere costruite per funzionare anche se la connessione al network del device non è presente o non lo è mai stata: l'utente può usare comunque la parte dell'applicazione che non necessita di una connessione per funzionare.

Le performance della rete affliggeranno la percezione dell'utente della velocità: il rendering dell'interfaccia utente mentre viene eseguita una richiesta per popolarla permette alla applicazione di rimanere "responsive" all'interazione dell'utente anche quando si rimane in attesa della connessione.

Per non parlare poi dello sviluppo di videogames. E' impossibile sviluppare giochi avanzati usando le tecnologie web. Un grande vantaggio della applicazioni native è, nonostante l'affollamento dell'App Store, l'esposizione. Se nessuno riesce a trovare la tua applicazione, nessuno te la pagherà, ed il web è uno spazio sconfinato. Tuttavia, a meno che non siamo particolarmente fortunati o che l'applicazione diventi virale, è comunque necessaria una operazione di marketing. Ad esempio attraverso i codici promozionali che permettono il download gratuito della applicazione. Oppure molti sviluppa-

tori sono soliti inviare delle copie delle loro applicazioni a blog o testate giornalistiche sperando che questi la recensiscano e pubblicizzino. Oppure ancora producendo uno "screencast" che mostri il funzionamento dell'applicazione e come utilizzarla al meglio; oppure offrendo un buon supporto tecnico.

Spesso il buon design significa aderire allo stile Apple, integrare l'applicazione con le altre già presenti. Non è necessario reinventare la ruota, ma utilizzare elementi grafici e interfacce familiari agli utenti iOS.

Come nota conclusiva, vorrei fare presente che ci sono disponibili dei progetti di terze parti, di cui il più noto risulta essere PhoneGap[5], che permette di sviluppare applicazioni tramite le più famose tecnologie web e poi distribuirle sotto forma di applicazioni native per le più note piattaforme mobile accennate in precedenza.

3.9 Ciclo di sviluppo

Per questo tipo di progetto ho deciso di adottare un approccio di sviluppo top-down, dove nella prima fase si formula una visione generale del sistema, senza dettagliare le sue parti. Successivamente, ogni parte del sistema è rifinita, aggiungedo maggiori dettagli. Ogni parte così ottenuta può essere ulteriormente definita specificando ulteriori dettagli, finchè la specifica completa è sufficientemente dettagliata.

Il primo step nello sviluppo di una applicazione (non importa se desktop, mobile o per quale piattaforma) è logicamente avere in mente un'idea di quello che si vorrebbe realizzare e possibilmente assicurarsi che non ci sia niente di identico in circolazione. Inquadrata l'idea, successivamente viene la fase di brainstorming per individuare le caratteristiche principali ed aggregare tutti i pensieri raccolti. Poi è utile assegnare le priorità, in modo da poter separare ad esempio ciò che l'app dovrà possedere nella sua versione iniziale e ciò che invece per il momento di può tralasciare ed implementare nelle versioni successive.

[5]Sito ufficiale: http://docs.phonegap.com/en/edge/guide_platforms_index.md.html

La fase successiva è quella di design: nel contesto specifico se non si ha abbastanza esperienza in questo campo, è consigliabile leggere il documento fornito da Apple intitolato "iOs Human Interface Guidelines" che fornisce linee guida per gli scenari più comuni ed una serie di regole a cui una applicazione iOS dovrebbe essere conforme. Ad esempio è consigliabile seguire il look and feel delle applicazioni standard Apple oppure creare qualcosa di totalmente differente, ma non mescolare le due cose. Oppure anche prendere spunto da altre applicazioni, capirne i punti di forza e di debolezza in modo da non commettere gli stessi errori. Personalmente per la fase di design mi sono cimentato prima col disegno su carta delle varie schermate e poi sono passato all'implementazione.

Una volta stesa l'ossatura ed il flusso di utilizzo è possibile passare alla fase di stesura del codice. Anche qui se non si ha esperienza, è possibile consultare i documenti forniti da Apple "iOS Application Programming Guide" e "iOS Development Guide" per ottenere una infarinatura generale. E' consigliabile seguire un approccio di sviluppo modulare, nel senso che ogni schermata possiede la sua classe ViewController associata, separato dagli altri e quindi riutilizzabile.

Infine abbiamo la fase di test, sia sul simulatore iOS compreso in Xcode oppure ancora meglio sul device fisico, in modo da testare le funzionalità hardware che non possono essere simulate. Un altro strumento importante è la distribuzione Ad Hoc del software in via di sviluppo. Rappresenta l'ultimo step prima della distribuzione nell'App Store e consente allo sviluppatore di eseguire test allargati comprendendo fino ad un massimo di 100 dispositivi.

Capitolo 4

Implementazione

In questo capitolo elencherò e descriverò tutte le funzionalità implementate dall'applicazione TrainSpotter.

4.1 Panoramica del sistema

Come passo successivo alla definizione dell'oggetto della mia applicazione, ho incominciato a pensare alla strutturazione dell'interfaccia: nello specifico a quali funzioni fare apparire a video, come collegare un'interfaccia all'altra, a quali opzioni fornire all'utente e quali elementi grafici mostrare e come disporli a video. Prima di passare direttamente alla progettazione con Xcode, ho fatto qualche prova su carta per vedere a grandi linee quale sarebbe stato il risultato.

Una volta definite le linee guida dell'interfaccia, sono passato all'implementazione della stessa tramite Xcode seguendo il metodo di implementazione esaminato prima, la storyboard. A differenza del metodo precedente -ovvero lavorando con i file relativi all'interfaccia .xib separati uno dall'altro- la storyboard permette di avere una panoramica esaustiva di tutte le view create e delle connessione tra esse.

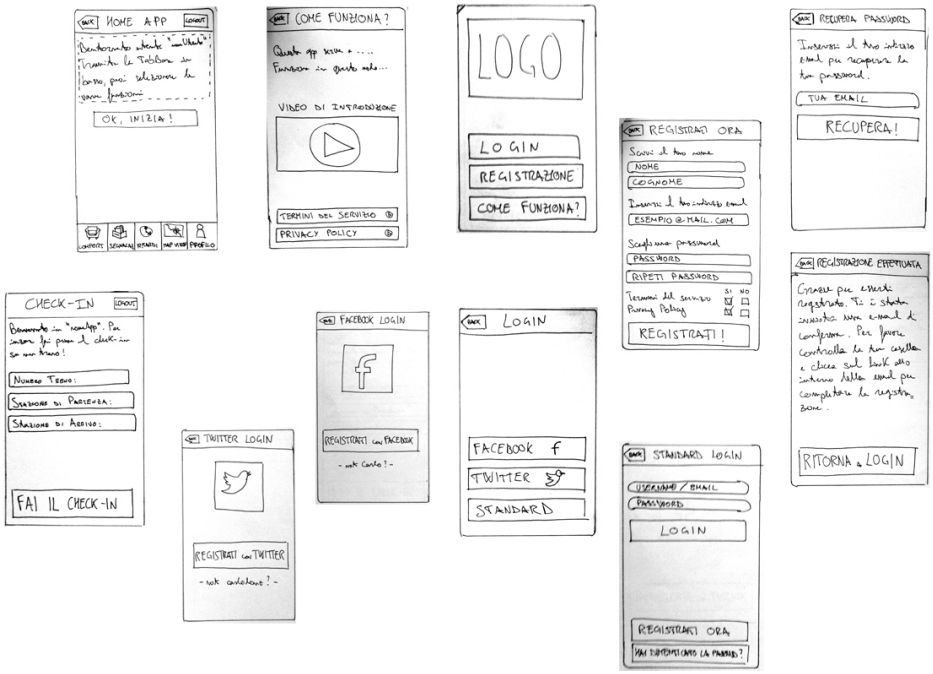


Figura 4.1: Alcuni prototipi di interfaccia disegnati a mano

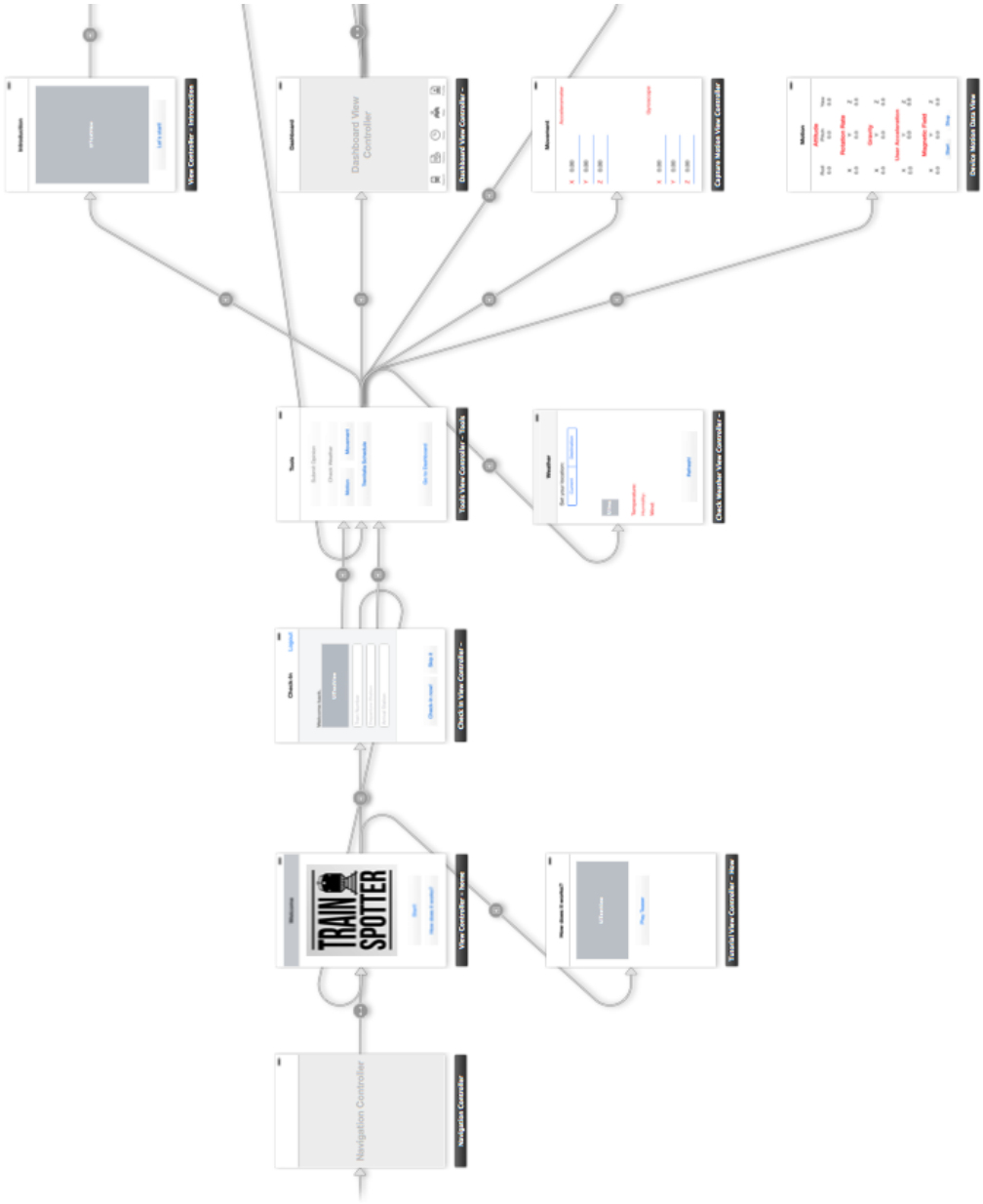


Figura 4.2: Storyboard dell'applicazione TrainSpotter - Prima parte

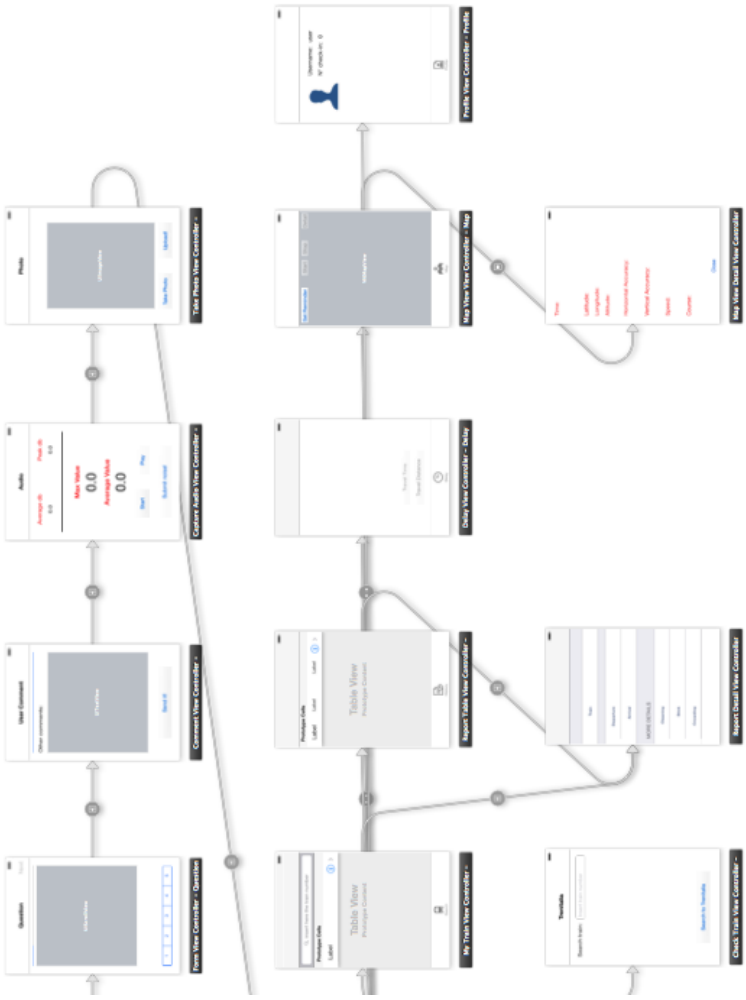


Figura 4.3: Storyboard dell'applicazione TrainSpotter - Seconda parte

Una volta creata l'ossatura, anche in questo caso molto basilare e approssimativa ma comunque sufficiente a delineare quello che sarà il risultato finale, sono passato all'implementazione della logica che sta dietro l'applicazione vera e propria. Essa è strutturata principalmente in tre parti: la prima, quella di raccolta dati, è composta da diverse schermate che devono essere completate dall'utente, con i dati che poi andranno a costituire la valutazione del treno. La seconda parte è adibita alla visualizzazione dei dati,

come la ricerca di una voce, visualizzazione dello storico, della mappe o del profilo utente. L'ultima parte infine raccoglie le restanti funzioni offerte dall'applicativo, come la visualizzazione meteo o la raccolta dei dati dai sensori di movimento.

4.2 Funzionalità dell'applicazione TrainSpotter

In questa sezione elencherò e descriverò brevemente tutte le funzioni di maggior rilievo implementate nell'applicazione TrainSpotter.

Figura 4.4: Icona dell'applicazione TrainSpotter

4.2.1 Schermata iniziale

Quando l'utente lancia l'applicazione, dopo un breve periodo di caricamento, si troverà di fronte alla schermata iniziale, che offre due possibilità. La prima permette di accedere alla schermata di tutorial che spiega il funzionamento dell'applicazione, sia con un testo che con un breve video; la seconda permette di accedere alla schermata di check-in previa registrazione e successivo login. Se utente risulta già connesso è possibile disconnettersi. Viene inoltre eseguito un controllo sulla presenza di connettività internet sia WiFi che di rete cellulare, in modo da avvisare l'utente quando essa è assente.

Figura 4.5: Schermata iniziale

4.2.2 Login e Signup

Per potere utilizzare TrainSpotter è necessario essere in possesso di un account utente: è possibile utilizzare un account esistente e se non se ne è in possesso, se ne può creare uno al momento. Questa operazione è eseguibile direttamente senza lasciare l'applicazione, sono sufficienti solo username, password ed email. Una volta registrato un nuovo account, questo viene salvato nel server, dopoché sarà possibile fare il login tramite i dati selezionati.

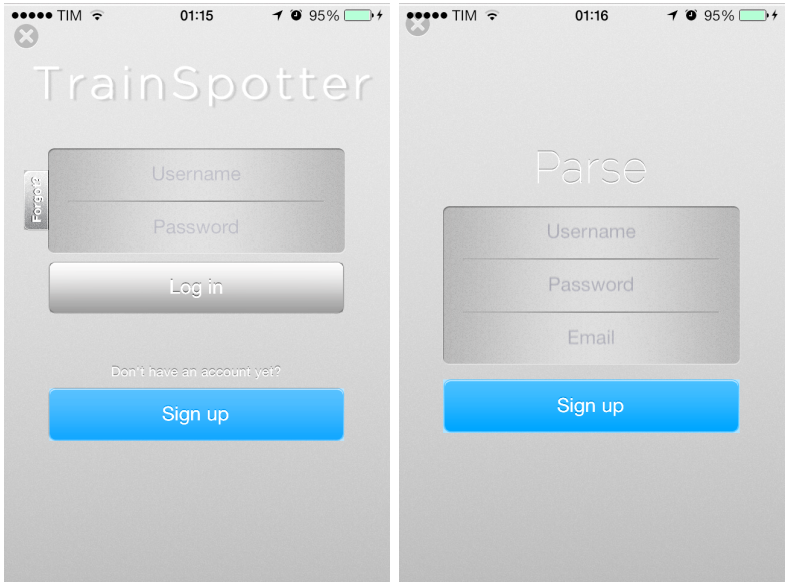


Figura 4.6: Schermate di login e registrazione utente

4.2.3 Check-In

La motivazione che sta dietro al meccanismo di check-in è quella di associare l'utente ad un specifico treno in circolazione nel momento in cui si sta viaggiando. Come dati di input sono necessari il numero del treno, la stazione di partenza e quella di destinazione: una volta inseriti è sufficiente selezionare l'opzione relativa al check-in e i dati immessi verranno inviati e salvati nel database risiedente nel server. Lato server verrà creata una nuova entry nel database contenente i dati inseriti dall'utente assieme ad un objectId univoco, il nome dell'utente che ha mandato la richiesta e la data/ora di creazione. La data e l'ora torneranno utili perché, supponendo che l'utente effettui il check-in quando effettivamente il treno parte, possiamo supporre che sia quella l'orario effettivo di partenza, in modo da calcolare poi la reale durata del viaggio.

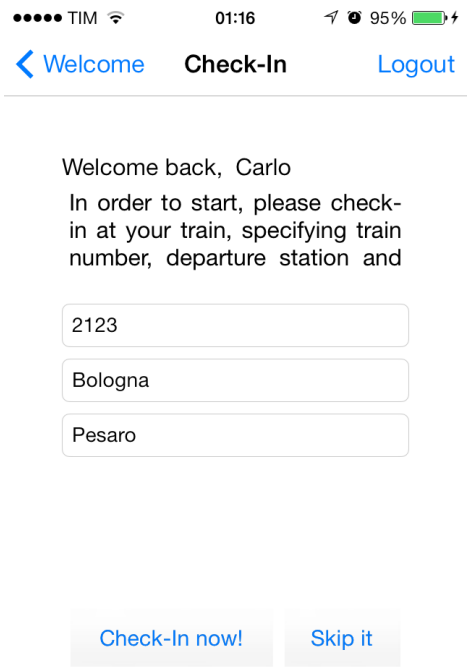


Figura 4.7: Schermata di Check-in

Da notare che l'operazione di check-in non è obbligatoria da eseguire; infatti è possibile saltare questo step e andare alla schermata successiva, nel caso l'utente voglia semplicemente consultare le valutazioni dei treni già presenti nel database. In alternativa è presente anche una opzione per effettuare il logout e tornare alla schermata principale.

4.2.4 Sezione Informazioni e Tutorial

In questa sezione è presente un breve testo dove viene spiegato il meccanismo di utilizzo dell'applicazione ed un breve video introduttivo.

4.2.5 Submit Opinion

Per fare una stima della valutazione della qualità del treno e del comfort di viaggio, è stato scelto un approccio di tipo collaborativo, basato come descritto in precedenza sul modello del crowdsourcing. Sulla scia di applicazioni come la sopracitata Waze, dove l'utente è il fautore della creazione e correzione delle mappe per la navigazione o come Foursquare dove l'utente si "registra" in un determinato posto e può lasciare foto, consigli o critiche, anche in questo caso è l'utente stesso, coinvolto in prima persona, a contribuire alla costruzione ed all'ampliamento del database dell'applicazione. Attraverso la voce di menu "submit opinion" l'utente potrà esprimere la sua valutazione su alcuni parametri definiti a priori.

In questo caso, a scopo dimostrativo sono stati presi in esame:

- pulizia: in che condizioni igieniche si trova la carrozza? sono presenti dei rifiuti? scritte sui finestrini?
- cattivi odori e temperatura: la carrozza emana cattivi odori? la temperatura è troppo alta o troppo bassa?
- affollamento: ci sono posti a sedere? il corridoio è libero? i collegamenti tra una carrozza e l'altra sono agibili?
- qualità del servizio: qui possono essere racchiusi una serie di parametri come l'accessibilità alla carrozza, le condizioni dei posti a sedere (sedili rotti o danneggiati) o la reperibilità a bordo del personale di servizio.

Per i quattro parametri appena elencati, è possibile esprimere un voto da 1 a 5, dove chiaramente 1 rappresenta il valore più basso (peggiore) e 5 quello più alto (migliore). Quindi per ogni domanda posta, l'utente dovrà selezionare nell'apposita sezione (nella parte inferiore della schermata) il valore della valutazione che vuole dare e selezionare "Next" (in altro a destra) per confermare tale valore e andare alla domanda successiva. Infine, nella parte superiore, una barra di scorrimento indica l'avanzamento del processo di valutazione.

Da notare che i parametri su cui effettuare la valutazione qui implementati, fungono solo da esempio: infatti è possibile con delle semplici modifiche aggiungere/togliere parametri o modificarne la forma.

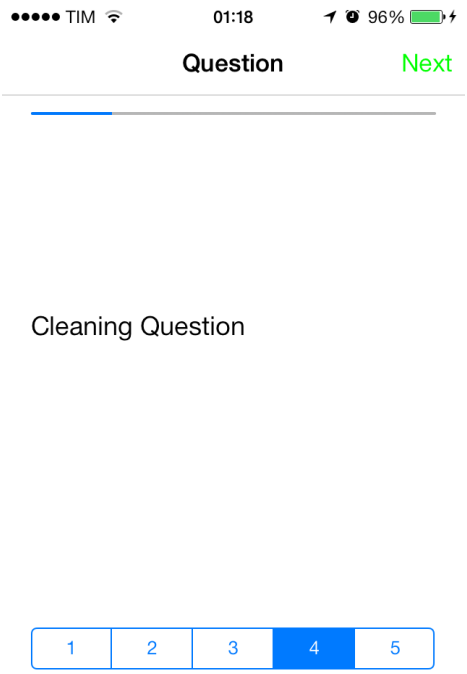


Figura 4.8: Schermate valutazione treno

La sezione successiva comprende una schermata dove l'utente può aggiungere liberamente a parole una sua opinione, consigli, critiche o comunicazioni. Ad esempio può risultare utile quando la valutazione dei parametri effettuata nel passo precedente non era completamente esaustiva e quindi si vuole specificare meglio, oppure semplicemente si può usare questo spazio per aggiungere commenti personali.

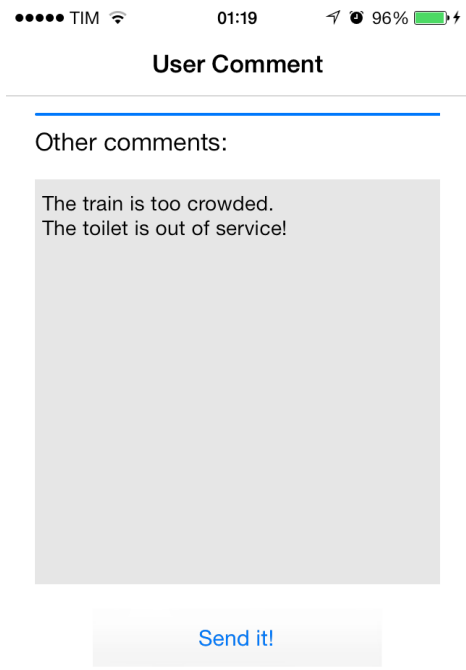


Figura 4.9: Schermata commenti utente

Dopo questa fase, si passa alla sezione audio. Ho deciso di implementare questa funzione perché, riflettendo su quali parametri ambientali possano influenzare la qualità di viaggio, ho pensato che un ulteriore parametro fosse il livello di rumorosità presente all'interno della carrozza. Per misurare la rumorosità mi sono servito del microfono incorporato nel dispositivo. Per utilizzare questa funzione l'utente non dovrà fare altro che registrare un breve campione audio che verrà analizzato in tempo reale, con visualizzazione del picco di decibel raggiunto. Una volta terminata la registrazione, verrà anche calcolata la media, sempre in decibel. Una volta registrato è possibile riascoltare il campione audio e/o inviare al server il valore medio, espresso in decibel, della rumorosità.

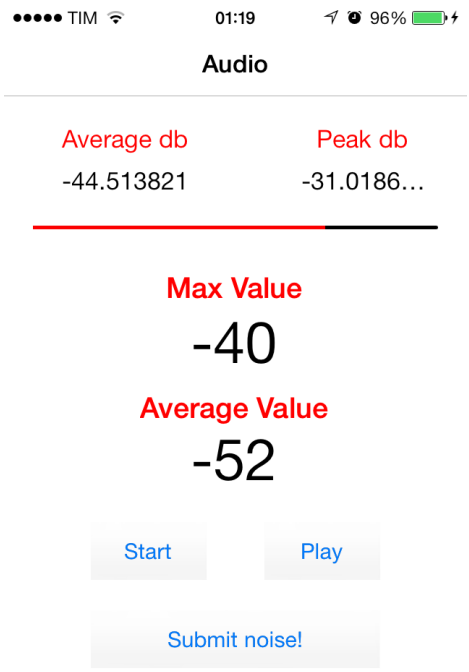


Figura 4.10: Schermata audio monitor

Infine l'ultima sezione riguarda le foto. Anche in questo caso, questa è una funzionalità di supporto alla valutazione dei parametri di qualità. In questa sezione l'utente dovrà prima scattare una foto con soggetto a proprio piacimento, come ad esempio un danno della carrozza o più semplicemente una panoramica che renda l'idea dell'affollamento. Una volta che si è soddisfatti della foto scattata, essa deve poi essere caricata sul server. Dato che normalmente una foto scattata da uno smartphone di ultima generazione può arrivare a pesare diversi megabyte, prima di essere caricata la foto viene compressa (ad una risoluzione impostabile a piacimento, al momento impostata a 640x480): ciò assicura lato utente un caricamento veloce e lato server poco spazio di memorizzazione utilizzato.

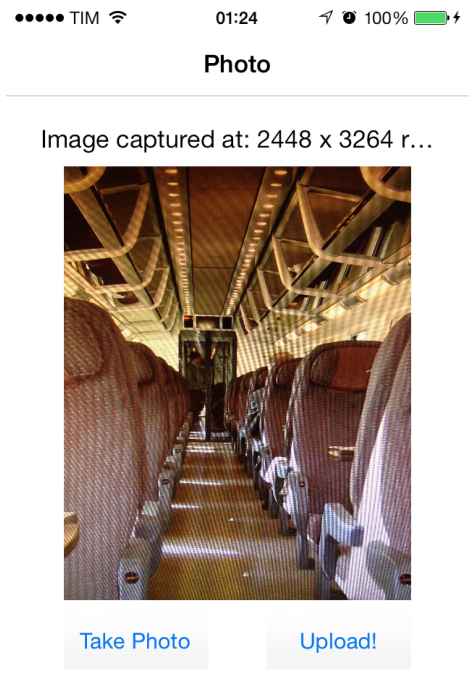


Figura 4.11: Schermata cattura foto

Completate queste fasi il tutto viene salvato sul server, nella stessa entry del database dove sono già presenti i valori relativi al check-in associati all'utente corrente. In questo modo è possibile ottenere una misurazione obiettiva, veloce e con un piccolo sforzo da parte dell'utente. Terminata questa fase si ritorna alla schermata principale degli strumenti.

4.2.6 Motion Data

Per quando riguarda la funzionalità di interpretazione del movimento, sono stati presi in esame i sensori di movimento, ovvero accelerometro e giroscopio, e anche il magnetometro. Assieme al Prof. Ghini abbiamo ipotizzato che osservando questi dati, sia possibile stabilire la posizione corrente nello spazio dell'utilizzatore, se ad esempio si trovi in piedi o a sedere.

L'accelerometro opera su tre assi (x, y, z) e misura l'accelerazione lineare del dispositivo, registrando roll e pitch, e se combinato con il magnetometro riesce a misurare anche l'orientamento, senza però riuscire a misurare le rotazioni. Qui entra perciò in gioco il giroscopio che nel caso in cui il dispositivo registri una rotazione attorno ad un asse, permette di misurare il movimento su un sistema inerziale a sei gradi di libertà, permettendo di usare la dead reckoning[1] (navigazione stimata) per ricavare la locazione fisica e l'orientamento del dispositivo relativa ad una posizione iniziale. Infine il magnetometro può essere visto come una sorta di bussola digitale: combinando i suoi dati con quelli relativi all'heading ricavato dal giroscopio e il roll e il pitch ricavato dall'accelerometro, restituisce la reale orientazione del telefono.

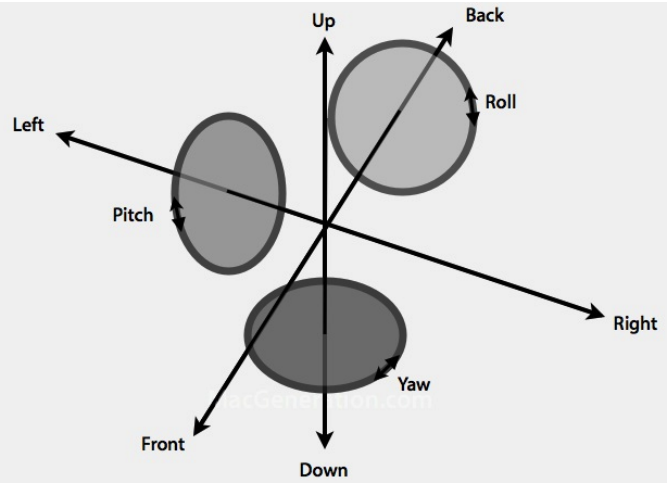


Figura 4.12: Grafico dell'accelerazione lineare ed angolare catturata da accelerometro e giroscopio

Nello specifico, sono state implementate due varianti di acquisizione dei dati di movimento: la prima interessa solamente accelerometro e giroscopio da cui vengono acquisiti direttamente i dati grezzi, relativi allo spostamen-

[1]Metodologia di previsione della determinazione del punto stimato tramite la conoscenza di elementi del moto e della loro variazione (come velocità e direzione).

to del dispositivo relativamente ai 3 assi considerati (x, y, z). La seconda variante invece, effettua una prima elaborazione dei dati grezzi acquisiti dai sensori e provvede a calcolare i valori di attitude, rotation rate, gravity, user acceleration, magnetic field.

Per il momento le funzionalità implementate si limitano a mostrare a video i dati catturati dai sensori e, opzionalmente, salvarli. Questi dati dovranno poi essere utilizzati per eseguire una analisi tramite l'utilizzo di algoritmi specifici di interpretazione del movimento.

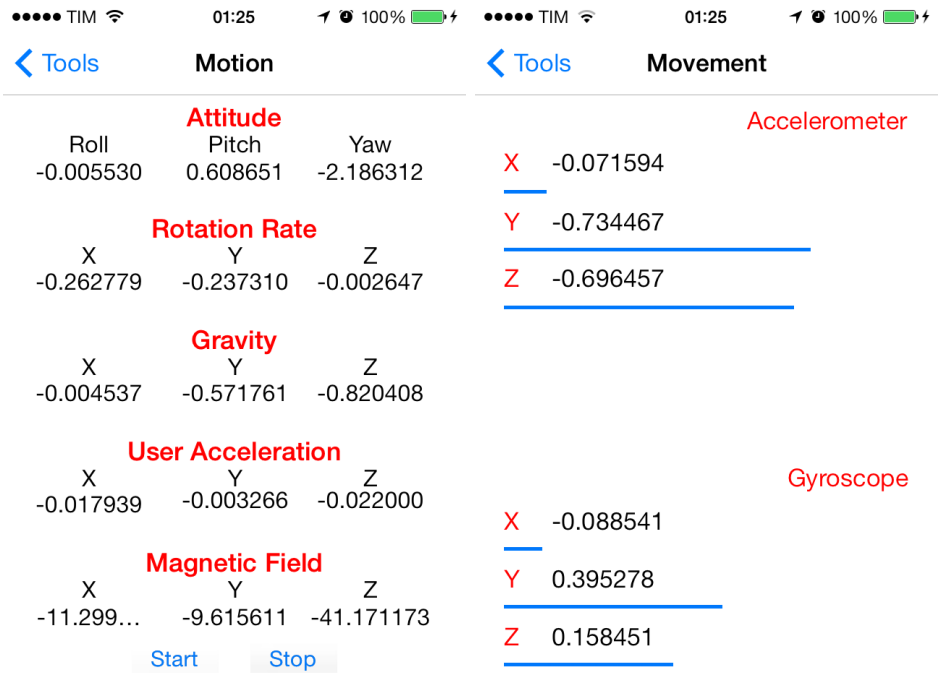


Figura 4.13: Schermata cattura dati di movimento

Degno di nota il fatto che sull'ultimo modello di iPhone (5S) sia stata aggiunta un'unità di computazione complementare chiamata M7, dedicata esclusivamente al collezionamento di dati proveniente dai sensori di movimento e orientamento. Essa ha il vantaggio di funzionare anche quando il dispositivo è in standby riducendo notevolmente il consumo di batteria e sen-

za dover usare costantemente la CPU principale poiché i dati provenienti dai sensori sono passati direttamente all'unità computazionale M7. Permetterà quindi alle applicazioni di riconoscere il tipo di movimento che l'utente sta compiendo ad esempio come guidare, camminare, correre o dormire. Altre applicazioni potrebbero interessare il mapping interno degli edifici.

4.2.7 Search Train

Questa sezione è la prima di cinque che vanno a comporre la TabBarController. Essa permette all'utente di eseguire una ricerca, tramite il numero identificativo del treno, di tutte le segnalazioni effettuate fino a quel momento da tutti gli utilizzatori di TrainSpotter. Per utilizzare questa funzione è sufficiente inserire il numero del treno e poi selezionare cerca: verrà quindi inviata una richiesta al server che restituirà la lista delle segnalazioni, ordinate cronologicamente.

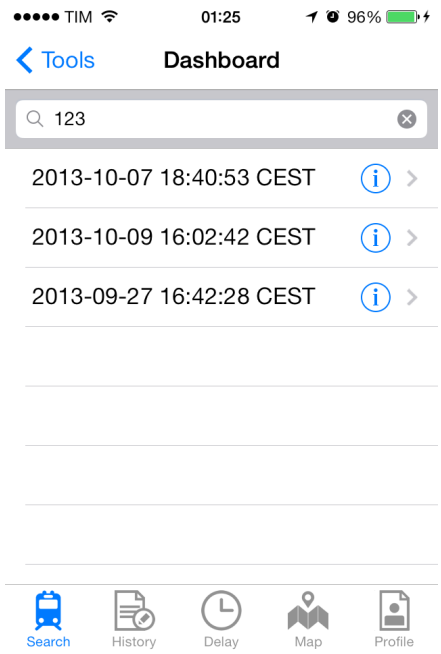


Figura 4.14: Schermata ricerca treno

Selezionando l'elemento di interesse è possibile vedere i dettagli di quella segnalazione, contenente tutte le voci della sezione relative alla valutazione del treno.

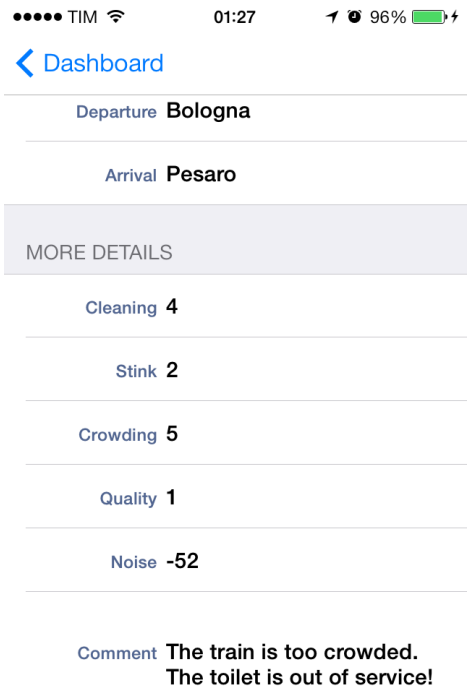


Figura 4.15: Schermata dettaglio valutazione treno

In questo modo l'utente che si trova nella fase di scelta del treno da prendere può velocemente controllare la valutazione che il treno che è intenzionato a prendere possiede, in modo poi da indirizzare al meglio la sua scelta.

4.2.8 History

In questa sezione, il server è automaticamente interrogato per restituire la lista delle segnalazioni che l'utente attualmente collegato ha effettuato. Ogni elemento della lista è composto da: numero identificativo del treno, stazione di partenza e stazione di arrivo. Analogamente a quanto succede nella sezione

ricerca, anche in questo caso quando l'utente seleziona un elemento dalla vista, si passa alla visuale del dettaglio segnalazione contenente tutti i dati.

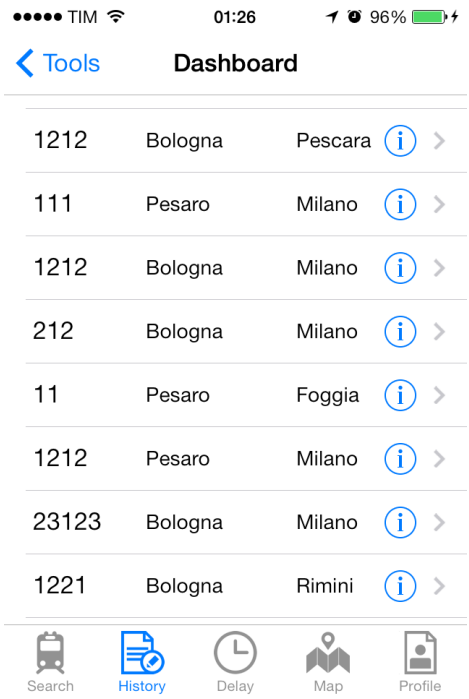


Figura 4.16: Schermata cronologia segnalazioni

Come funzionalità aggiuntiva, l'utente ha la possibilità di eliminare una segnalazione effettuata, di aggiungerne una nuova (che potrebbe riferirsi ad un viaggio effettuato in precedenza) o di modificarla.

4.2.9 Delay

Questa sezione è adibita alla visualizzazione dei dati utili di viaggio come tempo trascorso, distanza percorsa e ritardo rispetto la tabella di marcia. Per il tempo trascorso eseguo la differenza tra l'orario corrente e quello relativo alla creazione della classe CheckIn all'interno del database (questo perché si suppone che l'utente effettui l'operazione di check-in non appena inizi il suo

viaggio). Per quanto riguarda la distanza percorsa invece, eseguo il calcolo considerando la distanza tra due punti sulla mappa, espressi in coordinate geografiche (latitudine e longitudine). Il primo è quello che viene salvato in fase di esecuzione del check-in, mentre il secondo è quello in cui l'utente si trova correntemente.

Per il calcolo dei ritardi la situazione si un po' più complicata. Come prima cosa viene interrogato il servizio http://mobile.viaggiatreno.it[2] utilizzando il numero del treno fornito in fase di check-in e vengono recuperati, parsati e memorizzati i dati ufficiali trenitalia, in particolare il ritardo del treno in questione, aggiornato all'ultima stazione superata. Dall'altro lato viene mostrato invece la differenza tra il tempo effettivamente trascorso e il tempo che il treno avrebbe dovuto impiegare per raggiungere quella determinata stazione, in modo da ottenere l'effettivo ritardo, che non sempre è quello comunicato da Trenitalia.

4.2.10 Map View

Questa sezione è costituita da una mappa configurata con visualizzazione standard (ovvero quella stradale) su cui viene visualizzata la posizione corrente dell'utente e le stazioni dei treni di tutte le fermate relative al treno su cui si è fatto il check-in, in modo da focalizzare l'attenzione su quella determinata zona e di creare un percorso visivo di facile lettura e interpretazione.

E' stato implementato anche un sistema che avvisa l'utente quando si è in prossimità di una stazione, in modo che ci si possa preparare per tempo alla discesa. Per prima cosa tramite l'opzione "set reminder" l'utente inserisce il nome della città di interesse. Confermata la selezione, viene tracciato un perimetro circolare avente come centro le coordinate della stazione impostata, perimetro che ha un certo raggio di copertura (espresso in radianti) decidibile a piacimento. Quindi, una volta avviato il sistema di localizzazione, nel momento in cui le coordinate della posizione corrente ricadano all'interno del

[2]Servizio web di Trenitalia che informa sull'andamento del traffico ferroviario nazionale e regionale.

perimetro impostato, ciò causerà la scatenazione di un evento che consisterà in un avviso di notifica all'utente.

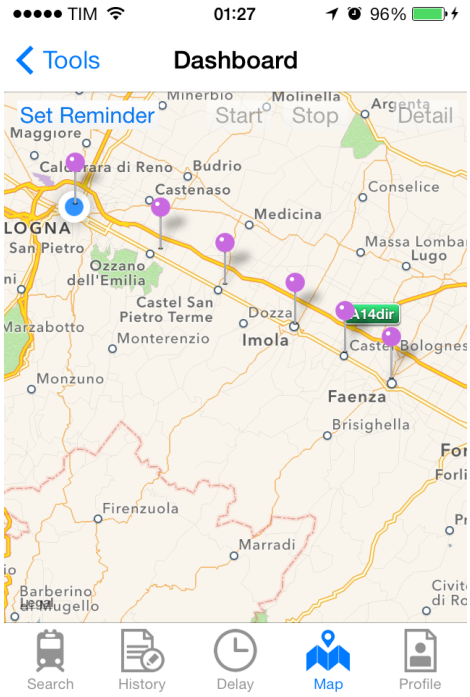


Figura 4.17: Schermata vista mappa

Inoltre sempre quando i servizi di localizzazione sono avviati, possono essere visualizzare alcune informazioni come le coordinate in cui ci si trova, la velocità, l'orario.

4.2.11 User Profile

Questa sezione, la quinta ed ultima facente parte della parte TabBarController, è dedicata alla visualizzazione delle informazioni relative all'utente collegato. Tra queste abbiamo l'immagine profilo, l'username, il totale delle segnalazioni effettuate ed i riconoscimenti ottenuti nell'utilizzo dell'applicazione.

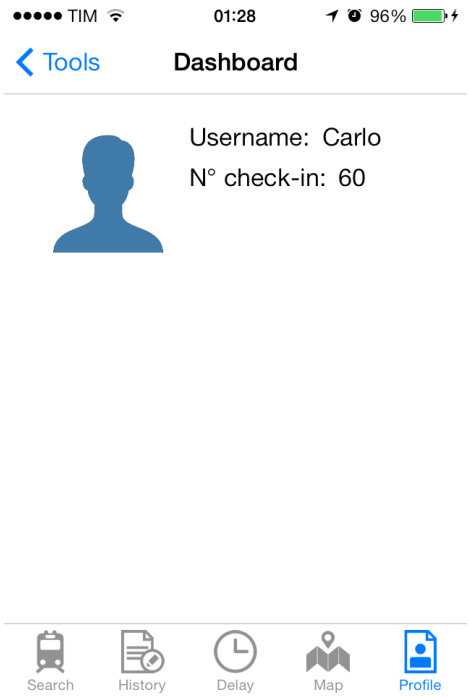


Figura 4.18: Schermata profilo utente

Capitolo 5

Valutazione

Le applicazioni mobile, poiché eseguite su dispositivi con una disponibilità minore di risorse di sistema rispetto alla loro controparte desktop, richiedono una maggiore attenzione per quanto riguarda la fase di testing. Questo fattore spesso consiste in una durata superiore in termini di tempo alla fase di analisi dei requisiti e sviluppo software, per evitare il presentarsi del rischio di perdita di competitività sul mercato, poiché si offre un prodotto che presenta problematiche. Oltre ai soliti requisiti funzionali e requisiti dell'interfaccia grafica, rispetto alle standard applicazioni desktop dobbiamo considerare fattori aggiuntivi come dimensione dello schermo, piattaforma di sviluppo, calcolo computazionale, durata della batteria.

La fase di testing delle applicazioni mobile è un processo tramite il quale il software sviluppato è testato con riguardo alle sue funzionalità, usabilità e consistenza. E' possibile effettuare differenti tipologie di testing che vanno a coprire diversi aspetti:

- **functional testing**: ci si assicura che l'applicazione funzioni secondo le specifiche dei requisiti funzionali. In questa fase sono previsti test sull'interfaccia utente ed esecuzione delle funzionalità;
- **laboratory testing**: eseguito per scoprire eventuali problematiche quando viene usato la connessione voce o dati per l'esecuzione di certe funzioni;

- **performance testing**: eseguito per misurare le performance e il comportamento dell'applicazione sotto certe condizioni come possono essere scarsa autonomia residua, scarsa copertura di rete, scarsa memoria disponibile, utilizzo della CPU, accesso simultaneo al server da parte di molti utenti. Le performance dell'applicazione possono essere influenzate sia lato server che client e questo test le prende in considerazione entrambe;

- **memory leakage testing**: molto importante data la scarsa quantità di memoria disponibile sui dispositivi mobili, serve ad individuare malfunzionamenti nel sistema di gestione della memoria;

- **interrupt testing**: eseguito per verificare se durante la sua esecuzione l'applicazione continua a funzionare correttamente nonostante il verificarsi di eventi quali la ricezione di chiamate o SMS, di notifiche, disconnessione dalla rete;

- **usability testing**: importante perché strettamente legato al successo commerciale, viene eseguito per verificare se l'applicazione raggiunge gli obiettivi preposti con una risposta positiva degli utenti, con efficienza e accuratezza;

- **user interface testing**: vengono controllati posizione, dimensione, colori degli elementi grafici componenti l'interfaccia, se i messaggi di errore vengono visualizzati correttamente, se il testo e le immagini sono leggibili, se sono presenti campi per input ed output di dati;

- **installation testing**: viene verificato se il processo di installazione, aggiornamento e disinstallazione viene portato a termine senza difficoltà per l'utente;

- **security testing**: deve assicurare l'autenticazione dell'utente, autorizzazione, protezione dei dati.

5.1 Criteri di valutazione dell'applicazione

Dopo la realizzazione di un primo prototipo dell'applicazione è possibile passare ad una prima fase di test e valutazione di ciò che è stato realizzato. Ma prima di valutare l'operato, bisogna definire dei criteri, dei parametri di valutazione su cui basarsi. Ad esempio:

- l'interfaccia utente è facile da utilizzare?
- l'applicazione soddisfa un qualche bisogno o necessità?
- sono presenti alcune funzionalità nascoste o non facilmente raggiungibili e utilizzabili?
- c'è qualche funzionalità che l'utente vorrebbe avere ma che non è attualmente disponibile?

Mentre i passi da seguire per eseguire una valutazione, possono essere individuati in:

1. indentificazione del target di utenza
2. indentificazione dei bisogni dell'utenza
3. costruzione di un primo prototipo
4. valutazione del prototipo
5. modifica del prototipo
6. riesecuzione della valutazione ripetendo i passi 5 e 6 finché è necessario
7. rilascio di una beta
8. rilascio definitivo

E' importante scegliere degli utenti il più possibile vicini al target di utenza e identificabili chiaramente per differenze nelle abilità e conoscenze che essi

possiedono; chiedere il consenso per includerli nello studio, rendere chiaro ciò che devono fare e per quanto tempo, rendere i risultati anonimi.

Negli stadi iniziali dello sviluppo è tipicamente più conveniente coinvolgere un piccolo numero di utenti così da valutare l'applicativo in modo approfondito e fornire un feedback dettagliato. Mentre quando si è giunti in fase di beta testing è consigliabile impiegare un largo numero di utenti che sebbene possano fornire dei feedback superficiali, forniscono anche delle buone statistiche sull'effettivo utilizzo del sistema.

Per quanto concerne invece la valutazione del design, alcuni dei criteri possono essere:

- l'interfaccia grafica è usabile? le funzionalità sono chiare e facilmente utilizzabili? manca qualcosa di fondamentale?
- le funzionalità implementate sono perfettamente funzionanti oppure soffrono di qualche bug o crash?
- il contesto di utilizzo dell'applicazione è appropriato?
- l'utente è motivato o incoraggiato nell'utilizzo dell'applicazione?
- ci sono dei possibili miglioramenti per l'usabilità, la stabilità e l'utilità?

Mentre dei possibili metodi di valutazione del design sono:

- questionario: chiedere all'utente opinioni, miglioramenti. E' facile da analizzare ma l'utente potrebbe mentire o comunque non essere in grado di rispondere a domande troppo tecniche;
- osservazione: osservare l'utente mentre usa il sistema, invitarlo a parlare ad alta voce mentre esegue operazioni. Si vede in prima persona quello che veramente succede, ma l'osservatore potrebbe rendere l'utente nervoso o in soggezione;
- interviste di gruppo: formare dei piccoli gruppi di persone che discutono dell'applicativo (dopo averlo già utilizzato). Potrebbero emergere

delle osservazioni interessanti ma c'è il rischio che il gruppo non sia abbastanza rappresentativo del target o che l'intervistatore possa forzare i risultati;

- diario: viene chiesto all'utente di prendere nota scritta delle proprie osservazioni. E' un metodo poco intrusivo ma il documento dovrà risultare compatto, strutturato e di facile comprensione altrimenti non potrà essere utilizzato.

5.2 Resoconto della fase di testing

Come spiegato nella sezione precedente, la fase di testing ricopre un ruolo fondamentale per la buona riuscita di una applicazione mobile. L'ambiente di sviluppo integrato utilizzato per lo svolgimento di questo progetto di tesi, Xcode, offre due possibili modalità di testing: la prima tramite il simulatore iOS, ovvero una virtualizzazione del device che ne simula le caratteristiche e comportamento anche se con delle limitazioni rispetto alla versione fisica (ad esempio assenza di fotocamera e sensori di movimento). La seconda modalità consiste nel deploy dell'applicazione sul proprio dispositivo dopo averlo correttamente configurato come device adibito allo sviluppo. Con l'acquisto della licenza di sviluppatore personale, è possibile impostare fino ad un massimo di 100 dispositivi adibiti per il testing ma logicamente non potendo disporre di una quantità così elevata di dispositivi, ho utilizzato solo quello in mio possesso e quelli di alcuni colleghi.

Per quanto riguarda il testing relativo a tutto ciò che comprende l'interfaccia grafica, oltre ad una analisi personale ho provveduto ad inviare delle versioni di test di TrainSpotter ad una cerchia ristretta di utenti. Ad essi ho chiesto specificatamente di provare ad utilizzare le funzionalità offerte, di segnalare eventuali anomalie del design dell'interfaccia e del posizionamento degli elementi e di segnalare tutto ciò che non era chiaro durante l'utilizzo. Grazie ai feedback ricevuti, sono stato in grado di migliorare l'usabilità di

TrainSpotter, intervenendo su quelle parti che a me risultavano già chiare ma, per utenti meno esperti, non lo erano affatto.

Invece per quanto riguarda le funzionalità relative a mappe e posizionamento GPS, poiché era infattibile eseguire test "sul campo" sia per una questione di tempo che di costi, sono stato forzato ad utilizzare il simulatore di iOS integrato dalla suite di Xcode. Tra le tante possibilità che esso offre, vi è anche quella di simulare una locazione personalizzata attraverso la specifica di coordinate GPS (latitudine e longitudine). Come primo passo ho creato un file di tipo GPS eXchange Format (GPX[1]), una tipologia particolare di file XML usato per descrivere waypoint e tracciati. All'interno di questo file ho poi inserito le coordinate di latitudine e longitudine relative alle stazioni dei treni che volevo rappresentare, in modo tale da simulare il percorso che segue il treno nella realtà. Tramite questo metodo sono stato in grado di testare sia la parte relativa alle informazioni di viaggio (velocità, posizionamento, direzione) e sia quella relativa alla funzione del reminder.

Infine per quando riguarda la parte incentrata sull'acquisizione di dati relativi ai sensori di movimento, l'operazione di testing si fa più complicata. Con i dati che sono riuscito a raccogliere, sia quelli grezzi provenienti da accelerometro e giroscopio e sia quelli sottoposti ad una prima elaborazione, sono solamente riuscito a verificare che effettivamente essi rispecchiano i movimenti effettuati dal dispositivo. Ma dare una interpretazione più dettagliata in modo da ricavare la tipologia di movimento effettuata dall'utente, richiede numerose operazioni e studio di algoritmi di interpretazione che non rientrano all'interno di questo lavoro di tesi.

[1]GPS Exchange Format: http://www.topografix.com/gpx.asp

Conclusioni

I dispositivi mobili di ultima generazione costituiscono un sistema di integrazione tra hardware e software tecnologicamente evoluto ed elaborato, presentando le tipiche caratteristiche dei sistemi desktop ma aggiungendo funzionalità interessanti quali l'integrazione di sensori hardware di acquisizione dati e la possibilità di operare in mobilità. Con la crescita della domanda di applicazioni mobile, sta aumentando di conseguenza anche la domanda di sviluppatori di applicazioni mobile.

L'obiettivo del mio progetto di tesi era quello di sviluppare una applicazione mobile collaborativa per il monitoraggio e l'analisi dei viaggi ferroviari, adibita alla raccolta di dati sia tramite il contributo dell'utente e sia tramite i sensori integrati nel dispositivo, rendendo poi i dati raccolti disponibili alla consultazione da parte di altri utenti del sistema o per una profilazione di analisi e statistiche per ulteriori studi.

Ho aperto il documento di tesi con una panoramica dello scenario su cui sono andato ad operare, cioè quello della mobilità sui treni, e sulla situazione tecnologica attuale, soffermandomi prima sulla presentazione delle maggiori piattaforme di sviluppo di sistemi mobile e poi su una descrizione delle tipologie di servizi cloud più comuni. Ho cercato quindi di svolgere un lavoro di analisi delle tecnologie e piattaforme esistenti in modo da capire quale fosse quella più adatta alle mie esigenze. In seguito, nel secondo capitolo, ho descritto gli strumenti di cui ho fatto uso nello sviluppo di questo progetto, partendo da quelli fisici quali il dispositivo mobile (iPhone) e l'utente, per passare a quelli software ovvero il sistema operativo mobile, l'ambiente

di sviluppo integrato, il linguaggio di programmazione e il backend. Successivamente nel terzo capitolo sono passato alla descrizione della fase di progettazione in cui ho specificato gli obietti da raggiungere, le scelte progettuali, i requisiti funzionali e non funzionali da soddisfare e l'approccio architetturale allo sviluppo dell'applicazione. Dopo questa fase, nel quarto capitolo sono passato all' implementazione di quanto specificato in precedenza e ho descritto (documentando con screenshots) tutte le principali funzioni che l'applicazione realizzata è in grado di offrire. Si conclude infine, con il quinto capitolo, la sezione relativa ai criteri di valutazione dell'applicazione e ai test effettuati.

Le problematiche prese in esame, anche se nel caso specifico relative al trasporto ferroviario, sono quelle tipiche dei trasporti pubblici: ritardi, sovraffollamento, scarsa qualità del servizio erogato. Con questo progetto ho cercato di trovare una possibile soluzione.

Per quanto riguarda la parte relativa al client, la piattaforma mobile creata da Apple si è dimostrata adeguata allo scopo di questo lavoro, offrendo numerosi strumenti a supporto dello sviluppo software, API molto potenti e ottima documentazione. Dall'altro lato, l'utilizzo del servizio cloud mobile Backend as a Service ha consentito di snellire notevolmente il processo di sviluppo fornendo "out of the box" tutto quello che era necessario, permettendo quindi di impiegare il tempo destinato alla progettazione del backend per la realizzazione del front-end, consentendo la realizzazione di applicazioni di elevata qualità rimanendo comunque svincolati dalla piattaforma che si è scelto di utilizzare, poiché compatibli con le piattaforme mobile e desktop di maggior rilievo presenti sul mercato. I mBaaS possono essere considerati come la risposta naturale allo spostamento dello sviluppo software verso il cloud, offrendo le stesse performance ed assicurando scalabilità. Quindi mentre prima il processo di sviluppo era molto più complesso e costoso, con questo procedimento chiunque può sviluppare a costo (quasi) ridotto zero applicazioni elaborate e renderle disponibili in pochi passi ad un bacino d'utenza potenzialmente mondiale.

TrainSpotter è incentrata sui viaggi ferroviari perché è uno scenario che mi coinvolge in prima persona, essendo un pendolare da diversi anni. Ma ritengo importante sottolineare che la struttura dell'applicazione è facilmente adattabile ad altri contesti: autobus, metro, traghetti. Ogni mezzo di trasporto pubblico soffre di problemi più o meno gravi ed utilizzando uno strumento di questo tipo è ipoteticamente possibile risolverli. Questa applicazione segue inoltre un trend che negli ultimi tempi ha registrato una notevole diffusione, quello della valutazione di luoghi e servizi. Basti pensare a Foursquare[2], TripAdvisor[3], Yelp[4], solo per citare le più famose.

Un'altra applicazione di TrainSpotter potrebbe essere la seguente. Il personale di bordo in servizio durante il viaggio in treno, poiché possiede già un dispositivo mobile per la comunicazione e il controllo biglietti, potrebbe essere abilitato anche alla lettura e revisione in tempo reale delle segnalazioni che vengono effettuate dagli utenti a bordo, in modo da intervenire tempestivamente per i problemi più gravi o a fine corsa per quelli minori.

Sviluppi futuri

La vastità del progetto a cui ho lavorato, lascia spazio a numerosi perfezionamenti e aggiunta di nuove funzionalità. Alcuni tra i possibili miglioramenti che ritengo possano contribuire al proseguimento dello sviluppo di TrainSpotter sono:

- possibilità di registrare un nuovo account utente tramite le credenziali del proprio account Facebook o Twitter, in modo da semplificare il processo evitando di fare inserire manualmente i dati personali all'utente e aggiungendo la possibilità di condivisione delle segnalazioni sui social network;

[2]Foursquare Labs, Inc.: http://foursquare.com/
[3]TripAdvisor, Inc.: http://www.tripadvisor.com/
[4]Yelp, Inc.: http://www.yelp.com/

- portare a termine l'implementazione dell'integrazione del servizio fornito da Trenitalia, Viaggiatreno[5], un servizio fornito da Trenitalia che fornisce informazioni su orari dei treni, stazioni, binari di partenza e altro. Purtroppo non sono disponibili delle API liberamente utilizzabili dagli utenti, quindi è necessario implementare un metodo personalizzato. I dati ritornati dal servizio viaggiatreno potrebbero essere utilizzati nella fase di check-in, in modo da ricavere le stazioni di partenza e arrivi automaticamente dal numero del treno, o per confrontare i valori di ritardo forniti ufficialmente con quelli registrati dagli utenti, ed ad esempio calcolare il ritardo reale del treno tramite la differenza di orario di partenza e quella prestabilita. Discorso analogo per l'arrivo;

- aggiungere il supporto al co-processore M7: esso prevede la registrazione dei dati provenienti dai sensori di movimento grazie ad un processo sempre attivo in background, memorizzandoli sul chip per un periodo massimo di 7 giorni. Quindi non c'è più bisogno di lanciare l'applicazione ed iniziare il monitoraggio. Qualsiasi applicazione che abbia accesso al chip M7 può analizzare i dati e individuare il tipo di movimento. Poiché ormai abbiamo abitudilmente il telefono sempre con noi, sarà possibile ottenere una interpretazione realistica del movimento;

- feedback e rewards: per il corretto funzionamento della applicazione è fondamentale il contributo degli utenti. Anche se i passi da completare per la valutazione dei treni sono molto semplici e veloci, è consigliabile implementare un meccanismo che incentivi l'utente a dare il suo contributo, che gli permetta di confrontarsi con gli altri e in un certo modo di competere, in modo che lo spinga ad essere il migliore tra la sua cerchia di contatti. L'esempio classico sono i badge, cioè l'attribuzione di un premio simbolico all'utente ogniqualvolta vengano raggiunti determinati traguardi (ad esempio un certo numero di segnalazioni). Per

[5]Viaggiateno.it: servizio fornito da trenitalia che fornisce informazioni su orari dei treni, stazioni, binari di partenza e altro.

quanto riguarda il feedback, si potrebbe implementare la possibilità di votare le segnalazioni, in modo da attribuire credibilità all'utente.

Bibliografia

[1] Index, Cisco Visual Networking, "Global Mobile Data Traffic Forecast Update, 2012-2017,"

[2] Italiana, RFI Rete Ferroviaria, "La rete oggi,"

[3] Emilia-Romagna, Regione, "Mobilità regione Emilia-Romagna,"

[4] Yashpalsinh Jadeja, Kirit Modi, "Cloud Computing - Concepts, Architecture and Challenges," *International Conference on Computing Electronics and Electrical Technologies ICCEET*, 2012

[5] Liu, Wentao, "Research on Cloud Computing Security Problem and Strategy,"

[6] Farhan Bashir Shaikh, Sajjad Haider, "Security Threats in Cloud Computing," *6th International Conference on Internet Technology and Secured Transactions*, 2011

[7] Michael Andres Feliu Gutierrez, Neco Ventura, "Mobile Cloud Computing based on service oriented architecture: embracing network as a service for 3rd party application service providers,"

[8] Mann, Stephanie, "Backend as a Service points to mobile development futures,"

[9] Alexander Lenk, Michael Menzel, Johannes Lipsky Stefan Tai Philipp Offermann, "What are you paying for? Performance benchmarking

for Infrastructure-as-a-Service offerings," *International Conference on Cloud Computing*, 2011

[10] Rahul Ghosh, Kishor S. Trivedi, Vijay K. Naik and Kim, Dong Seong, "End-to-End Performability Analysis for Infrastructure-as-a-Service Cloud: An Interacting Stochastic Models Approach," *Pacific Rim International Symposium on Dependable Computing*, 2010

[11] Michael Boniface, Bassem Nasser, Juri Papay Stephen C. Phillips Arturo Servin Xiaoyu Yang Zlatko Zlatev Spyridon V. Gogouvitis Gregory Katsaros Kleopatra Konstanteli George Kousiouris Andreas Menychtas Dimosthenis Kyriazis, "Platform-as-a-Service Architecture for Real-time Quality of Service Management in Clouds," *Fifth International Conference on Internet and Web Applications and Services*, 2010

[12] Yongqing Zheng, Jinshan Pang, Jian Li Lizhen Cui, "Business Process Oriented Platform-as-a-Service Framework for Process Instances Intensive Applications," , 2012

[13] Keiko Hashizume, Eduardo B. Fernandez, Maria M. Larrondo-Petrie, "A pattern for Software-as-a-Service in Clouds," *ASEIEEE International Conference on BioMedical Computing*, 2012

[14] "Backend as a Service (BaaS) Market worth 7.7 Billion by 2017,"

[15] Costin Raiciu, Christoph Paasch, Sebastien Barre Alan Ford-Michio Honda Fabien Duchene Olivier Bonaventure and Handley, Mark, "How Hard Can It Be? Designing and Implementing a Deployable Multipath TCP," *Proceeding NSDI12 Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation*, 2012