Università degli studi di Modena e Reggio Emilia

Facoltà di Ingegneria

Corso di Laurea Magistrale in

Ingegneria Informatica

Progetto e sviluppo di un'applicazione mobile per il calcolo dei percorsi

Relatore:	Correlatore:	Candidato:
Prof. Sonia Bergamaschi	Prof. Riccardo Martoglia	Daniele Cristofori

Anno Accademico 2012/2013

Alla mia famiglia

e a tutti quelli che mi sono stati accanto

Indice

Elenco delle figure

Introduzione

Le modalità di spostamento in città sono cambiate nel corso degli ultimi anni. In passato le persone si muovevano a piedi, in bicicletta, con tram e autobus, perché pochi avevano l'automobile di proprietà. Oggi, con la crescita delle città, gli spostamenti dalla periferia verso il centro, l'accresciuto potere di acquisto delle persone e lo stile di vita basato sul confort, il numero di spostamenti effettuato con veicoli privati sta aumentando in maniera significativa. Questa evoluzione della mobilità urbana ha in molti casi come risultato il peggioramento delle condizioni del traffico, con un conseguente incremento di ingorghi ed aumento delle emissioni inquinanti nelle aree urbane.

I sistemi di trasporto sostenibile danno un contributo significativo alla sostenibilità ambientale, sociale ed economica delle comunità che servono. I requisiti che dovrebbero essere soddisfatti sono almeno tre: potersi muovere con tempi e costi ragionevoli; avere impatti limitati sull'ambiente urbano e sulla qualità dell'aria ed evitare i danni alla salute; limitare i consumi di fonti energetiche non rinnovabili.

Lo scopo di questa tesi e dell'applicazione, "Route Planner", che è stata sviluppata è quello di incentivare un Piano di Mobilità Urbana Sostenibile cioè una serie di azioni che hanno come obiettivo l'introduzione di forme di spostamento più sostenibili, quali camminare, andare in bicicletta, utilizzare mezzi pubblici all'interno della città o utilizzare mezzi di trasporto privato condivisi (bike sharing e car rental) ovvero modalità di trasporto che rendono compatibile la crescita economica, la coesione sociale e la protezione dell'ambiente, assicurando al contempo una migliore qualità della vita per i cittadini.

Il problema principale affrontato è stato quello di proporre dei percorsi alternativi nel raggiungere una destinazione attraverso l'utilizzo di una combinazione di mezzi di trasporto (auto dei car rental, biciclette pubbliche, mezzi pubblici e a piedi). Tale problema è stato risolto attraverso la progettazione e l'implementazione

di un algoritmo ad-hoc che utilizzasse gli Open Data che sono stati trovati per il comune di Bologna e i dati già posseduti dal servizio Google Maps.

In questa tesi sono stati affrontati diversi argomenti che sono stati necessari nella progettazione e nell'implementazione dell'app "Route Planner": è stato studiato ed approfondito il framework di riferimento Titanium; è stata affrontata l'analisi degli Open Data per il comune di Bologna e, più in generale, della regione Emilia-Romagna; è stato ideato un algoritmo ad-hoc per il calcolo di tutti i possibili percorsi tra due punti tramite l'utilizzo di una combinazione di mezzi di trasporto; sono state studiate tutte le API messe a disposizione da Google Maps necessarie all'implementazione dell'algoritmo ideato; sono state acquisite conoscenze utili per lo sviluppo di interfacce di applicazioni per iPhone.

La struttura della tesi è la seguente:

- Il primo capitolo è dedicato al framework Titanium che è stato adottato nello sviluppo dell'applicazione "Route Planner": viene riportata una sua descrizione, i vantaggi e gli svantaggi del suo utilizzo e ne viene fatto un paragone con altri framework Cross Platform;
- Nel secondo capitolo viene affrontata la tematica degli Open Data: vengono riportate una descrizione e i vantaggi che potrebbero derivare da un loro utilizzo intelligente. Infine vengono analizzati gli Open Data ritenuti utili per la mobilità;
- Il terzo capitolo è dedicato interamente a Google Maps e alle sue API: viene fornita una descrizione di tutte le API che sono state utilizzate nell'implementazione dell'app "Route Planner" e vengono riportati i punti di interesse che sono stati utilizzati;
- Nel quarto capitolo viene descritta in dettaglio l'applicazione sviluppata: ne viene fatta un'analisi dettagliata tramite un esempio e viene descritto l'algoritmo ad-hoc che è stato ideato per il calcolo dei percorsi;
- Nell'Appendice A è riportato il codice di risposta dei vari servizi di Google Maps che sono stati utilizzati.

Capitolo 1

Titanium Appcelerator Mobile

Lo sviluppo di app per il mondo mobile è ormai il fulcro di ogni sviluppatore. Gli smartphone sono così diffusi che la richiesta di app è esplosa, per qualsiasi tipologia o settore, non solo per quanto riguarda l'intrattenimento, ma anche per il mondo Enterprise. Sviluppare app mobile, però, significa seguire le scelte degli utenti e i trend del mercato, richiedendo un notevole impegno per rimanere aggiornati sullo sviluppo. La situazione attuale vede un mercato mobile suddiviso principalmente tra le piattaforme Android, iOS e Windows Phone, in ordine di market share. Tutte e tre dispongono di un proprio sistema operativo e di un modello di sviluppo totalmente diverso l'uno dall'altro. Su Android si utilizza Java come linguaggio di programmazione, Eclipse come IDE, su iOS si utilizza Objective C e XCode come IDE, mentre su Windows Phone si utilizza C#/VB e Visual Studio come IDE. Avere le competenze necessarie a coprire tutte e tre le piattaforme è un'impresa piuttosto ardua, soprattutto se si vogliono sviluppare app di qualità. E' necessario perciò avere tre figure professionali distinte che non tutte le realtà di sviluppo si possono permettere.

Sicuramente scrivere un codice unico per più piattaforme dà la possibilità di intervenire su tutti i device minimizzando i tempi di sviluppo di un applicativo e dando la possibilità di aggiornare, modificare e ottimizzare un’applicazione in maniera uniforme e univoca.

Sviluppare app native con il proprio SDK e i propri linguaggi (Xcode/Objective-C per iOS, Eclipse/Java per Android, Visual Studio/C# per Windows Phone) porta a risultati migliori, soprattutto in termini di user experience, velocità di esecuzione e interfacciamento hardware con i vari dispositivi (GPS, fotocamera, accelerometro ecc.).

Tuttavia, il tempo impiegato per crearle si moltiplica per il numero di piattaforme. Sono quindi nati degli strumenti, i framework intermedi, per supportare e velocizzare il lavoro del programmatore.

Questi Mobile framework[1] permettono di scrivere il codice una sola volta e di compilarlo per le varie piattaforme, creando la stessa app per iOS, Android, Windows Phone ecc.

In conclusione si hanno due strade molto differenti tra di loro per la creazione di applicazioni mobile. La prima consiste nello sviluppare una singola applicazione in maniera specifica e differente (differenti linguaggi, differenti ambienti di sviluppo, ecc...) per ogni singola tipologia di device. La seconda consiste nell'utilizzare degli strumenti di terze parti, dotati spesso di un proprio linguaggio di programmazione specifico ed indipendente dalla piattaforma finale, conosciuti come framework intermedi, a cui poi demandare il lavoro di compilazione e distribuzione per ogni singolo sistema supportato. Se nel secondo caso si ha la comodità di dover gestire un unico progetto, per tutto quello che riguarda lo sviluppo, l'aggiornamento e la manutenzione del prodotto, nel primo caso si hanno a disposizione strumenti di sviluppo specifici in grado di ottimizzare tempo, potenzialità e di fornire l'accesso completo all'intera gamma di funzionalità di uno specifico prodotto, là dove invece un framework di terze parti potrebbe supportarne solamente alcune o comunque non le più recenti.

[1] Un framework è un ambiente di sviluppo evoluto su cui un software può essere organizzato e progettato.

Alla base di un framework Mobile c'è una serie di librerie di codice utilizzabili con determinati linguaggi di programmazione (SDK), un editor di codice sorgente, un compilatore, un tool di building automatico e un debugger o altri strumenti ideati per aumentare la velocità di sviluppo del prodotto finito.

Lo scopo di questi framework multipiattaforma è duplice: risparmiare allo sviluppatore la riscrittura di un codice già scritto in precedenza per compiti simili e compilare il codice per i vari tipi di smartphone.

1.1 Scenario

Ogni dispositivo mobile, come ad esempio iOS, Android, BlackBerry, ha una propria gamma più o meno ampia di infrastrutture software per poter creare applicazioni native. Oltre alla scrittura di tali applicazioni in linguaggio nativo, per ogni singolo dispositivo mobile, è possibile utilizzare framework di sviluppo Cross Platform, che permette di sviluppare un'applicazione con un solo codice e di poterla eseguire su più piattaforme hardware. Uno dei maggiori framework open source che si è affermato negli ultimi anni è il framework Appcelerator Titanium.

Appcelerator Titanium, sviluppato da Appcelerator Inc. ed introdotto nel Dicembre 2008, consente di sviluppare app cross platform per iOS, Android e BlackBerry tramite tecnologie Web. Tale framework si compone di un SDK basato su JavaScript e di un ambiente di sviluppo integrato (IDE) basato su Eclipse. Il punto di forza di questo framework sta nel fatto che compila il codice in un'app nativa iPhone, Android o BlackBerry e ciò significa che compila direttamente un'app nativa. Il codice quindi non ha nulla a che vedere con HTML, CSS o sviluppo web in generale, bensì si appoggia a un object model proprietario che ingloba ed espone interfacce per gli oggetti nativi delle varie piattaforme. Questo framework permette, quindi, di sviluppare delle app che assomigliano in tutto e per tutto alle app native, con un'ottima gestione di tutte le componenti hardware degli smartphone, dalla fotocamera al GPS, dall'accelerometro alla bussola. E' possibile inoltre realizzare dei wrapper di funzionalità native ed esportarle tramite JavaScript in Titanium. Ovviamente, però, in questo caso si ha la necessità di scrivere codice in Objective-C e/o Java (a seconda della piattaforma); sono quindi richieste delle conoscenze superiori e il trade-off è determinato dalla quantità di codice nativo che è necessario scrivere in più e i vantaggi che si ottengono.

Appcelerator mette a disposizione la piattaforma Titanium Mobile che consente di sviluppare app cross-platform per iOS, Android e BlackBerry. Lo sviluppo di applicazioni per iOS è possibile soltanto su un Mac in quanto è

necessaria l'installazione del relativo SDK[2] (Software Development Kit) che, in questo caso, non è disponibile per le piattaforme Windows e Linux. Per quanto riguarda Android e BlackBerry invece, è possibile lo sviluppo delle applicazioni in tutte e tre le piattaforme cioè in Windows, OS X e Linux.

Le applicazioni Titanium Mobile vengono sviluppate con il solo linguaggio JavaScript, ma non si tratta di applicazioni web, ma di vere e proprie applicazioni native: la piattaforma contiene infatti un layer per ogni piattaforma che consente di richiamare ed utilizzare tutti i controlli nativi dell'interfaccia grafica offerta da ognuno dei sistemi operativi supportati. Il codice JavaScript al momento della compilazione viene incluso nell'app che non è altro che un "motore" nativo che legge ed interpreta il codice JavaScript a runtime, richiamando poi la grafica e le funzioni native del sistema operativo in cui sta girando l'applicazione.

1.2 Soluzioni di sviluppo mobile in Titanium

Titanium Appcelerator supporta Cross Platform in due diversi modi, tramite "Cross Compilation" e "Mobile Web Compilation".

1.2.1 Cross Compilation

Il framework Titanium Mobile permette di scrivere le applicazioni attraverso l'uso del solo linguaggio JavaScript e di poter poi compilare tale codice per far sì che esso sia eseguibile sui diversi dispositivi mobili (iOS, Android, Blackberry).

[2] Un SDK è tipicamente un insieme di strumenti per lo sviluppo di software che permette la creazione di applicazioni per un determinato pacchetto software, framework, piattaforma hardware, computer, console, sistema operativo o simili piattaforme di sviluppo

Vantaggi:

- l'applicazione finale verrà eseguita sul dispositivo mobile come se fosse un'applicazione nativa, infatti il cross compilation farà un deploy dell'app nei vari linguaggi nativi;
- possibilità di scrivere una solo volta il codice sorgente permettendo così un forte risparmio sia in termini economici che temporali;
- maggiore velocità di esecuzione;
- maggiore integrazione con il "look and feel" della piattaforma.

Svantaggi:

- uno dei più grandi svantaggi è quello di essere vincolati ad una specifica piattaforma. Per esempio ogni qual volta esca una nuova feature, per poter essere adottata dal framework specifico ci vorrà necessariamente del tempo;
- il rendering è fatto attraverso la WebView invece che della UI delle applicazioni native;
- lentezza di esecuzione nell'accesso alle risorse locali;
- molte piattaforme sono di tipo proprietarie quindi per poterle utilizzare bisognerà pagare un determinato ammontare di denaro.

1.2.2 Mobile Web Application

Il framework Titanium Mobile permette anche la creazione di applicazioni Mobile Web cioè applicazioni che girano su browser web ma che sfruttano JavaScript e tutte le novità di HTML5 e CSS per offrire all'utente una esperienza simile a quella presentata dalle applicazioni native.

Vantaggi:

- le mobile web app sono applicazioni indipendenti dalla piattaforma, in quanto visibili a tutti gli smartphone con browser compatibili;
- costa poco sviluppare le mobile web app perché non richiede grosse competenze tecniche dato che sono basate su tecnologie in buona parte note

agli sviluppatori web (quali HTML5, CSS3 e JavaScript) e quindi facilmente adottabili;

- Non richiedendo grosse competenze tecniche sia il tempo di sviluppo che di testing è dimezzato rispetto alla creazione di una applicazione nativa.

Svantaggi:

- le mobile web apps sono poco performanti dato che girano all'interno di un browser web;
- richiede una connessione Internet attiva;
- offrono una peggiore user experience: non tutti i browser fanno rendering nella stessa maniera, quindi si potrebbero avere esperienze differenti in base ai dispositivi ed ai browser utilizzati;
- non è possibile utilizzare tutte le views standard (barre, bottoni, tabs, ecc.) normalmente utilizzate nelle app native, ma solo quelle interfacce utilizzate nelle classiche pagine web;
- non è possibile accedere a tutte le funzionalità del dispositivo mobile (es. fotocamera, sensori, ecc.).

1.3 Confronto di Titanium con altri framework

Viene ora descritto il posizionamento di Titanium rispetto ad altri framework in termini di performance e flessibilità.

Si può notare in *figura 1* come la zona vicina al punto 1 è relativa ai framework nativi dato che si hanno massime performance ma minima flessibilità. Il framework più vicino al linguaggio nativo è il framework RubyMotion. Le app sviluppate con RubyMotion sono eseguite come quelle create in Objective-C, impiegando le stesse risorse hardware. Per questo framework le performance sono massime proprio perché le applicazioni sono scritte in un linguaggio più vicino a quello nativo; la flessibilità però è minima perché è possibile far eseguire le applicazioni create in una sola piattaforma (iOS nel caso del framework RubyMotion).

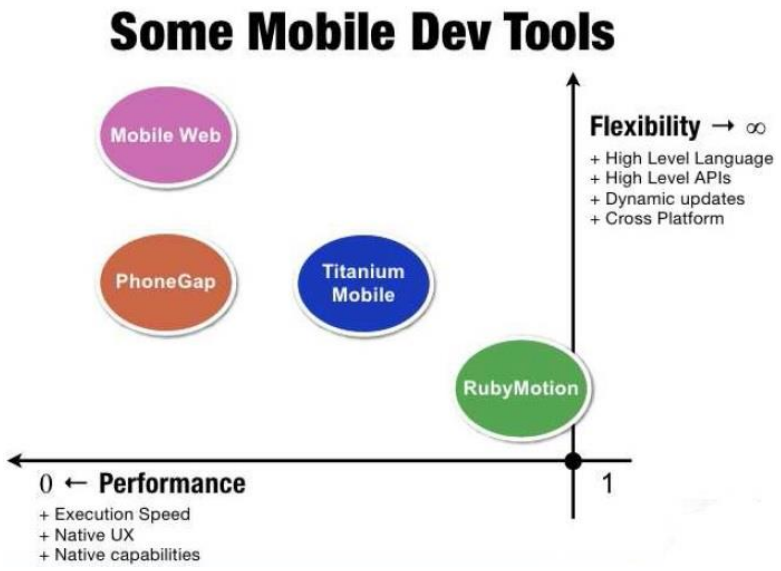

Figura 1: Posizionamento di Titanium rispetto agli altri framework

All'estremo opposto è possibile trovare il Mobile Web cioè tutte quelle applicazioni che girano in un browser e che sfruttano le tecnologie Web, in particolare HTML5, JavaScript e CSS3. Le performance sono molto basse proprio perché l'applicazione sviluppata è eseguita all'interno di un browser e, oltre all'overhead dell'applicazione, è necessario aggiungerci l'overhead del browser che esegue l'applicazione. La flessibilità però è massima perché l'HTML, il CSS e il JavaScript possono essere eseguiti da un qualunque browser su un qualunque dispositivo.

PhoneGap si posiziona un pochino sotto nel grafico rispetto al Mobile Web perché consente di sviluppare delle applicazioni ibride native attraverso l'utilizzo di tecnologie web quali HTML5, CSS3 e JavaScript. Tali applicazioni vengono eseguite localmente all'interno di un'applicazione nativa. Rispetto al Mobile Web si hanno stesse performance proprio perché vengono utilizzate le stesse tecnologie Web ma si ha minore flessibilità perché le applicazioni sviluppate non sono più eseguite all'interno di un browser ma vengono eseguite all'interno di un'applicazione nativa e quindi diminuiscono le piattaforme supportate.

Il framework Titanium Mobile si posiziona in mezzo al grafico perché è un livello sopra il codice nativo ma abbandona i concetti di browser e di tecnologie Web cioè viene utilizzato solo il linguaggio Javascript.

Viene ora fatto un paragone in *figura 2* tra i framework più conosciuti riguardo le piattaforme supportate, il target, i linguaggi di sviluppo e le licenze.

Figura 2: Confronto framework Cross Platform

1.4 Architettura Titanium Appcelerator

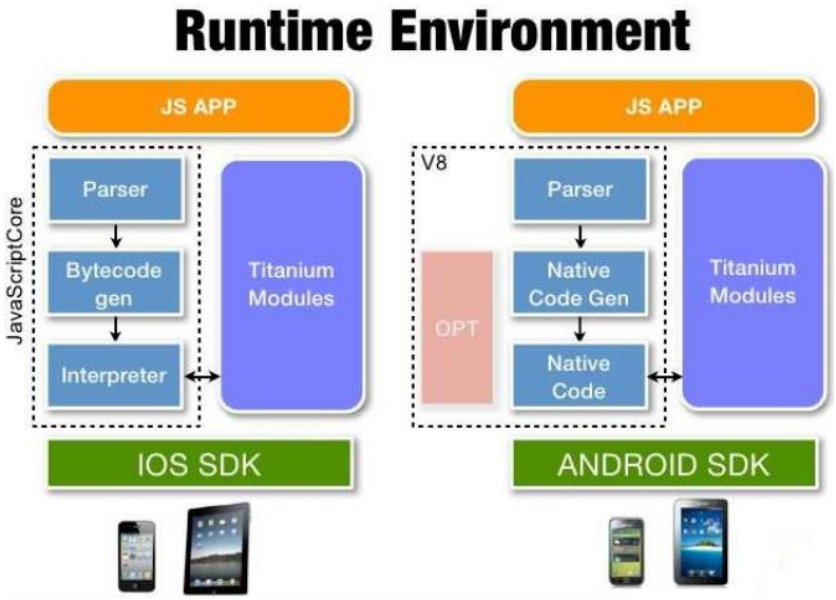

Figura 3: L'architettura di Titanium Mobile

In *figura 3* è possibile vedere l'architettura di Titanium Mobile.

Si può notare che esiste un linguaggio comune che è Javascript per poter eseguire l'applicazione sia sul sistema operativo iOS e sia su Android.

Per quanto riguarda iOS, Titanium utilizza l'interprete JavaScriptCore (l'interprete nativo di iOS) che, dialogando con i moduli nativi di Titanium permette di ottenere un file .ipa che potrà essere eseguito solo sul sistema operativo iOS. I moduli fanno da ponte cioè traducono una certa funzionalità scritta in JavaScript in un servizio specifico del sistema operativo iOS. In questo modo quando per esempio si crea un bottone in JavaScript, l'interprete JavaScriptCore creerà un bottone specifico per iOS.

Per quanto riguarda Android, Titanium utilizza l'interprete V8 che compila il codice JavaScript per ottenere un codice nativo (file .apk) che potrà essere eseguito sul solo sistema operativo Android. Durante l'esecuzione del file .apk, l'interprete V8 dialoga con i moduli nativi esattamente come succede per iOS. Una differenza

qui sta nel fatto che l'interprete V8 viene completamente incluso nel file finale .apk, dato che non è l'interprete nativo di Android, per cui l'applicazione risulterà più pesante rispetto ad iOS.

1.5 Altri framework Cross Platform

Dopo aver preso in considerazione il framework Titanium Appcelerator, tramite il quale si è proceduto a realizzare l'applicazione "Route Planner", si vuole ora fornire un elenco dei framework Cross Platform più famosi e conosciuti nell'ambito mobile.

1.5.1 PhoneGap

Phonegap è un progetto Open Source della Nitobi Software, un'azienda che crea applicazioni mobile e web applications da più di dieci anni, recentemente acquisita da Adobe. Consiste in un insieme di librerie statiche che permettono di sviluppare velocemente ed efficacemente applicazioni per dispositivi mobili di diverse piattaforme.

In particolare PhoneGap è compatibile contemporaneamente con ben 7 sistemi operativi mobili:

- iOS;
- Android;
- Windows Phone;
- Blackberry OS;
- webOS;
- Symbian;
- Bada.

Sono tanti i vantaggi che si possono ottenere con PhoneGap, tra cui l'utilizzo di HTML5 e CSS3, o la possibilità di scrivere in JavaScript il codice o ancora, di poter accedere alle funzionalità native della piattaforma sulla quale si ha l'intenzione di far girare l'app (notifiche push, vibrazione, accelerometro, fotocamera ecc...).

Tuttavia presenta anche dei limiti davvero pesanti a causa dei quali la sua scelta è stata scartata. Si è fondamentalmente costretti a reinventare la ruota. Bisogna dimenticarsi di tutti i controlli nativi per le varie piattaforme, con il risultato che le applicazioni perdono completamente il "*look and feel*" tipico del dispositivo su cui girano, con il rischio di non essere accettate dagli store (App Store, Google Play, …) una volta terminate. Il codice è interpretato, niente compilazione. Quindi gli unici errori sono a runtime, questo complica la parte relativa allo sviluppo dell'applicazione stessa. Per alcune funzionalità (es. riproduzione video) è necessario comunque imparare il codice nativo. Infine PhoneGap non è "*Write Once Run Anywhere*", semplicemente perché se si vogliono supportare "*n*" dispositivi si dovranno avere "*n*" progetti, magari ognuno la copia dell'altro, ma sempre "*n*" progetti da manutenere.

1.5.2 Sencha Touch

Sencha Touch è una piattaforma della compagnia Sencha (precedentemente conosciuta come ExtJS) grazie alla quale è possibile creare app per dispositivi mobili con HTML5, sfruttando ampiamente le capacità di CSS3 e Javascript. Sencha Touch è multi-piattaforma, ossia è stata disegnata per essere compatibile con iOS, Android, BlackBerry e Windows Phone.

Sencha Touch viene definito un framework HTML5 dai suoi creatori. Le ragioni ci sono tutte. Con HTML5, le applicazioni Sencha Touch possono essere usate offline e la Geolocation HTML5 permette una facile integrazione di dati geografici nelle app. L'uso di CSS3 permette di non avere praticamente nessuna

immagine nelle librerie dei componenti: gli stili, i bordi, i gradienti, le ombre, le transizioni, i menu, i pulsanti, assolutamente tutti i componenti sono in CSS puro.

Sencha Touch è un framework JavaScript che introduce un nuovo modo di sviluppare applicazioni Web per il mondo mobile. Tramite una sintassi ed una logica un po' particolari, ed in qualche modo ispirate al mondo Swing di Java, Sencha Touch consente di descrivere interfacce web attraverso istruzioni JavaScript e tecniche di templating.

Il framework si prende poi in carico l'onere di generare il documento HTML finale e nel farlo applica al descrittore di interfaccia un tema che può ricalcare quello delle applicazioni native di un particolare modello di smartphone (iPhone, Android, BlackBerry) o può essere originale e sviluppato da noi.

Sencha Touch trae vantaggio dalle nuove specifiche HTML5 e consente di memorizzare informazioni direttamente sul device dell'utente utilizzando le API localStorage. Sono inoltre a disposizione dello sviluppatore componenti Audio e Video che basano le loro funzionalità sui corrispettivi elementi HTML5.

1.5.3 JQuery Mobile

Basato su jQuery[3] ed ispirato al progetto UI, jQuery Mobile è un progetto molto giovane, ma offre una buona stabilità ed un numero di funzionalità adeguate per sviluppare applicazioni web mobile complete.

La caratteristica peculiare di jQuery Mobile è che per realizzare un'applicazione base, basta scrivere del codice HTML5. La libreria, infatti, fa leva sulla struttura semantica delle pagine HTML5 e sugli attributi *data-* per definire le varie parti dell'interfaccia. Una volta caricata la pagina, jQuery Mobile utilizzerà questa struttura per arricchirla con altri tag e agganciare gli eventi e le interazioni ai componenti dell'applicazione.

[3] jQuery è un framework basato su JavaScript che facilita non lo sviluppo web offrendo una serie di strumenti per la gestione di animazioni, eventi, AJAX e quant'altro con l'uso di codice solido, semplice e soprattutto già pronto.

Rispetto ad altre librerie come Sencha Touch, jQuery Mobile ha quindi il vantaggio di non richiedere tecnologie particolari, per molti browser mobile JavaScript lo è, per rendere un contenuto navigabile.

Le piattaforme compatibili con JQuery Mobile sono iOS, Android, Windows Phone, BlackBerry e tanti altri.

1.5.4 JQTouch

jQTouch è un plugin di jQuery che permette di sviluppare applicazioni mobile per browser basati su WebKit, compatibili quindi coi sistemi iOS (iPhone e iPad) e Android.

Tutto il lavoro è basato su semplice codice HTML, CSS e JavaScript, ma le applicazioni realizzate risultano essere molto simili a delle applicazioni native di iOS e Android in quanto ne ricalcano perfettamente lo stile ed il funzionamento.

1.5.5 MonoTouch

MonoTouch è uno strumento a pagamento e proprietaria di Mono grazie al quale è possibile realizzare applicazioni .NET scritte in C# che poi potranno essere eseguite su iPhone, iPod e iPad. MonoTouch include, oltre ad un IDE, anche un comodo simulatore di iPhone che permetterà di testare le applicazioni senza dover necessariamente possedere un iPhone.

MonoTouch è sostanzialmente un compilatore statico che converte codice C# .NET in codice nativo Apple. Non è quindi un compilatore JIT (Just In Time) in quanto le applicazioni per iPhone non possono accedere a compilatori JIT a causa di una precisa scelta da parte di Apple per impedire il diffondersi di malware ed altro software dannoso.

Le applicazioni .NET saranno quindi convertite in puro codice nativo per iPhone, iPod e iPad.

Capitolo 2

Open Data

Con “Open data” o “Dati aperti” si fa riferimento ad una filosofia e ad una pratica di condivisione di determinati dati e informazioni in modo da consentirne il libero accesso direttamente online, in maniera semplice, veloce e senza limitazioni.

La definizione tecnica che OpenDefinition.org dà di Open Data è la seguente: “*Open data is data that can be freely used, reused and redistributed by anyone - subject only, at most, to the requirement to attribute and share-alike*”.

Con “Data” si intende ogni singolo pezzo di informazione di qualsiasi tipo (dalle immagini ai numeri, definizioni testuali, mappe, audio, ecc.) che:

- sono descrizioni dirette di fatti (es. le temperature medie di una città, le fasce di reddito in alcuni paesi, il percorso di un fiume tracciabile su una mappa, ...) o sono strettamente legate a fatti, quindi non sono copiabili;
- sono riproducibili senza ambiguità quando il metodo usato per generarli è conosciuto in tutti i suoi dettagli;
- sono parti, o possono essere immediatamente usati come parte, di informazioni più grandi;

Figura 4: Tipi di dati

- hanno (quasi sempre) molto più significato e valore quando sono legati tra loro e completati da "metadata" (dati riguardanti altri dati, per esempio il nome dell'autore o la data di creazione);
- possono, grazie alle caratteristiche sopra descritte, essere espressi e archiviati in formati digitali, anche se non originariamente generati sotto questa forma.

Come si può vedere in *figura 4*, i dati più diffusi sono: dati scientifici (es. dati geografici, medici, genoma, composti chimici, ...) o dati governativi che sono raccolti dalle Pubbliche Amministrazioni (es. censimenti) o che riguardano le Pubbliche Amministrazioni (es. bilanci).

Il detentore dei dati rilascia gli stessi in modo tale da poter essere rielaborati liberamente da chiunque (aziende, cittadini, giornalisti, ricercatori) così da poter essere riutilizzati per altri scopi, e quindi generare nuove conoscenze ed aprire nuove strade di sviluppo sociale ed economico.

L'Open data si richiama alla più ampia disciplina dell'Open Government, cioè una dottrina che prevede l'apertura delle informazioni della pubblica amministrazione, intesa sia in termini di trasparenza che di partecipazione diretta dei cittadini, anche attraverso l'uso delle nuove tecnologie della comunicazione. Questa pratica è piuttosto diffusa nei paesi anglosassoni dove grazie ad essa le amministrazioni hanno avuto la possibilità di superare gli schemi rigidi e burocratici di accesso ai dati e di gestione delle risorse informative, sia al loro interno, sia nei confronti della cittadinanza acquisendo così maggiore efficienza e trasparenza.

In sintesi i "Dati aperti" devono essere:

- **Pubblici:** visibili a tutti;
- **Accessibili:** Gli utenti devono poter usare queste risorse direttamente attraverso i protocolli internet, senza alcuna sottoscrizione di contratto, pagamento, registrazione o richiesta ufficiale;
- **Liberi:** Gli utenti devono poter utilizzare e processare i dati attraverso programmi, applicazioni e interfacce aperte, al contempo i dati devono essere

pubblicati e riusabili in formati semplici e generalmente supportati dai programmi più utilizzati dalla collettività digitalizzata;

- **Completi**: I dati devono comprendere tutte le componenti (metadati) che consentano di esportarli, utilizzarli *online* e *offline*, integrarli e aggregarli con altre risorse e diffonderli in rete;
- **Primari:** le risorse digitali devono essere strutturate in modo tale che i dati siano presentati in maniera sufficientemente granulare, così che possano essere utilizzate dagli utenti per integrarle e aggregarle con altri dati e contenuti in formato digitale;
- **Riutilizzabili:** Affinché i dati siano effettivamente aperti, gli utenti devono essere messi in condizione di riutilizzarli e integrarli, fino a creare nuove risorse, applicazioni e servizi di pubblica utilità;
- **Tempestivi:** gli utenti devono essere messi in condizione di accedere e utilizzare i dati presenti in Rete in modo rapido e immediato massimizzando il valore e l'utilità derivanti da accesso e uso di queste risorse.
- **Documentati:** sono descritti i passaggi di raccolta ed elaborazione;
- **Ricercabili:** i dati devono essere facilmente identificabili in rete, grazie a cataloghi e archivi facilmente indicizzabili dai motori di ricerca.
- **Permanenti:** Le peculiarità fino ad ora descritte devono caratterizzare i dati nel corso del loro intero ciclo di vita.

2.1 Open Data nella Pubblica Amministrazione

Le istituzioni pubbliche producono e possiedono una enorme quantità di dati che appartengono alla collettività. Parlare di Open Data con riferimento ai dati della P.A. significa, quindi, rendere i dati e le informazioni disponibili e accessibili direttamente online per cittadini ed imprese sia per elaborazioni che per creare applicazioni di pubblica utilità.

Le P.A. devono utilizzare le nuove tecnologie per essere aperte e trasparenti verso i cittadini attraverso la distribuzione dei dati, evidenziando così l'efficienza e l'utilità del loro lavoro.

Gli obiettivi delle PA sono:

- Aumentare la trasparenza e il coinvolgimento dei cittadini, istituzioni e tutti gli altri stakeholders;
- Migliorare il mercato;
- Favorire l'innovazione tecnologica, lo sviluppo e la diffusione di nuove tecnologie;
- Creare valore sui dati e benefici per i cittadini.

L'adozione di un modello Open Data per l'accesso e l'utilizzo in Rete di dati e risorse legati all'ambito pubblico rappresenta un passaggio necessario per il rinnovamento delle istituzioni nella direzione di "**apertura**" e "**trasparenza**", a tutti i livelli amministrativi.

Rendere l'amministrazione aperta: distribuire i dati pubblici in un formato aperto e libero da restrizioni sia dal punto di vista dell'accesso che dell'integrazione e del riutilizzo, rappresenta il presupposto di base affinché possa svilupparsi un vero e proprio processo di collaborazione tra le istituzioni e la comunità dei cittadini sulle scelte di governo, anche la rielaborazione in forma nuova e diversa dei dati messi a disposizione. Mediante strategie di apertura dei dati della Pubblica Amministrazione i cittadini non sono più soltanto consumatori passivi di informazioni messe a disposizione dalle Amministrazioni. Hanno invece l'opportunità di riutilizzare e integrare i dati messi loro a disposizione, fino a sviluppare servizi e applicazioni a vantaggio dell'intera comunità di utenti, che vanno ad affiancarsi a quelli creati centralmente dalle Istituzioni. In questo modo i cittadini collaborano effettivamente con i soggetti istituzionali e partecipano attivamente alle azioni di governo della cosa pubblica.

Rendere l'amministrazione trasparente: il libero accesso a documenti, atti e saperi sul governo della *res publica* e sulle scelte politico istituzionali compiute dalle Amministrazioni è un aspetto centrale per la trasparenza delle Istituzioni. Tale

aspetto stimola e facilita i cittadini ad un controllo continuo e costante sull'operato e sui processi decisionali dei soggetti istituzionali. Attraverso l'attuazione di politiche di apertura reale delle informazioni e dei dati pubblici, i cittadini sono in condizione di verificare l'efficienza dell'apparato burocratico.

In ogni caso, se da una parte l'adozione del formato Open Data costituisce una condizione necessaria per lo svecchiamento delle amministrazioni verso "apertura" e "trasparenza", dall'altra non si tratta di un passaggio sufficiente. Il rinnovamento delle istituzioni, dal punto di vista della gestione dei dati e delle informazioni e del contatto con i cittadini, deve passare necessariamente per un ripensamento del modello organizzativo tradizionale e degli schemi burocratici che caratterizzano gli enti pubblici.

2.2 Open Data nel mondo e in Italia

Una serie di data store governativi sono stati aperti negli ultimi anni in tutto il mondo. Il primo e più celebre, che ha sostanzialmente fatto da modello ai successivi, è stato *data.gov* americano, lanciato dal governo Obama a seguito della Direttiva sull'Open government nel dicembre 2009 come misura anticrisi, per permettere di produrre beni e servizi attraverso i dati aperti intesi come materia prima. Di lì a pochi mesi anche il Regno Unito ha aperto il suo *data.gov.uk* fortemente voluto e sponsorizzato da Tim Berners-Lee, l'inventore del World Wide Web, seguito dalla Nuova Zelanda con *data.govt.nz*. In pochi anni la pratica degli open data e dei data store governativi si è estesa alla maggior parte dei paesi più industrializzati del mondo.

Sono molteplici anche le iniziative d'apertura del patrimonio informativo avviate in Italia da parte di pubbliche amministrazioni centrali e locali. Il primo data store italiano è stato quello della Regione Piemonte (2010), *dati.piemonte.it*, all'interno del quale sono catalogati dati aperti riconducibili ai vari enti regionali (comuni, province, ...). Dopo circa un anno dalla nascita del data store piemontese

anche la Regione Emilia-Romagna, ad ottobre 2011, pubblica online il suo catalogo di dataset (*dati.emilia-romagna.it*). Con il lancio del portale *dati.gov.it*, avvenuto il 18 Ottobre 2011, si è aperta una nuova stagione per l'innovazione e la trasparenza nella P.A., una strada verso l'Open Data italiano. Il processo di gestione delle informazioni prodotte sta progressivamente evolvendo, verso modelli più aperti. Nell'ordinamento Italiano però non esiste una norma che impone di rendere pubblici i dati in possesso delle P.A., come la Direttiva americana lanciata da Obama. La maggior parte degli archivi sono ancora "scatole chiuse".

Come si può vedere dal grafico in *figura 5*, in questo momento in italia sono

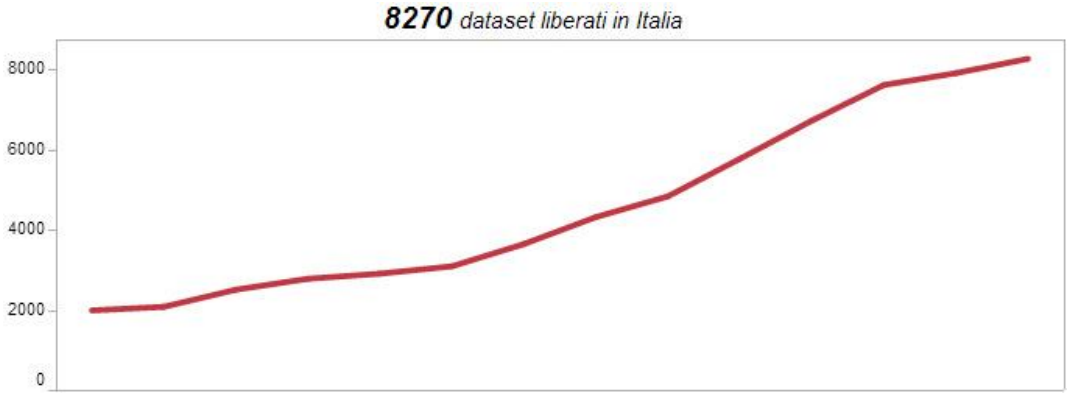

Figura 5: Numero totale di dataset liberati in Italia, a partire dal mese di Marzo 2012 a Dicembre 2013

disponibili oltre 8000 dataset rilasciati in formato aperto. In tale figura viene mostrato l'andamento totale di dataset liberati in Italia, a partire dal mese di marzo 2012 fino a dicembre 2013.

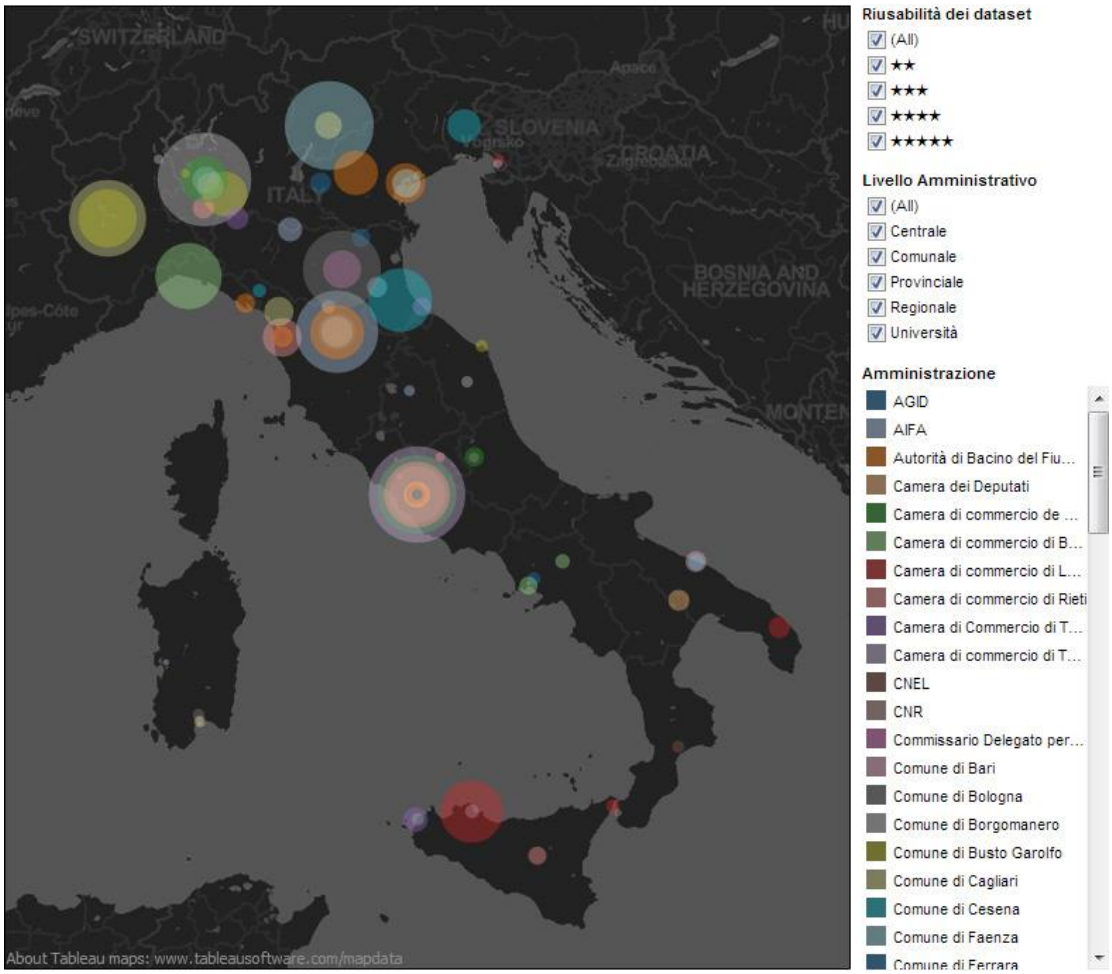

Figura 6: distribuzione geografica delle amministrazioni italiane che rilasciano Open Data

Nella mappa in *figura 6* viene visualizzata la distribuzione geografica delle amministrazioni che rilasciano open data, l'area di ognuna delle bolle è direttamente proporzionale al numero di dataset rilasciati dalla specifica amministrazione.

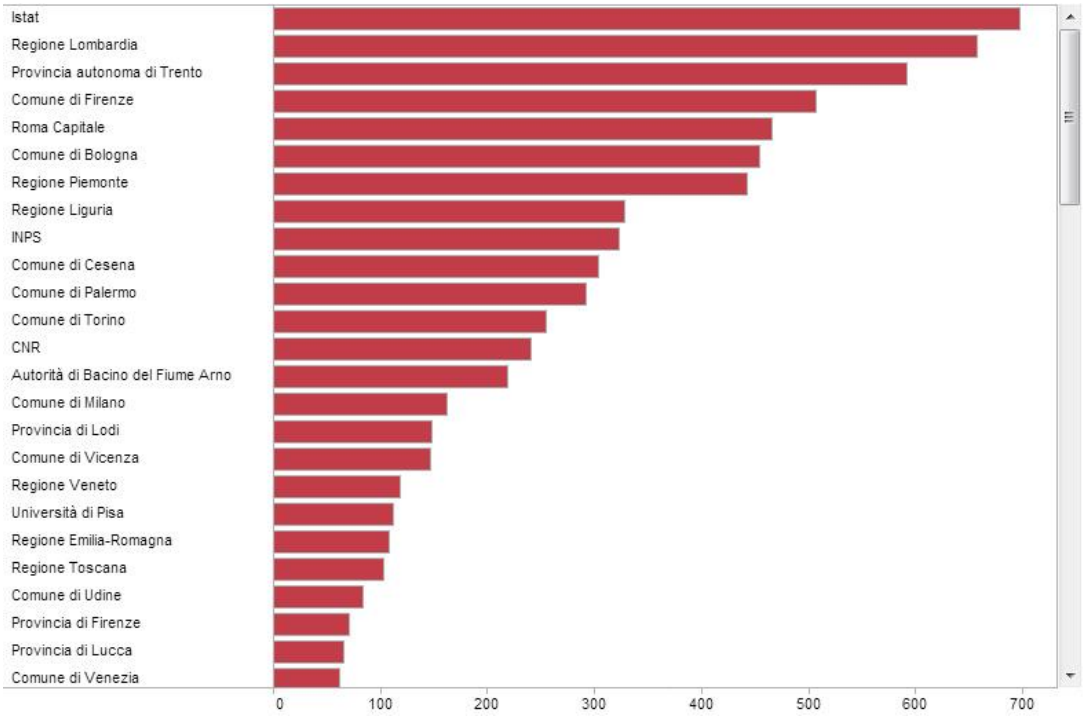

Figura 7: Confronto tra le pubbliche amministrazioni che pubblicano i propri dati in formato aperto

In *figura 7* viene fatto un confronto tra le pubbliche amministrazioni che pubblicano i propri dati in formato aperto

Figura 8: Ripartizione del numero di dataset rilasciati rispetto al livello amministrativo degli enti

In *figura 8* viene fatta una ripartizione del numero di dataset rilasciati rispetto al livello amministrativo degli enti.

2.3 Open Data come strategia economica e nuove opportunità

La vera opportunità commerciale nel praticare Open Data è l'aiutare le persone a scoprire tutti questi dati, a vedere il loro potenziale e a realizzare come possono essere utilizzati per avviare attività di mercato, prendere decisioni migliori e identificare nuove opportunità. Inoltre la manutenzione dei datasets e l'aggiornamento continuo dei dati sono importantissimi per creare valore e sviluppare efficienti servizi pubblici o buone opportunità commerciali. Affinchè creino valore, in particolar modo se entrano a far parte di un modello di business, i dati devono possedere un livello di aggiornamento garantito (in termini di qualità del dato e tempi certi di rilascio). Il valore dell'Open Data sta nel suo utilizzo, non nella sua grezza esistenza.

Tali dati, infatti, una volta liberati dalla segretezza, diventano la materia prima per la produzione di beni e servizi del mondo digitale e quindi un'opportunità per l'imprenditore e per lo startupper che, raccogliendo le informazioni su un dato settore, possono sviluppare un'applicazione web, o un'applicazione per smartphone creando un modello di business.

Lo sviluppo di applicazioni per dispositivi mobili (tablet e telefoni cellulari) è stata la prima idea di business che si è sviluppata con l'apertura dei dati pubblici. Sono tantissime le Apps disponibili create con i dati rilasciati dalle PA e la maggior parte dei siti governativi contiene una sezione in cui vengono descritte le Apps sviluppate. Sicuramente questo settore di business non è quello che porta maggiori entrate economiche ma sicuramente è il più sfruttato. Meteo, trasporti e altre utilità pubbliche sono tra le Apps più scaricate per gli Smartphone. Spesso vengono anche organizzati dei concorsi per lo sviluppo di applicazioni con i dati forniti dalle PA, che creano valore economico per lo Stato. Molte applicazioni sono basate sulla geolocalizzazione e permettono, ad esempio, di individuare tutti i servizi presenti in

una determinata zona, anche facendo uso di più dataset in modo da offrire un'informazione più esaustiva. Infatti anche molte zone turistiche utilizzano applicazioni e siti web per mostrare i principali luoghi di interessi e svago, al fine di pubblicizzare le attività della zona. Più persone conoscono un servizio, più aumenta la possibilità che questo servizio venga utilizzato (es. trasporti, ristorazione, negozi, luoghi di svago come cinema, bowling, discoteche, ecc.).

2.4 Open Data nell'app "Route Planner"

L'app per il sistema operativo iOS che è stata sviluppata durante il tirocinio sfrutta pienamente il concetto di Open Data. Durante la prima fase di lavoro è stata fatta una ricerca approfondita su quali tipi di dati della regione Emilia-Romagna potessero essere utili, e quindi dare un valore aggiunto, alla pianificazione dei percorsi.

E' stata fatta un'analisi iniziale per comprendere meglio quali tipi di aziende gestiscono il trasporto pubblico nella regione Emilia-Romagna e sono state individuate quattro diverse aziende: TPER (Trasporto Passeggeri Emilia-Romagna), SETA (Società Emiliana Trasporti Autofiloviari), START ROMAGNA e TEP (Trasporti pubblici Parma).

Poi è stata fatta un'ulteriore analisi sugli Open Data disponibili nei comuni di Modena e Bologna. Tale analisi ha evidenziato il fatto che non esiste un portale dedicato agli Open Data del comune di Modena, ma esiste soltanto un portale dedicato al comune di Bologna.

Verrà ora fatta una piccola descrizione per ciascuna azienda che è stata analizzata ed infine verranno analizzati gli Open Data presenti nel portale del comune di Bologna.

2.4.1 TPER (Trasporto Passeggeri Emilia-Romagna)

TPER (Trasporto Passeggeri Emilia-Romagna) è la società di trasporti pubblici nata il 1° febbraio 2012 dalla fusione dei rami-trasporto di ATC, azienda di trasporti su gomma di Bologna e Ferrara, e FER, società regionale ferroviaria. TPER si posiziona al sesto posto per fatturato tra gli operatori di trasporto passeggeri in Italia ed è la più grande azienda dell'Emilia-Romagna per numeri e volumi di servizio nel settore del trasporto pubblico di persone. Le aree di attività della società coprono diversi segmenti del settore del trasporto, da quello automobilistico, a quello filoviario, a quello ferroviario di passeggeri. Il valore della produzione annua della nuova società per azioni – in cui operano oltre 2.500 dipendenti – è stimato in 246 milioni di Euro. Con i suoi 1.350 bus, l'azienda parte da una percorrenza annua di circa 50 milioni di chilometri; 5 milioni di chilometri, invece, quelli effettuati per il trasporto passeggeri in ambito ferroviario. TPER trasporta ogni giorno oltre 340.000 passeggeri su bus o filobus; mentre 30.000 sono i viaggiatori che quotidianamente utilizzano il trasporto ferroviario regionale.

All'interno del sito di TPER (www.tper.it/tper-open-data) sono stati trovati diversi Open Data riguardo il trasporto pubblico. Tutti i dati che sono stati trovati sono disponibili nei formati CSV e XML. Inoltre tali dati possono essere acceduti da un'applicazione esterna mediante web service. I dati per il trasporto pubblico che sono resi disponibili come Open Data comprendono:

- Elenco delle linee ferroviarie: tali dati descrivono le linee ferroviarie presenti nella regione Emilia-Romagna. La relazione in esame è composta da tre attributi che sono: codice_linea, sigla, denominazione. Un esempio di tupla è (110, BOPOR, BOLOGNA – PORTOMAGGIORE);
- Elenco delle stazioni ferroviarie: tali dati descrivono le stazioni ferroviarie presenti nella regione Emilia-Romagna. La relazione in esame è composta da nove attributi che sono: codice, denominazione, ubicazione, comune, coordinata_x, coordinata_y, latitudine, longitudine, codice_zona. Un esempio

di tupla è (100, BOLOGNA CENTRALE, PIAZZA DELLE MEDAGLIE D'ORO 4, BOLOGNA, '', '', 44.5063, 11.3423, '');

- Elenco degli archi delle linee bus: tali dati descrivono nei minimi dettagli tutti gli archi (sottopercorsi) che fanno i bus da una certa fermata alla fermata successiva. La relazione in esame è composta da sette attributi che sono: codice, offset_posizione, coordinata_x, coordinata_y, latitudine, longitudine, codice_fermata. Un esempio di tupla è (1, 0, 677451, 933636, 44,532431, 11,232319, 6510);
- Elenco delle fermate bus: tali dati descrivono tutte le fermate dei bus nelle province di Bologna e Ferrara. La relazione in esame è composta da nove attributi che sono: codice, denominazione, ubicazione, comune, coordinata_x, coordinata_y, latitudine, longitudine, codice_zona. Un esempio di tupla è (1, STAZIONE CENTRALE, PIAZZA MEDAGLIE D`ORO (PENSILINA C), BOLOGNA, 686330, 930913, 44,505694, 11,342996, 500);
- Elenco delle linee bus: tali dati descrivono soltanto quali linee di bus sono presenti nelle province di Bologna e Ferrara. La relazione in esame è composta da un solo attributo che è codice_linea. Un esempio di tupla è (1);
- Linee bus come sequenza di archi: tali dati descrivono esattamente tutti i percorsi che fanno i bus sulla base degli archi (sottopercorsi) contenuti nella relazione "Elenco degli archi delle linee bus". Quindi per ogni linea bus si hanno tutte le informazioni necessarie per tracciare minuziosamente il tragitto. La relazione in esame è composta da sei attributi che sono: codice_linea, verso, percorso, posizione, codice_arco, bacino. Un esempio di tupla è (1, asc, 1, 1, 5057, BO);
- Linee bus come sequenza di fermate: tali dati descrivono i percorsi che fanno i bus sulla base delle fermate contenute nella relazione "Elenco delle fermate bus". Quindi per ogni linea bus si hanno tutte le fermate che devono essere raggiunte. La relazione in esame è composta da 10 attributi che sono: codice_linea, codice_fermata, denominazione, ubicazione, comune,

coordinata_x, coordinata_y, latitudine, longitudine, codice_zona. Un esempio di tupla è (1, 110105, FORNACE, VIALE AMENDOLA 68, IMOLA, 714533, 915843, 44.362326, 11.691198, 510);

TPER rende inoltre disponibili servizi interattivi real-time sulla disponibilità dei propri mezzi, attraverso i servizi CHIAMATRENO e Hello Bus. Il servizio CHIAMATRENO consente agli utenti di conoscere con una telefonata gratuita eventuali ritardi, soppressioni e provvedimenti relativi ai treni in transito nella stazione di interesse. Il servizio Hello Bus fornisce informazioni sull'orario di arrivo di ciascun autobus alla fermata di interesse del Cliente. Questo tipo di servizio è disponibile anche mediante web service.

I dati di interesse per quanto riguarda l'app che è stata realizzata, riguardano soprattutto tutte le tratte dei bus, le loro fermate e i tempi di arrivo di un certo bus in una determinata fermata.

Tuttavia si è scoperto che tutti gli Open Data messi a disposizione, sono stati comunicati e quindi esportati in Google Maps. Quindi non è stato necessario usare direttamente i dati da TPER. E' stato più conveniente interrogare direttamente Google Maps per quanto riguarda la pianificazione di un certo percorso utilizzando i mezzi pubblici. Grazie alle API messe a disposizione da Google Maps è stato infatti possibile interrogare tramite una richiesta HTTP il percorso migliore dati un'origine ed una destinazione, disinteressandosi completamente di tutte le problematiche in background, come per esempio la scelta degli algoritmi utilizzati per cercare i cammini minimi.

2.4.2 SETA Spa (Società Emiliana Trasporti Autofiloviari)

SETA Spa (Società Emiliana Trasporti Autofiloviari) è il nuovo gestore unico del servizio di trasporto pubblico locale automobilistico nei territori provinciali di Modena, Reggio Emilia e Piacenza.

Operativa dal 1° gennaio 2012, SETA nasce dall'aggregazione delle aziende di trasporto pubblico di Modena, Reggio Emilia e Piacenza: per dimensioni e distribuzione territoriale, l'aggregazione - scaturita dalla confluenza nella nuova Società di quattro soggetti: Atcm di Modena; Tempi di Piacenza; AE-Autolinee dell'Emilia e il ramo gomma Act di Reggio Emilia - è una delle maggiori operazioni di fusione aziendale ad oggi mai avvenute, in ambito nazionale, nel settore del trasporto pubblico locale. La nuova azienda unitaria operante nel territorio dell'Emilia occidentale è responsabile di tutto il sistema di produzione del servizio di TPL su gomma dei tre bacini provinciali: dall'esercizio dei trasporti bus urbani ed extraurbani, alla manutenzione dei mezzi, alla vendita dei titoli di viaggio, alla gestione delle biglietterie e dei servizi per l'utenza (informazioni, reclami, ecc.). SETA Spa gestisce 29,6 milioni di chilometri di trasporto pubblico locale, ha un organico di 1.059 dipendenti e 946 mezzi marcianti, per un valore della produzione di poco inferiore a 103 milioni di euro.

All'interno del sito di SETA (http://www.setaweb.it/index.php) non è stato trovato alcun Open Data riguardo il trasporto pubblico. Tuttavia un'attenta analisi ha portato alla scoperta che SETA possiede i dati del trasporto pubblico delle provincie di Modena, Reggio Emilia e Piacenza, ma tali dati non sono stati pubblicati sul sito Web e, quindi, non è possibile accedervi. Inoltre si è scoperto che tutti i dati di SETA sono stati comunicati e quindi esportati in Google Maps. Quindi sono state utilizzate direttamente le API di Google Maps per utilizzare i dati del trasporto pubblico delle provincie di Modena, Reggio Emilia e Piacenza.

2.4.3 START ROMAGNA

START ROMAGNA è la società di trasporto pubblico dell'area romagnola nella quale sono confluite le tre Aziende storiche di gestione del trasporto, ovvero AVM (Forlì-Cesena), ATM (Ravenna) e TRAM SERVIZI (Rimini). Il 4 Novembre 2009 ci fu il primo atto di costituzione della Holding che unisce le tre aziende

storiche del trasporto romagnolo. Il 30 Luglio 2010 la Holding si amplia con l'ingresso anche degli enti soci provenienti dai tre bacini di trasporto della Romagna. Il 28 Settembre 2011 venne siglato l'atto di fusione. START ROMAGNA è la nuova Società di gestione del trasporto pubblico in Romagna e il 31 Dicembre 2011 si concluse il processo di fusione in START ROMAGNA. Dal 1° gennaio 2012, a tutti gli effetti START ROMAGNA è il gestore del trasporto pubblico locale per la Romagna.

Sono stati cercati Open Data sul trasporto pubblico all'interno del sito Web di START ROMAGNA (www.startromagna.it/) ma, come per l'azienda SETA Spa, non è stato trovato alcun Open Data ed è stato scoperto che i dati posseduti, ma non pubblicati, sono stati comunicati a Google Maps. Sono state quindi utilizzate le API di Google Maps per reperire i dati sul trasporto pubblico nelle provincie di Forlì-Cesena, Ravenna e Rimini.

2.4.4 TEP S.p.A. (Trasporti pubblici Parma)

La TEP S.p.A., di proprietà del Comune di Parma e della Provincia di Parma, gestisce il trasporto pubblico di superficie della città e della provincia di Parma. Sorta nel 1975 – ma la sua storia è molto più antica – dalla fusione fra AMPS Trasporti e TEP Provincializzata, che gestivano rispettivamente il servizio di trasporto pubblico urbano e extraurbano, ha percorso un lungo cammino che l'ha vista crescere e trasformarsi per proporsi oggi come azienda del trasporto pubblico e non solo. La Tep S.p.A. ha assunto un ruolo sempre più attivo nella progettazione e gestione del sistema complessivo della mobilità, per fornire adeguate risposte alle diverse esigenze (trasporti, parcheggi ecc.) e per garantire standard elevati di servizio.

Sono stati cercati Open Data sul trasporto pubblico all'interno del sito Web di TEP (www.tep.pr.it/) ma, come per le aziende SETA Spa e START ROMAGNA, non è stato trovato alcun Open Data ed è stato scoperto che i dati posseduti, ma non

pubblicati, sono stati comunicati a Google Maps. Sono state quindi utilizzate le API di Google Maps per reperire i dati sul trasporto pubblico nella provincia di Parma.

2.4.5 Open Data del comune di Bologna

Il portale dedicato agli Open Data del comune di Bologna è disponibile sul sito OpenDataBologna (dati.comune.bologna.it/). All'interno del portale sono pubblicati i dati dell'Amministrazione comunale in formato aperto. Il sito permette ad aziende, associazioni e cittadini di utilizzare e valorizzare i dati dell'Amministrazione comunale, migliorando l'accessibilità delle informazioni e sviluppando nuove applicazioni a beneficio di tutta la comunità. L'apertura delle banche dati pubbliche è uno dei modi per aumentare la trasparenza, l'innovazione e l'efficienza dell'amministrazione pubblica ed è un'opportunità per creare servizi a valore aggiunto per migliori, e più differenziate prestazioni, e una più dinamica crescita economica. Il progetto OpenData del Comune di Bologna prevede che a scadenza regolare vengano resi pubblici nuovi dati aggiornati sul sito web fruibili, tecnicamente aperti e collegabili. Il percorso progettuale OpenData del Comune di Bologna che coinvolge, a livello interdipartimentale, tutta l'Amministrazione intende favorire e stimolare, in collaborazione con la Regione Emilia-Romagna, le imprese e la società civile, lo "sfruttamento intelligente" del patrimonio informativo pubblico. Il sito contiene 460 dataset pubblicati fino a Dicembre 2013.

I dati di particolare interesse per quanto riguarda la mobilità che sono stati analizzati sono:

- **Noleggio biciclette:** dati in formato KML che descrivono le postazioni di bike sharing presenti. Il sistema di riferimento adottato per quanto riguarda la locazione delle postazioni di bike sharing è LAT,LONG;
- **Piste ciclopedonali:** dati in formato SHP che descrivono le piste ciclopedonali presenti. Il sistema di riferimento adottato per quanto riguarda la locazione delle piste ciclopedonali è l'UTM ED50;

- **Parchimetri:** dati in formato SHP che descrivono i parchimetri presenti. Il sistema di riferimento adottato per quanto riguarda la locazione dei parchimetri è il WGS84;
- **Colonnine di ricarica per veicoli elettrici:** dati in formato KML che descrivono le postazioni di ricarica per veicoli elettrici presenti. Il sistema di riferimento adottato per quanto riguarda la locazione delle postazioni di ricarica per veicoli elettrici è LAT,LONG;
- **Parcheggi pubblici:** dati in formato CSV che descrivono i parcheggi pubblici.

Dopo un'attenta analisi dei dati disponibili, si è scelto di prendere in considerazione i dati riguardo le postazioni di bike sharing, le piste ciclopedonali e le colonnine di ricarica per i veicoli elettrici.

Tuttavia sono emerse delle problematiche per quanto riguarda l'analisi delle piste ciclopedonali a causa del non riconoscimento del sistema di riferimento che è stato utilizzato. Tali dati sono presentati attraverso il formato SHP cioè in un formato vettoriale Shapefile ESRI. Tale formato prevede 3 file obbligatori (.shp .shx .dbf in questi file sono presenti esattamente i punti di ogni pista ciclopedonale) e dei file opzionali. Tra i file opzionali è presente un file (.prj) che purtroppo è assolutamente necessario per sapere il sistema di coordinate adottato. Senza questo file non è stato possibile risalire al sistema di riferimento adottato, nonostante tale sistema di riferimento sia enunciato nella pagina Web in cui sono disponibili i dati.

Viene ora presentata una tabella riassuntiva (*figura 9*) che presenta gli Open Data che sono stati trovati per quanto riguarda la mobilità. Dato che nei siti delle aziende SETA, START ROMAGNA e TEP non sono stati trovati Open Data, tali aziende non sono state incluse in questa tabella. Più avanti verrà presentata un'altra tabella riassuntiva dei punti di interesse direttamente presenti in Google Maps, compresi i punti di interesse che sono stati comunicati dalle tra aziende appena citate.

		TPER (BO-FE)	Open Data Bologna
AUTO	Parchimetri	☒	☑
	Colonnine di ricarica per i veicoli elettrici	☒	☑
	Parcheggi	☒	☑
BICI	Piste ciclopedonali	☒	☑
	Noleggio biciclette	☒	☑
MEZZI PUBBLICI	Fermate bus	☑	☒
	Orari di arrivo dei bus	☑	☒
	Percorso delle linee bus	☑	☒
	Postazioni taxi	☒	☑
	Stazioni dei treni	☑	☒

Figura 9: Tabella riassuntiva Open Data disponibili

Capitolo 3

Google Maps e le sue API

L'uso sempre crescente di dispositivi mobili, smartphone, netbook, la loro integrazione con la vita quotidiana, ha aumentato la necessità di servizi e applicazioni che sfruttino in pieno le potenzialità di questi dispositivi, e quindi ha fatto si che la spinta dei fornitori di API fosse molto solida. La quantità di API disponibili per lo sviluppo di applicazioni di vario tipo è impressionante e su tutte Google ha messo a disposizione degli sviluppatori un set completo di strumenti potenti per rilasciare ogni tipo di applicazione, comprese quelle in grado di interfacciarsi con i servizi web based della società di Mountain View. Tra questi servizi, uno di quelli sicuramente più interessante, è il servizio di posizionamento Google Maps, soprattutto per l'uso sempre più spinto dei sistemi GIS[4] (Geographic Information System) e GPS in ambito personal e enterprise. Google Maps consente tramite il semplice inserimento di un indirizzo di una località, di individuare la locazione richiesta su delle mappe disponibili online e che coprono ogni angolo del mondo.

Questo servizio è supportato da una API che garantisce allo sviluppatore la possibilità di integrare nel proprio sito web, o nella propria applicazione, il servizio di localizzazione di Google basato sulle stesse mappe senza dover per questo scrivere lunghe e complesse funzioni di localizzazione a partire dalle coordinate richieste. Infatti non sarà necessario conoscere in dettaglio il database di Google o la logica dei suoi programmi. Fino a prima dell'introduzione di queste API, sviluppare applicazioni web e non che facessero uso di mappe interattive comportava conoscenze specifiche di ambienti GIS.

[4] Un Geographic information system (GIS) è un sistema progettato per catturare, immagazzinare, manipolare, analizzare, gestire e rappresentare dati di tipo geografico.

Gli sviluppatori che intendono utilizzare queste API, trovano sul web uno "spazio", Google Code, dove è possibile reperire tutta la documentazione necessaria per utilizzare tutte le funzioni delle API; queste informazioni aiutano anche l'utente meno esperto in un percorso di formazione che lo porta a padroneggiare in maniera completa ed efficace questi strumenti.

Google Maps fornisce un insieme di servizi che non sono solo mappe, immagini dei satelliti o un ibrido di entrambi, ma consente di eseguire operazioni di zoom, panoramiche, pop-up informativi e tanto altro.

Esistono due tipi di API per poter sfruttare al meglio i servizi offerti da Google Maps:

- **Google Maps JavaScript API v3:** le API sono scritte in JavaScript e quindi "universali", vale a dire utilizzabili da qualunque dispositivo abbia supporto Java integrato. Per utilizzare tali API è necessaria l'installazione del relativo SDK. Tutti i servizi di Google Maps sono messi a disposizione dello sviluppatore attraverso degli oggetti JavaScript che possono essere inseriti facilmente nel codice HTML. La potenza e la semplicità di questo strumento non sono in conflitto, in quanto le operazioni di setup sono tali che le funzioni lavorano in modo completamente trasparente e non invasivo per l'utente, dando l'impressione di utilizzare il servizio originale di Google, ma senza per questo richiedere specifiche competenze di Ajax o di essere a conoscenza dei dettagli implementativi delle API stesse. Per poter utilizzare le API JavaScript di Google Maps è necessario ottenere una "API key". L'API key serve a Google ad identificare l'utilizzatore dell'API e misurare il numero di richieste effettuate dato che vige il limite di 25000 richieste al giorno;
- **Google Maps API Web Services:** è un insieme di interfacce HTTP ai servizi di Google che forniscono dati geografici per le applicazioni mappe. I web services che vengono messi a disposizione sono:
  1) Directions: consente il calcolo del percorso tra due località e la visualizzazione sulla mappa con un disegno polyline in aggiunta a una serie di marker con descrizione testuale;

2) Distance Matrix: dà la possibilità di calcolare la distanza e la durata del percorso impostato, prendendo in considerazione una o piò origini e una o più destinazioni;
3) Elevation: consente di reperire l'altitudine (positiva o negativa) dato un punto sulla mappa;
4) Geocoding: consente la conversione di un indirizzo testuale in coordinate geografiche espresse in latitudine e longitudine e viceversa;
5) Time Zone: consente di ottenere il fuso orario per una certa locazione sulla mappa;
6) Places: consente di trovare determinanti punti di interesse in una zona geografica. Inoltre dà la possibilità di aggiungere/rimuovere nuovi punti di interesse.

L'API che è stata utilizzata nell'app "Route Planner" è Google Maps API Web Services. E' stata una scelta obbligata l'utilizzo di tale API dato che il progetto è stato sviluppato con il framework Titanium. Quindi non è stato possibile installare l'SDK di Google Maps per il sistema operativo iOS.

I servizi che sono stati approfonditi e quindi studiati sono i servizi Directions e Places. Tali servizi verranno ora descritti brevemente.

3.1 Directions API

Google Directions API è un servizio che calcola il percorso tra locazioni usando una richiesta HTTP. E' possibile calcolare i percorsi con differenti mezzi di trasporto tra cui i mezzi pubblici, l'automobile, a piedi o la bicicletta. Le origine, le destinazioni e le possibili tappe intermedie possono essere specificate sia come stringhe di testo (per esempio "Chicago, IL" o "Darwin, NT, Australia") e sia come coordinate geografiche latitudine/longitudine. Le Directions API possono ritornare un insieme di sottopercorsi se si fa uso delle tappe intermedie.

Questo servizio è generalmente progettato per il calcolo dei percorsi tra indirizzi statici (conosciuti in anticipo) per consentire il piazzamento del contenuto su una mappa all'interno di un'applicazione; questo servizio non è progettato per rispondere all'utente in tempo reale dato che il calcolo dei percorsi è un compito molto oneroso dal punto di vista del tempo di processamento e dalle risorse impiegate.

Queste API sono state ampliamente usate nell'algoritmo del calcolo di tutti i possibili percorsi da una certa origine ad una destinazione.

3.1.1 Limiti d'uso

Le Directions API hanno i seguente limiti in vigore:

- 2500 richieste all'interno di un periodo di 24 ore;
- Quando il mezzo di trasporto è l'automobile, la bicicletta o a piedi, ogni directions search conta come una singola richiesta;
- Quando il mezzo di trasporto sono i mezzi pubblici, ogni directions search conta come 4 richieste;
- Quando il mezzo di trasporto è l'automobile, la bicicletta o a piedi, è possibile aggiungere fino a 8 tappe intermedie all'interno della richiesta. Le tappe intermedie non possono essere specificate per il calcolo dei percorsi con i mezzi pubblici.

I client Google Maps for Business hanno limiti più alti:

- 100000 richieste all'interno di un periodo di 24 ore;
- 23 tappe intermedie permesse per ogni richiesta. Le tappe intermedie non sono disponibili per i mezzi pubblici.

Le Directions API possono essere usate solo se i dati restituiti vengono visualizzati all'interno di una mappa Google; l'uso dei dati senza visualizzarli all'interno di una mappa è proibito.

3.1.2 Directions Requests

Una richiesta Directions API presenta la seguente forma:

```
https://maps.googleapis.com/maps/api/directions/output?parameters
```

Vengono ora descritti in modo schematico i parametri necessari a soddisfare la richiesta. Alcuni parametri sono obbligatori, altri sono opzionali. Tutti i parametri devono essere separati dal carattere "e commerciale" (&):

output	L'output può essere specificato in due diversi modi: • *json*: indica l'output nel formato JavaScript Object Notation (JSON); • *xml*: indica l'output in formato XML
origin	L'indirizzo o le coordinate latitudine/longitudine dalle quali si vuole calcolare il percorso. Se si passa un indirizzo come stringa, il servizio Directions geolocalizzerà la stringa e la convertirà nelle coordinate latitudine/longitudine
destination	L'indirizzo o le coordinate latitudine/longitudine verso le quali si vuole calcolare il percorso. Se si passa un indirizzo come stringa, il servizio Directions geolocalizzerà la stringa e la convertirà nelle coordinate latitudine/longitudine;
sensor	indica se la richiesta proviene da un dispositivo che presenta un GPS. Questo valore deve essere *true* o *false*
mode (opzionale)	Specifica il mezzo di trasporto impiegato nel calcolo del percorso. Il valore può essere *driving*

	(default), *walking*, *bicycling* o *transit*. Se si setta il valore a *transit* è obbligatorio specificare anche o il tempo di partenza (*departure_time*) o il tempo di arrivo (*arrival_time*)
waypoints (opzionale)	Specifica un array di tappe intermedie. Una tappa intermedia può essere specificata o come latitudine/longitudine o tramite un indirizzo che sarà poi geolocalizzato. Le tappe intermedie sono supportate solo per i percorsi con la macchina, a piedi e con la bicicletta
alternatives (opzionale)	se settato a *true*, specifica che il servizio Directions può restituire più di un percorso alternativo nella risposta. La restituzione dei percorsi alternativi può incrementare il tempo di risposta del server
avoid (opzionale)	Indica che i percorsi calcolati dovrebbero evitare delle features indicate. Questo parametro supporta i seguenti due argomenti: • *tolls*: indica che il percorso calcolato dovrebbe evitare le strade e i ponti a pedaggio; • *highways*: indica che il percorso calcolato dovrebbe evitare le autostrade
language (opzionale)	Setta la lingua della risposta
units (opzionale)	Specifica il sistema di misura da utilizzare quando vengono visualizzati i risultati
region (opzionale)	Il codice della regione, specificato come ccTLS (top-level domain)
departure_time (opzionale)	Specifica il tempo di partenza in secondi dalla mezzanotte dell'1 Gennaio 1970 UTC. Il tempo di

	partenza può essere specificato in due casi: • Per le *Transit Directions*: uno tra *departure_time* e *arrival_time* deve essere specificato; • Per le *Driving Directions*: i clienti Maps for Business possono specificare il tempo di partenza per ricevere la durata del viaggio considerando le correnti condizioni di traffico. Il tempo di partenza deve essere settato entro pochi minuti dal tempo corrente
arrival_time (opzionale)	Specifica il tempo di arrivo per i percorsi con i mezzi pubblici in secondi dalla mezzanotte dell'1 Gennaio 1970 UTC. Uno tra *arrival_time* e *departure_time* deve essere specificato quando viene richiesto di calcolare il percorso con i mezzi pubblici

3.1.3 Directions Responses

Le risposte sono ritornate nel formato indicato dal flag *output* all'interno dell'URL di richiesta. Nell'applicazione "Route Planner" è stato scelto di farsi restituire la risposta in formato JSON data la grande semplicità nel fare il parsing con il linguaggio JavaScript.

Generalmente viene ritornata solo una entry nell'array "*routes*" anche se il servizio Directions può ritornare diversi percorsi se si passa il parametro *alternatives=true*.

Per estrarre i valori dal risultato, è stato necessario fare il parsing della risposta.

3.1.4 Directions Response Elements

La risposta contiene due elementi di root:

- *status*: contiene dei metadati sulla richiesta;
- *routes*: è un array in cui vengono posti i risultati della richiesta in formato JSON (i percorsi dall'origine alla destinazione). Questo array consiste di *Legs* e *Steps* nidificati. Anche se il servizio non ritorna nessun risultato (come se l'origine e/o la destinazione non esistessero), viene comunque ritornato questo array ma in questo caso non contiene elementi. Ogni elemento di questo array contiene un singolo risultato da un origine ed una destinazione specificati. Questo percorso può essere costituito da una o più *legs* a seconda se sono state specificate delle tappe intermedie. Ogni *legs* infine può contenere uno o più *step* che rappresentano i singoli passi che è necessario fare per raggiungere la destinazione.

  Viene ora dato uno schema riassuntivo dei campi presenti all'interno dell'array *routes*:

summary	Contiene una breve descrizione testuale sul percorso, adatto per disambiguare il percorso principale dalle alternative
legs[]	Contiene un array che contiene informazioni riguardo una tratta intermedia (*leg*) all'interno di un dato percorso. Sarà presente una *leg* per ogni tappa intermedia o destinazione specificati. Un percorso senza tappe intermedie conterrà esattamente una *leg* all'interno dell'array *legs*. Ogni *leg* consiste di una serie di *step*
waypoint_order	Contiene un array che indica l'ordine delle tappe intermedie nel percorso che è stato calcolato. Queste tappe intermedie possono essere riordinate

	se nella richiesta è stato passato il parametro *optimize=true*
overview_polyline	Contiene un oggetto che contiene un array di punti codificati che rappresentano un percorso approssimato (smoothed) del risultato
bounds	Contiene il riquadro di delimitazione di *overview_polyline*
copyrights	Contiene il testo di copyright da visualizzare per questo percorso
warnings[]	Contiene un array di warnings da visualizzare all'utente

Ogni elemento nell'array *legs* specifica una singola leg (tappa) del percorso dall'origine alla destinazione nel percorso calcolato. Per i percorsi che non contengono le tappe intermedie, esisterà una singola *leg*, ma per i percorsi che definiscono una o più tappe intermedie, il percorso consisterà di una o più *legs*.

Viene ora dato uno schema riassuntivo dei campi presenti all'interno dell'array *legs*:

steps[]	Contiene un array di *steps* (passi) che indicano informazioni riguardo ogni *step* della *leg* del percorso
distance	Indica la distanza totale coperta da questa *leg*. Contiene i seguenti elementi: • *value*: indica la distanza totale in metri; • *text*: contiene una rappresentazione human-readable della distanza. Questi campi possono essere assenti se la distanza è sconosciuta. Tale campo è stato utilizzato per conoscere in modo immediato la

	distanza del percorso
duration	Indica la durata totale di questa *leg*. Contiene i seguenti elementi: • *value*: indica la durata in secondi; • *text*: contiene una rappresentazione human-readable della durata. Questi campi possono essere assenti se la durata è sconosciuta. Tale campo è stato utilizzato per conoscere in modo immediato la durata del percorso
duration_in_traffic	Indica la durata totale di questa *leg*, tenendo conto delle condizioni correnti di traffico. Questo campo verrà ritornato solo se si è clienti di Maps for Business
arrival_time	Contiene il tempo stimato di arrivo per questa *leg*. Questa proprietà è ritornata solo per il calcolo dei percorsi con i mezzi pubblici
departure_time	Contiene il tempo stimato di partenza per questa *leg*. Questa proprietà è disponibile solo per il calcolo dei percorsi con i mezzi pubblici
start_location	Contiene le coordinate latitudine/longitudine dell'origine di questa *leg*
end_location	Contiene le coordinate latitudine/longitudine della destinazione di questa *leg*
start_address	Contiene l'indirizzo (tipicamente di una strada) che rispecchia il punto specificato nel campo *start_location*
end_address	Contiene l'indirizzo (tipicamente di una strada) che rispecchia il punto specificato nel campo

	end_location

Ogni elemento nell'array *steps* definisce un singolo step (passo) delle Directions calcolate. Uno step è l'unità più atomica di tutte e descrive un'istruzione specifica, singola del percorso. Per esempio "Turn left at W. 4th St.". Lo step non solo riporta le istruzioni ma contiene anche informazioni sulla distanza e la durata dello step di riferimento.

Per quanto riguarda il calcolo del percorso con i mezzi pubblici, l'array *steps* includerà dei dettagli addizionali all'interno dell'array *transit_details*. Se le Directions includono multipli mezzi di trasporto, saranno fornite indicazioni dettagliate per gli steps "walking" o "driving" all'interno di un *sub_steps* array.

Viene ora dato uno schema riassuntivo dei campi presenti all'interno dell'array *steps*:

html_instructions	Contiene le istruzioni formattate per questo *step*, presentate come una stringa di testo HTML. Tali istruzioni sono state utilizzate per guidare l'utente step by step verso la destinazione finale e sono state inserite nella schermata in cui vengono mostrati all'utente i dettagli di uno specifico percorso
distance	Contiene la distanza coperta da questo *step* fino al prossimo *step*. Questo campo può non essere definito se la distanza è sconosciuta. E' stato utilizzato questo campo per sapere in modo immediato la distanza coperta da questo *step*
duration	Contiene il tempo richiesto per compiere lo *step*, fino al prossimo *step*. Questo campo può non essere definito se la durata è sconosciuta. E' stato utilizzato questo campo per sapere in modo

	immediato la durata di questo *step*
start_location	Contiene il punto di origine di questo *step* espresso nelle coordinate latitudine/longitudine
end_location	Contiene il punto di destinazione di questo *step* espresso nelle coordinate latitudine/longitudine
sub_steps	Contiene le direzioni dettagliate per gli *step* "walking" o "driving". I sottosteps sono disponibili solo quando la proprietà *travel_mode* è settata a "transit"
transit_details	Contiene informazioni specifiche sul mezzo pubblico utilizzato. Questo campo viene ritornato soltanto se la proprietà *travel_mode* è settata a "transit". E' stato fatto ampiamente uso di questo campo

Le transit directions ritornano informazioni aggiuntive che non sono rilevanti per gli altri mezzi di trasporto. Queste proprietà addizionali sono esposte tramite l'oggetto *transit_details*.

Viene ora dato uno schema riassuntivo dei campi presenti all'interno dell'oggetto *transit_details*:

arrival_stop e *departure_stop*	Contengono informazioni riguardo la fermata/stazione di questa parte di percorso. I dettagli delle fermate possono includere: • *name*: il nome della fermata/stazione del mezzo pubblico, per esempio "Union Square"; • *location*: la locazione della fermata/stazione, rappresentata dalle coordinate latitudine/longitudine

	Solo il campo *arrival_stop* è stato utilizzato e visualizzato nella schermata in cui vengono mostrati all'utente i dettagli di uno specifico percorso
arrival_time e *departure_time*	Contengono i tempi di arrivo e partenza per questa *leg* specificati dalle seguenti tre proprietà: • *text*: il tempo espresso in una stringa; • *value*: il tempo passato dalla mezzanotte dell'1 Gennaio 1970 UTC; • *time_zone*: contiene il fuso orario di questa stazione Il campo *departure_time* è stato inserito nella schermata in cui vengono mostrati all'utente i dettagli di uno specifico percorso
headsign	Contiene la stazione del capolinea. Questo campo è stato inserito nella schermata in cui vengono mostrati all'utente i dettagli di uno specifico percorso
headway	Specifica il numero previsto di secondi alla partenza del bus/treno
num_stops	Contiene il numero di fermate, contando la fermata di arrivo ma non la fermata di partenza. Questo campo è stato inserito nella schermata in cui vengono mostrati all'utente i dettagli di uno specifico percorso
line	Contiene informazioni riguardo la linea del mezzo pubblico utilizzato in questo *step* e può includere le seguenti proprietà: • *name*: contiene il nome completo di questa

	linea. Per esempio “7 Avenue Express”; • *short_name*: contiene il nome breve di questa linea. Questo sarà normalmente il numero della linea come per esempio “M7” o “355”; • *color*: contiene il colore comunemente usato nella segnaletica per questa linea. Il colore sarà specificato come una stringa esadecimale come per esempio “#FF0033”; • *agencies*: contiene un array di oggetti TransitAgency dove ogni oggetto fornisce informazioni riguardo il gestore della linea (nome, url, telefono) E’ stato ampliamente usato il campo *short_name* che indica il numero della linea di un certo bus/treno

3.2 Places API

Google Places API è un servizio che ritorna informazioni riguardo i Places (stabilimenti, località geografiche o importanti punti di interesse) usando richieste HTTP. All’interno di una richiesta HTTP viene specificata la locazione tramite le coordinate latitudine/longitudine.

Google Places API definisce le seguenti richieste:

- *Places Searches*: ritorna una lista di Places basata sulla locazione dell’utente o su una search string;
- *Place Details*: ritorna informazioni più dettagliate su uno specifico Place, incluse le recensioni utente;

- *Place Actions*: permette l'integrazione dei dati del Google's Places Database con i dati dalla mia applicazione. E' possibile la schedulazione degli eventi, aggiungere o rimuovere Places, ecc.
- *Place Photos*: dà l'accesso a milioni foto relative ai Places salvate nel database dei Place di Google;
- *Place Autocomplete*: può essere usato per riempire automaticamente il nome e/o l'indirizzo di un Place durante la sua digitazione;
- *Query Autocomplete*: può essere usato per fornire un servizio di predizione di una certa query per ricerche geografiche text-based, ritornando le query suggerite durante la digitazione.

Ognuno dei servizi appena citati è acceduto tramite una richiesta HTTP, e ritorna una risposta in format JSON o XML. Tutte le richieste devono usare il protocollo HTTPS e includere sia il parametro *key* e sia il parametro *sensor*.

Le Google Places API utilizzano un'API key per identificare l'applicazione che origina le richieste HTTP. Le API keys sono gestite attraverso la piattaforma Google APIs Console. E' necessario quindi ottenere una propria API key prima di iniziare ad usare le API.

Per attivare le Places API e creare la propria key, è necessario svolgere i seguenti step:

1) Visitare la piattaforma Google APIs Console al'indirizzo *https://code.google.com/apis/console* ed effettuare il login con le credenziali di Google;
2) Quando ci si logga per la prima volta alle APIs Console, viene creato automaticamente un progetto di default chiamato "API Project". E' possibile usare questo progetto o crearne uno nuovo cliccando sul bottone "API Project" e selezionando "Create";
3) Cliccare il link "Services" nel pannello di sinistra;
4) Cliccare sullo switch (il bottone On/Off) accanto alla voce Places API. Lo switch viene settato quindi a On;
5) Cliccare su "API Access" nel pannello di sinistra;

6) Cliccare su “Create new Server key”;
7) Inserire uno o più indirizzi IP se si desidera limitare I server che possono mandare le richieste HTTP;
8) Cliccare su “Create”. L’API key è stata creata

3.2.1 Place Search

Tramite il servizio Place Search è possibile ricercare i Places per prossimità o attraverso una stringa di testo. Il servizio Place Search ritorna una lista di Places con informazioni di riepilogo su ogni Place ritornato; informazioni aggiuntive sono disponibili tramite il servizio Place Details.

Esistono tre tipi diversi di ricerca di un certo Place:

- *Nearby Search Requests*: permette di ricercare i Places all’interno di una specifica area;
- *Text Search Requests*: permette di ricercare i Places sulla base di una specifica stringa di testo. Vengono ritornati tutti quei Places per i quali fa match la stringa di testo;
- *Radar Search Requests*: permette di ricercare fino a 200 Places in una sola richiesta HTTP ma con meno dettagli rispetto a Nearby Search Requests e Text Search Requests.

Nello sviluppo dell’applicazione “Route Planner” si è fatto un uso intenso delle Nearby Search Requests per la ricerca dei punti di interesse all’interno dell’algoritmo del calcolo dei percorsi. Verranno quindi approfonditi solo questi tipi di richieste.

Come già detto una Nearby Search permette la ricerca dei Places all’interno di un’area specifica. E’ possibile raffinare la ricerca adottando certe keyword o specificando il tipo di Place che si vuole cercare.

Una richiesta Nearby Search presenta la seguente forma:

```
https://maps.googleapis.com/maps/api/place/nearbysearch/output?parameters
```

Vengono ora descritti in modo schematico i parametri necessari a soddisfare la richiesta. Alcuni parametri sono obbligatori, altri sono opzionali. Tutti i parametri devono essere separati dal carattere "e commerciale" (&):

output	L'output può essere specificato in due diversi modi: • *json*: indica l'output nel formato JavaScript Object Notation (JSON); • *xml*: indica l'output in formato XML
key	L'API key dell'applicazione
location	La latitudine/longitudine intorno alle quali si vogliono ritornare i Places
radius	Definisce la distanza (in metri) all'interno della quale si vogliono ritornare i Places. Il raggio massimo permesso è di 50000 metri. Il parametro *radius* non deve essere incluso solo se viene settata la proprietà *rankby=distance*
sensor	Indica se la richiesta proviene da un dispositivo che presenta un GPS. Questo valore deve essere *true* o *false*
keyword (opzionale)	Un termine che farà match con tutti i contenuti che Google ha indicizzato per questo Place tra cui il nome, il tipo, l'indirizzo, le recensioni dei clienti e altri contenuti di terze parti
language (opzionale)	Il codice lingua che indica quale lingua utilizzare nei risultati
minprice e *maxprice* (opzionali)	Limita i risultati solo a quei Places che sono dentro un range specificato. I valori validi sono compresi tra 0 (il più economico) e 4 (il più costoso)
name (opzionale)	Uno o più termini che faranno poi match con i nomi dei Places
opennow (opzionale)	Restituisce solo quei Places che sono aperti nel momento in cui la query viene mandata. I Places che non

	specificano gli orari di apertura non saranno restituiti
rankby (opzionale)	Specifica l'ordine dei risultati. I possibili valori sono: • *prominance* (default): questa opzione ordina i risultati in base alla loro importanza. L'importanza di un certo Place può essere influenzata dal ranking del Place nell'indice di Google, dalla popolarità globale o da altri fattori; • *distance*: questa opzione ordina i risultati in ordine crescente della loro distanza dalla locazione specificata. Se viene specificata questa opzione, verrà settato un raggio di ricerca di 50 km. Inoltre devono essere specificate una o più delle seguenti proprietà: *keyword, name, types*
types (opzionale)	Limita i risultati ai Places che matchano almeno uno dei tipi specificati
pagetoken (opzionale)	Ritorna i prossimi 20 risultati da una precedente ricerca. Se viene settato il *pagetoken*, verrà eseguita una ricerca con gli stessi parametri usati precedentemente

3.2.1.1 Search Responses

La risposta è ritornata nel formato indicato dal flag *output* all'interno della richiesta HTTP. Nell'applicazione "Route Planner" è stato scelto di farsi restituire la risposta in formato JSON data la grande semplicità nel fare il parsing con il linguaggio JavaScript.

Una risposta JSON contiene fino a quattro elementi di root:

status	Contiene dei metadati sulla richiesta
results	Contiene un array di Places. Vengono ritornati fino a 20

	Places per query
html_attributions	Contiene un insieme di attributi che devono essere visualizzati all'utente
next_page_token	Contiene un token che può essere usato per restituire fino a 20 Places aggiuntivi. Questa proprietà non verrà ritornata se sono stati ritornati già tutti i Places. Il numero massimo di Places che possono essere ritornati è 60. Esiste un breve ritardo tra l'emissione del token e quando esso diventerà valido

3.2.1.2 Search Results

I Places che vengono restituiti vengono piazzati all'interno dell'array *results*. Anche se non viene restituito alcun Places, viene ritornato comunque un array *results* vuoto.

Viene ora dato uno schema riassuntivo dei campi presenti all'interno dell'array *results*:

events[]	Array di uno o più elementi <*event*> che forniscono informazioni riguardo gli eventi correnti che stanno avvenendo in questo Place. Vengono ritornati fino a 3 eventi per ogni Place, ordinati dall'ora di inizio. Ogni evento contiene: • *event_id*: un ID unico di questo evento; • *summary*: una descrizione testuale dell'evento; • *url*: un URL in cui vengono forniti più dettagli riguardo questo evento
icon	Contiene un URL di un'icona consigliata che può essere visualizzata all'utente quando indica questo

	risultato
id	Contiene un identificatore unico che denota questo Place
geometry	Contiene informazioni di geometria riguardo questo Place. Generalmente viene inclusa la locazione (latitudine/longitudine) del Place e, opzionalmente, la finestra che identifica la sua zona di copertura. E' stato ampliamente usato questo campo per sapere la locazione esatta di un certo Place
name	Contiene un nome human-readable per il Place ritornato. Per le aziende, questo è usualmente il nome commerciale. E' stato usato questo campo per prelevare il nome completo dei car rental e dei taxi
opening_hours	Può contenere la seguente informazione: *open_now*: è un valore booleano che indica se il Place è in questo momento aperto
photos[]	Un array di oggetti *photo* dove ognuno contiene un riferimento ad un'immagine. Un Place Search ritornerà al massimo una foto mentre una richiesta Place Details può ritornare fino a 10 foto
price_level	Il livello di prezzo di questo Place su una scala che va da 0 a 4. I livelli di prezzo sono interpretati come segue: • *0*: gratuito; • *1*: economico; • *2*: moderato; • *3*: costoso;

	• *4*: molto costoso
rating	Contiene il rating del Place su una scala da 1.0 a 5.0 basata sulle recensioni degli utenti
reference	Contiene un token univoco che può essere usato per recuperare informazioni aggiuntive riguardo questo Place attraverso una richiesta Place Details
types[]	Contiene un array di tipi che descrivono il Place
vicinity	Contiene un nome di una locazione nelle vicinanze. Spesso questo nome si riferisce ad una strada o un quartiere all'interno dei risultati forniti. Questa proprietà viene restituita solo per le richieste Nearby Search
formatted_address	E' una stringa che contiene l'indirizzo human-readable di questo Place. Spesso questo indirizzo è l'equivalente dell'indirizzo postale. Questa proprietà viene restituita solo per le richieste Text Search

3.2.2 Place Details

Una volta che si ha il riferimento (*reference*) da una richiesta Place Search, si possono richiedere più informazioni riguardo una particolare azienda o punto di interesse attraverso la sottomissione di una richiesta Place Details. Una richiesta Place Details ritorna informazioni più specifiche riguardo il Place indicato come per esempio il suo indirizzo completo, il numero di telefono, le recensioni utente, ecc.

E' stato ampiamente usato questo servizio nell'app "Route Planner" per ritornare le coordinate geografiche dei Places estrapolati dalle risposte al servizio Place Autocomplete. I Places ritornati da una richiesta al servizio Place Automplete

infatti non contengono le coordinate geografiche ed è stato quindi necessario interrogare anche il servizio Place Details.

3.2.2.1 Place Details Requests

Una richiesta Place Details è una richiesta HTTP nella seguente forma:

```
https://maps.googleapis.com/maps/api/place/details/output?parame
ters
```

Vengono ora descritti in modo schematico i parametri necessari a soddisfare la richiesta. Alcuni parametri sono obbligatori, altri sono opzionali. Tutti i parametri devono essere separati dal carattere "e commerciale" (&):

output	L'output può essere specificato in due diversi modi: • *json*: indica l'output nel formato JavaScript Object Notation (JSON); • *xml*: indica l'output in formato XML
key	L'API key dell'applicazione
reference	Un identificatore testuale che identifica univocamente un Place
sensor	Indica se la richiesta Place Details viene da un dispositivo che utilizza un sensore di localizzazione (per esempio il GPS). Questo valore deve essere o *true* o *false*
extensions (opzionale)	Indica se la risposta dovrebbe includere campi aggiuntivi. I campi aggiuntivi possono includere Premium data o valori che non sono comunemente richiesti. Le estensioni sono correntemente sperimentali
language (opzionale)	Il codice della lingua che indica in quale linguaggio far ritornare i risultati

3.2.2.2 Place Details Responses

Le risposte Place Details sono ritornate nel formato indicato dal parametro *output* all'interno della richiesta HTTP. Nell'applicazione "Route Planner" è stato scelto di farsi restituire la risposta in formato JSON data la grande semplicità nel fare il parsing con il linguaggio JavaScript.

Una risposta in formato JSON contiene tre elementi di root:

status	Contiene metadati della richiesta
result	Contiene le informazioni dettagliate riguardo il Place richiesto
html_attributions	Contiene un insieme di attributi che devono essere visualizzati all'utente

Viene ora dato uno schema riassuntivo dei campi presenti all'interno del campo *result*:

address_components[]	E' un array di parti di indirizzo. Ogni *address_component* tipicamente contiene: • *types[]*: è un array che indica il tipo del componente indirizzo; • *long_name*: è il nome completo dell'indirizzo; • *short_name*: è un nome abbreviato dell'indirizzo
events[]	E' un array che fornisce informazioni riguardo gli eventi correnti che stanno avvenendo in questo Place. Sono ritornati fino a 10 eventi, ordinati dal tempo di inizio. Ogni evento contiene:

	• *start_time*: il tempo di inizio dell'evento, espresso nel tempo Unix; • *summary*: una descrizione testuale dell'evento; • *url*: un URL in cui sono presenti più dettagli dell'evento
formatted_address	E' una stringa che contiene l'indirizzo human-readable di questo Place
formatted_phone_number	Contiene il numero di telefono del Place nel suo formato locale
geometry	Contiene la seguente informazione: • *location*: contiene la latitudine e la longitudine del Place E' stato usato questo campo per sapere la locazione esatta di un Place recuperato con le Place Autocomplete API
icon	Contiene l'URL dell'icona consigliata che può essere visualizzata all'utente
id	Contiene un identificatore univoco che denota questo Place
international_phone_number	Contiene il numero di telefono del Place nel formato internazionale
name	Contiene il nome human-readable di questo Place
opening_hours	Contiene le seguenti informazioni: • *open_now*: è un valore booleano che indica se il Place è aperto

	all'ora corrente; • *periods[]*: è un array che mostra informazioni riguardo gli orari di apertura del Place nell'arco di una settimana
photos[]	E' un array di foto, ognuna contenente un riferimento ad un'immagine. Una richiesta Place Details può ritornare fino a dieci foto
price_level	Il livello di prezzo del Place su una scala da 0 a 4 dove 0 significa "gratis", 1 significa "non costoso", 2 significa "moderato", 3 significa "costoso", 4 significa "molto costoso"
rating	Contiene il rating del Place, da 1.0 a 5.0, basato sulle recensioni utente
reference	Contiene un token che può essere usato per interrogare il servizio Details in un secondo momento
reviews[]	Un array che contiene fino a cinque recensioni
types[]	Contiene un array di tipi di feature che descrivono il risultato
url	Contiene l'URL di Google ufficiale di questo Place
utc_offset	Contiene il numero di minuti del fuso orario del Place dall'UTC
vicinity	Elenca un indirizzo semplificato per questo Place, includendo la via, il

	numero civico e la località ma non la provincia/stato, il CAP, o la nazione
website	Elenca il sito autoritativo per questo Place (per esempio l'homepage di un'azienda)

3.2.3 Place Actions

Place Actions permette di integrare i dati presenti nel Database dei Places di Google con i dati che provengono da un'applicazione di terze parti. I dati che vengono aggiunti usando Place Actions verranno resi disponibili solo alle applicazioni che condividono la stessa API key. Con Place Actions è possibile aggiungere e rimuovere Places, schedulare eventi, o permettere agli utenti di votare una locazione o un evento.

Verranno ora descritti i servizi che sono stati implementati nell'applicazione "Route Planner" e cioè i servizi di aggiunta e cancellazione di un certo Place. E' stato infatti necessario aggiungere al database di Google Maps i Places relativi alle colonnine di ricarica dei veicoli elettrici e alle postazioni di bike sharing del comune di Bologna. Inoltre, a causa di un errore nell'aggiunta di tali Places, è stato necessario anche una loro cancellazione e un loro reinserimento.

Le richieste Place Report vengono usate per aggiungere nuovi Places o cancellarne degli altri. I nuovi Places saranno immediatamente disponibili per le richieste Place Nearby ed entreranno in una coda di moderazione per essere presi in considerazione in Google Maps. Un Place che è stato appena aggiunto non apparirà tra i risultati di una richiesta Text Search o Radar Search o ad altre applicazioni fino a quando esso sarà stato approvato dal processo di moderazione.

I Places che sono stati aggiunti da un'applicazione di terze parti possono anche essere cancellati, fino alla loro moderazione. Una volta che un Place è stato moderato, esso viene aggiunto tra tutti i risultati del servizio Place Search e non può

più essere cancellato. I Places che non sono stati accettati dal processo di moderazione continueranno ad essere visibili all'applicazione che li ha sottomessi.

3.2.3.1 Aggiunta di un Place

Una richiesta che aggiunge un certo Place, è una richiesta HTTP POST, come quella mostrata di seguito:

```
POST
https://maps.googleapis.com/maps/api/place/add/input_output?paramet
ers HTTP/1.1
Host: maps.googleapis.com
{
  "location": {
    "lat": -33.8669710,
    "lng": 151.1958750
  },
  "accuracy": 50,
  "name": "Google Shoes!",
  "types": ["shoe_store"],
  "language": "en-AU"
}
```

Vengono ora descritti in modo schematico i parametri necessari a soddisfare la richiesta POST URL. Alcuni parametri sono obbligatori, altri sono opzionali. Tutti i parametri devono essere separati dal carattere "e commerciale" (&):

input_output	L'input e l'output possono essere specificati in due diversi modi: • *json*: indica l'input e l'output nel formato JavaScript Object Notation (JSON); • *xml*: indica l'input e l'output in formato XML
sensor	Indica se la richiesta proviene da un dispositivo che presenta un GPS. Questo valore deve essere *true* o *false*
key	L'API key dell'applicazione

Il corpo della richiesta POST contiene informazioni sui Places. Esso dovrebbe essere strutturato secondo l'output specificato (JSON o XML). Viene ora data una descrizione dei campi presenti nel corpo della richiesta POST:

location	La latitudine/longitudine del nuovo Place
accuracy	La precisione della *location* espressa in metri
name	Il nome completo del Place. Esiste il limite di 255 caratteri
types	La categoria appartenente al Place. Anche se questo parametro prende un array, può essere specificato un solo tipo per un certo Place. Se nessun tipo disponibile fa match con questo Place, è possibile specificare *other*
language (opzionale)	La lingua utilizzata all'interno del campo *name*

3.2.3.2 Cancellazione di un Place

Un Place può essere cancellato se:

- Esso è stato aggiunto dalla stessa applicazione che lo vuole cancellare;
- Esso non ha ancora passato con successo il processo di moderazione di Google Maps.

Una richiesta che cancella un certo Place, è una richiesta HTTP POST, come quella mostrata di seguito:

```
POST
https://maps.googleapis.com/maps/api/place/delete/input_output?param
eters HTTP/1.1
Host: maps.googleapis.com

{
  "reference": "place_reference"
}
```

Vengono ora descritti in modo schematico i parametri necessari a soddisfare la richiesta POST URL. Alcuni parametri sono obbligatori, altri sono opzionali. Tutti i parametri devono essere separati dal carattere "e commerciale" (&):

input_output	L'input e l'output possono essere specificati in due diversi modi: • *json*: indica l'input e l'output nel formato JavaScript Object Notation (JSON); • *xml*: indica l'input e l'output in formato XML
sensor	Indica se la richiesta proviene da un dispositivo che presenta un GPS. Questo valore deve essere *true* o *false*
key	L'API key dell'applicazione

Il corpo della richiesta POST contiene informazioni sui Places. Esso dovrebbe essere strutturato secondo l'output specificato (JSON o XML). Viene ora data una descrizione dell'unico campo presente nel corpo della richiesta POST:

reference	L'identificativo testuale che identifica unicamente questo Place

3.2.3.3 Place Report Responses

Le risposte di un Place Report sono ritornate nel formato indicato dal flag *output* all'interno della richiesta POST HTTP. Nell'applicazione "Route Planner" è stato scelto di usare il formato JSON data la sua grande semplicità di manipolazione con il linguaggio JavaScript.

Viene ora descritto l'unico campo che viene ritornato per quanto riguarda una richiesta di cancellazione di un Place:

status	Il codice di stato della risposta. Viene ritornato "OK" se la richiesta è andata a buon fine

Vengono ora descritti i campi che vengono ritornati per quanto riguarda una richiesta di aggiunta di un nuovo Place:

status	Il codice di stato della risposta. Viene ritornato "OK" se la richiesta è andata a buon fine
reference	Un identificatore testuale che identifica univocamente un Place
id	Un identificatore unico che denota questo Place

3.2.4 Place Autocomplete

Il servizio Place Autocomplete è un servizio web che ritorna delle previsioni di Places in risposta ad una richiesta HTTP. La richiesta include una stringa di testo e opzionalmente delle coordinate geografiche.

Questo servizio è stato usato in "Route Planner" per fornire una funzionalità di autocompletamento durante la digitazione dell'utente di una destinazione ritornando indirizzi e località geografiche.

3.2.4.1 Place Autocomplete Requests

Una richiesta Place Autocomplete è una richiesta HTTP nella seguente forma:

```
https://maps.googleapis.com/maps/api/place/autocomplete/output?parameters
```

Vengono ora descritti in modo schematico i parametri necessari a soddisfare la richiesta. Alcuni parametri sono obbligatori, altri sono opzionali. Tutti i parametri devono essere separati dal carattere "e commerciale" (&):

output	L'output può essere specificato in due diversi modi:

	• *json*: indica l'output nel formato JavaScript Object Notation (JSON); • *xml*: indica l'output in formato XML
input	La stringa di testo per la quale si vogliono ricercare i Places
sensor	Indica se il dispositivo che manda la richiesta è provvisto di un sensore di localizzazione (per esempio il GPS). Questo valore vale *true* o *false*
key	L'API key dell'applicazione
offset (opzionale)	La posizione dell'ultimo carattere che il servizio deve usare per il match delle predizioni. Se non è presente l'offset verrà presa in considerazione l'intera stringa fornita nel parametro *input*
location (opzionale)	Il punto intorno al quale si desidera recuperare i possibili Places
radius (opzionale)	La distanza (in metri) all'interno della quale si vogliono ritornare i Places
language (opzionale)	La lingua che si desidera in output
types (opzionale)	I tipi dei Places che si vogliono ritornare
components (opzionale)	Un raggruppamento di luoghi per i quali si vogliono restringere i risultati. Correntemente si può usare questo parametro per filtrare i Places in base alle nazioni

3.2.4.2 Tipi di Place

Si possono filtrare i risultati e volere quindi solo certi tipi di Places passando il parametro *type*. Il parametro specifica un tipo o un insieme di tipi. Se non è specificato nulla, vengono ritornati tutti i tipi di Places. In generale viene permesso solo un singolo tipo. I tipi supportati sono:

- *geocode*: ritorna solo risultati geolocalizzati (indirizzi);
- *establishment*: ritorna solo risultati business (aziende);
- *regions*: ritorna qualsiasi risultato che fa match con i seguenti tipi:
  - *locality;*
  - *sublocality;*
  - *postal_code;*
  - *country;*
  - *administrative_area1;*
  - *administrative_area2*;
- *cities*: ritorna i risultati che fanno match sia con *locality* e sia con *administrative_area3*.

3.2.4.3 Place Autocomplete Responses

La risposta è ritornata nel formato specificato dal parametro *output* all'interno della richiesta HTTP. Nell'applicazione "Route Planner" è stato scelto di farsi restituire la risposta in formato JSON data la grande semplicità nel fare il parsing con il linguaggio JavaScript.

Una risposta in format JSON contiene due elementi di root:

status	Contiene i metadati della richiesta
predictions	Contiene un array di Places con diverse informazioni sui Places ritornati. Vengono ritornati fino a 5 risultati

Quando vengono ritornati dei Places, questi verranno messi all'interno dell'array *predictions*. Anche se il servizio non ritorna alcun risultato, viene ritornato un array *predictions* vuoto.

Viene ora dato uno schema riassuntivo dei campi presenti all'interno dell'array *predictions*:

description	Contiene il nome human-readable per il risultato ritornato. E' stato utilizzato questo campo per recuperare e visualizzare il nome di un certo Place durante la digitazione di un luogo da parte dell'utente
reference	Contiene un token che può essere usato per recuperare informazioni aggiuntive riguardo questo Place in una richiesta Place Details. E' stato utilizzato questo campo per poter recuperare, tramite il servizio Place Details, le coordinate geografiche di questo Place
id	Contiene un identificatore univoco che denota il Place
terms	Contiene un array di termini i quali identificano ogni sezione della descrizione ritornata (una sezione della descrizione è generalmente terminata da una virgola). Ogni entry dell'array ha un campo *value* che contiene il testo del termine, e un campo *offset* che definisce la posizione iniziale di questo termine nella descrizione
types	Contiene un array di tipi appartenenti a questo Place
matched_substring	Contiene un offset ed una lunghezza. Questi due

	valori descrivono la posizione del termine inserito nel testo ritornato

3.3 Places di Google aggiunti in "Route Planner"

Il database dei Places di Google Maps contiene molti punti di interesse che possono essere restituiti attraverso le API appena viste. E' stata fatta un'analisi dei punti di interesse relativi alla mobilità. Tali punti di interesse sono:

- Concessionari;
- Autonoleggio;
- Centro di riparazione auto;
- Autolavaggio;
- Parcheggio;
- Negozio di biciclette;
- Fermate bus;
- Postazione taxi;
- Stazione dei treni.

Tra i punti di interesse che possono essere visualizzati dall'utente sulla mappa presente nell'applicazione "Route Planner", sono stati presi in considerazione i due punti di interesse *autonoleggio* e *postazione taxi*.

I punti di interesse invece che sono stati presi in considerazione nell'algoritmo del calcolo dei percorsi sono *fermata bus*, *stazione dei treni* e *autonoleggio*.

3.4 Places aggiunti in Google Maps

L'applicazione "Route Planner" fa uso di dati che provengono principalmente da due fonti: Google Maps e il portale dedicato agli Open Data del comune di Bologna.

I Places appartenenti al comune di Bologna che sono stati aggiunti nel database di Google Maps sono le *postazioni di bike sharing* e le *colonnine di ricarica per i veicoli elettrici*.

Per quanto riguarda i dati che provengono direttamente da Google Maps, non è stato necessario effettuare alcuna operazione dato che l'estrapolazione di tali dati avviene semplicemente attraverso l'interrogazione del database dei Places di Google Maps con una richiesta HTTP al servizio Nearby Search.

Il discorso invece cambia per quanto riguarda i dati del comune di Bologna. La gestione di tali dati poteva avvenire direttamente nell'applicazione oppure poteva essere demandata direttamente a Google Maps. Un'analisi approfondita del problema ha portato ad adottare questa seconda scelta per rendere lo sviluppatore completamente indifferente riguardo la gestione dei Places. Inoltre non è stato necessario dover creare da zero un database apposito che contenesse tutti i Places dato che Google Maps possiede già un'infrastruttura ottimale per tale scopo.

Capitolo 4

L’applicazione “Route Planner”

La realizzazione dell’applicazione “Route Planner” nasce dall’idea di migliorare la mobilità urbana ed extraurbana delle persone che hanno la necessità di spostarsi all’interno della provincia di Bologna. In particolare l’applicazione vuole aiutare tutte quelle persone che non hanno un mezzo di trasporto proprio ma che desiderano comunque spostarsi in maniera efficace utilizzando i mezzi pubblici, i car rental e le biciclette pubbliche offerte dal comune di Bologna.

L’applicazione consente all’utente di scegliere una destinazione digitando un qualsiasi luogo geografico (città, indirizzo, punto di interesse), un tempo massimo di percorrenza e, grazie ad un algoritmo che verrà spiegato più avanti, l’applicazione calcolerà una lista di possibili percorsi per raggiungere la destinazione data utilizzando uno o più mezzi di trasporto (automobile, bicicletta, mezzi pubblici, a piedi) entro un certo orario.

L’obiettivo principale dell’applicazione è quello di calcolare correttamente tutti i possibili percorsi e di visualizzarli in maniera chiara all’utente. Per ogni percorso calcolato, l’applicazione è in grado di fornire tutte le indicazioni necessarie per raggiungere la destinazione data.

L’intero progetto è stato sviluppato per il sistema operativo iOS 6 dato il bug presente in Titanium riguardo le mancate stampe sulla console con il sistema operativo iOS 7.

4.1 Il progetto

Nelle sezioni seguenti verranno analizzate l'architettura dell'applicazione "Route Planner" e l'algoritmo principale che è stato ideato per il calcolo dei percorsi.

4.1.1 L'architettura dell'applicazione "Route Planner"

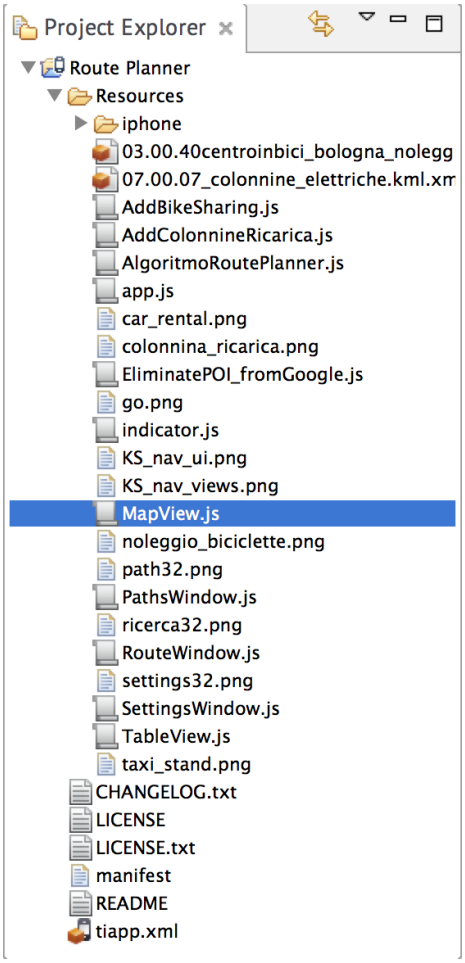

Figura 10: Architettura dell'applicazione

L'applicazione "Route Planner" è stata sviluppata in Titanium, per cui è stato usato il solo linguaggio JavaScript. E' stata ampiamente sfruttata la tecnica di programmazione "a eventi", cioè il flusso dell'applicazione è largamente determinato dal verificarsi di eventi esterni.

Viene ora data una breve descrizione dei file *.js* che sono stati sviluppati per realizzare l'applicazione. Come si può vedere dalla *figura 10*, sono stati implementati ben 11 file *.js*:

1. app.js: questo file è quello che contiene il codice JavaScript che viene lanciato all'avvio dell'applicazione. All'interno di questo file viene inizializzata tutta l'interfaccia grafica dell'app e vengono aggiunti i vari *listener* che verranno poi richiamati in risposta agli eventi esterni come per esempio la digitazione di una destinazione, la scelta di una destinazione o la pressione del tasto "Go";

2. AddBikeSharing.js: in questo file è presente tutto il codice necessario per aggiungere i punti esatti delle postazioni di bike sharing del comune di Bologna nel database di Google Maps;
3. AddColonnineRicarica.js: in questo file è presente tutto il codice necessario per aggiungere i punti esatti delle colonnine di ricarica per i veicoli elettrici del comune di Bologna nel database di Google Maps;
4. AlgoritmoRoutePlanner.js: in questo file è implementato l'intero algoritmo del calcolo dei percorsi;
5. EliminatePOI_fromGoogle.js: in questo file vengono gestite le cancellazioni dei punti di interesse che sono stati aggiunti in Google Maps;
6. indicator.js: in questo file viene gestito l'*Indicator View* che compare immediatamente dopo la pressione del pulsante "Go";
7. MapView.js: in questo file viene gestita la mappa che compare nella schermata iniziale;
8. PathsWindow.js: in questo file viene gestita la *Window* che mostra tutti i possibili percorsi (secondo item della *Tab Bar*);
9. RouteWindow.js: in questo file viene gestita la *Window* che mostra i dettagli di un certo percorso selezionato dall'utente;
10. SettingsWindow: in questo file viene gestita la *Window* che mostra le impostazioni da settare all'utente (terzo item della *Tab Bar*);
11. TableView.js: in questo file viene gestita la tabella che consente di mostrare o nascondere i punti di interesse (colonnine di ricarica per i veicoli elettrici, postazioni di bike sharing, car rental e taxi) sulla mappa della schermata iniziale.

All'interno del progetto sono presenti ulteriori file di supporto:

- 03.00.40centroinbici_bologna_noleggio_biciclette.kml.xml: è il file in formato XML che descrive la posizione esatta delle postazioni di bike sharing. Questo file è stato utilizzato una sola volta dato che le posizioni delle postazioni di bike sharing sono state aggiunte in Google Maps una sola volta;

- 07.00.07_colonnine_elettriche.kml.xml: è il file in formato XML che descrive la posizione esatta delle colonnine di ricarica per i veicoli elettrici. Questo file è stato utilizzato una sola volta dato che le posizioni delle colonnine sono state aggiunte in Google Maps una sola volta;
- car_rental.png: è l'icona che viene visualizzata sulla mappa per mostrare i car rental;
- colonnina_ricarica.png: è l'icona che viene visualizzata sulla mappa per mostrare le colonnine di ricarica per i veicoli elettrici;
- go.png: è l'icona del pulsante che viene mostrato all'interno di un'annotazione. Se premuto, parte l'algoritmo del calcolo dei percorsi;
- KS_nav_ui.png e KS_nav_views.png: file non utilizzati presenti alla creazione del progetto;
- noleggio_biciclette.png: è l'icona che viene visualizzata sulla mappa per mostrare le postazioni di bike sharing;
- path32.png: è l'icona che viene mostrata nel secondo item della *Tab Bar*;
- ricerca32.png: è l'icona che viene mostrata nel primo item della *Tab Bar*;
- settings32.png: è l'icona che viene mostrata nel terzo item della *Tab Bar*;
- taxi_stand.png: è l'icona che viene visualizzata sulla mappa per mostrare le postazioni dei taxi;
- tiapp.xml: è il file di configurazione dei parametri iniziali dell'applicazione.

4.1.2 L’algoritmo utilizzato per il calcolo dei percorsi

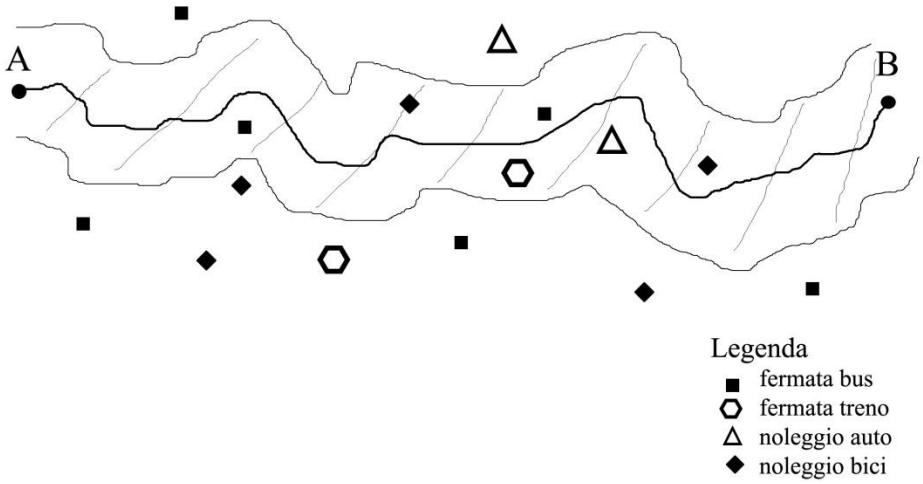

Figura 11: Possibile tragitto dal punto A al punto B

L’obiettivo principale dell’algoritmo che è stato sviluppato è quello di trovare il percorso migliore, cioè più breve in termini temporali, dal punto esatto in cui si trova l’utente (punto A della *figura 11*) ad una destinazione selezionata (punto B della *figura 11*) utilizzando una combinazione di mezzi di trasporto.

Viene ora data una descrizione step by step dell’algoritmo che è stato realizzato tramite l’esempio, mostrato in *figura 11*, di un percorso da A a B. In tale figura vengono mostrati anche i punti di interesse fermata bus, fermata treno, noleggio auto e noleggio bici presenti intorno al tracciato.

Si suppone quindi di dover partire dal punto A e di dover arrivare nel punto B.

1. Viene fatto il calcolo del percorso a piedi da A a B in modo tale da avere un’idea del tracciato che bisogna fare; dopo questo step infatti si conoscono tutti i punti (latitudine e longitudine) del percorso;
2. Calcolo di un intervallo proporzionale alla lunghezza del percorso in modo tale da prendere in considerazione tutti i punti di interesse vicini al percorso. In particolare viene considerato un intervallo pari a $1/100$ della lunghezza complessiva del percorso;

3. Ordinare in ordine crescente i punti di interesse trovati in base alla distanza dall'origine (dal più vicino al più lontano ad A). Nell'esempio considerato si ha $n = 6$ punti di interesse trovati. Tali punti di interesse verranno ordinati nel seguente modo:
   a) fermata_bus1 (fb1);
   b) noleggio_bici1 (nb1);
   c) fermata_treno (ft);
   d) fermata_bus2 (fb2);
   e) noleggio_auto (na);
   f) noleggio_bici2 (nb2)
4. Calcolo delle combinazioni semplici $C_{n,1}, C_{n,2}, \dots, C_{n,n}$ dei punti di interesse che si devono raggiungere. Tali combinazioni rappresenteranno dei possibili percorsi. Nell'esempio considerato è necessario calcolare le combinazioni semplici $C_{6,1}, C_{6,2}, C_{6,3}, C_{6,4}, C_{6,5}, C_{6,6}$:

$C_{6,1} = (fb1), (nb1), (ft), (fb2), (na), (nb2);$

$C_{6,2} = (fb1, nb1), (fb1, ft), (fb1, fb2), (fb1, na), (fb1, nb2), (nb1, ft),$
$(nb1, fb2), (nb1, na), (nb1, nb2), (ft, fb2), (ft, na), (ft, nb2),$
$(fb2, na), (fb2, nb2), (na, nb2);$

$C_{6,3} = (fb1, nb1, ft), (fb1, nb1, fb2), (fb1, nb1, na), (fb1, nb1, nb2),$
$(fb1, ft, fb2), (fb1, ft, na), (fb1, ft, nb2), (fb1, fb2, na),$
$(fb1, fb2, nb2), (fb1, na, nb2), (nb1, ft, fb2), (nb1, ft, na),$
$(nb1, ft, nb2), (nb1, fb2, na), (nb1, fb2, nb2), (nb1, na, nb2),$
$(ft, fb2, na), (ft, fb2, nb2), (ft, na, nb2), (fb2, na, nb2);$

$C_{6,4} = (fb1, nb1, ft, fb2), (fb1, nb1, ft, na), (fb1, nb1, ft, nb2),$
$(fb1, nb1, fb2, na), (fb1, nb1, fb2, nb2), (fb1, nb1, na, nb2),$
$(fb1, ft, fb2, na), (fb1, ft, fb2, nb2), (fb1, ft, na, nb2),$
$(fb1, fb2, na, nb2), (nb1, ft, fb2, na), (nb1, ft, fb2, nb2),$
$(nb1, ft, na, nb2), (nb1, fb2, na, nb2), (ft, fb2, na, nb2);$

$C_{6,5} = (fb1, nb1, ft, fb2, na), (fb1, nb1, ft, fb2, nb2),$

$(fb1, nb1, ft, na, nb2), (fb1, nb1, fb2, na, nb2),$

$(fb1, ft, fb2, na, nb2), (nb1, ft, fb2, na, nb2);$

$C_{6,6} = (fb1, nb1, ft, fb2, na, nb2);$

5. Applicazione di un filtro alle combinazioni calcolate al passo precedente: eliminare tutte quelle combinazioni che presentano due o più mezzi di trasporto uguali adiacenti (il mezzo di trasporto "treno" è considerato uguale al mezzo di trasporto "bus" dato che quando si richiede di calcolare un percorso al servizio Directions con i mezzi pubblici, vengono presi in considerazione sia i bus e sia i treni). Viene ora mostrato un esempio del perché è conveniente applicare tale filtro: suppongo di avere la combinazione A-B-C-D-E dove A è l'origine, E la destinazione, B e C due fermate dell'autobus e D una postazione di car rental. Diventa conveniente eliminare questa combinazione perché non c'è alcun motivo di dover per forza passare per le fermate B e C. Con questa combinazione diventa necessario chiedere al servizio Directions di Google il calcolo del percorso da A a B a piedi, da B a C con i mezzi pubblici, da C a D con i mezzi pubblici e da D ad E con la macchina. E' possibile rilassare il vincolo di dover passare per forza per C, cioè è meglio far calcolare a Google il percorso con i mezzi pubblici direttamente da B a D senza passare per C (combinazione A-B-D-E) dato che per questa combinazione si dovrebbe andare da B a D sempre con i mezzi pubblici. Poi le linee di bus che è necessario prendere e le posizioni precise delle fermate dove prendere i bus vengono restituite da Google (Google potrebbe restituire un percorso che passa per la fermata C ma potrebbe anche non farlo se non conviene passare per C per arrivare in D). Quindi è conveniente eliminare tale combinazione (A-B-C-D-E) perché tanto le combinazioni A-B-D-E e A-C-D-E esisterebbero e non verrebbero eliminate dai filtri. Nell'esempio considerato verrebbero eliminate le combinazioni $(fb1, fb2), (nb1, nb2), (fb1, fb2, nb2)$, ... e così via;
6. Aggiungere a tutte le combinazioni rimaste l'origine A e la destinazione B. Nell'esempio considerato le combinazioni diventerebbero $(A, fb1, B), (A, nb1, B)$, ... e così via;

7. Aggiungere tra i possibili percorsi quelli che utilizzano un solo mezzo di trasporto dal punto A al punto B senza passare per altri punti di interesse. Si hanno solo due possibili percorsi dato che è possibile andare da A a B soltanto a piedi e con i mezzi pubblici (si suppone sempre che l'utente non possiede né la macchina e né la bici). Si suppone di avere m combinazioni rimaste. Vengono calcolati quindi $m + 2$ percorsi;
8. Durante il calcolo dei possibili percorsi vengono scartati passo passo quelli che non rispettano il vincolo di arrivo a destinazione entro un certo orario. Inoltre è stata implementata un ulteriore ottimizzazione: dato che molti sottopercorsi che è necessario calcolare sono in comune, essi vengono calcolati una sola volta;
9. I percorsi che rispettano i vincoli vengono ordinati in ordine crescente nei tempi di percorrenza.

4.1.2.1 Considerazioni generali

- Nel primo tratto (dal punto A al primo punto di interesse) è possibile utilizzare solo i mezzi pubblici o andare a piedi;
- Durante tutto il tragitto si cerca di avvicinarsi sempre di più alla destinazione B. Non capita mai di raggiungere un punto di interesse che sta indietro a dove ci si trova;
- Se in una certa tratta si accumula del ritardo (per esempio c'è tanto traffico) non verranno più rispettati tutti i tempi delle tratte successive (si potrebbe non riuscire a prendere più per esempio un certo bus in una tratta successiva);
- Si suppone che una volta presa la bicicletta in una postazione di bike sharing, è possibile lasciarla ovunque (basta legarla con un lucchetto in un qualsiasi palo). Stesso discorso vale per la macchina (semplificazione);
- Dato che non è possibile richiedere al servizio Directions di Google un possibile percorso con la bicicletta, si è deciso di calcolare il sottopercorso in

questione a piedi e poi calcolare a parte il tempo di percorrenza tenendo conto di una velocità media di 20 km/h;

- Vengono aggiunti dei ritardi ai tempi di percorrenza dei sottopercorsi in bicicletta e in macchina rispettivamente di 5 e 20 minuti. Si suppone infatti che prendere la bicicletta costa all'utente in media 5 minuti mentre prendere la macchina costa all'utente in media 20 minuti (ci potrebbe essere fila, stipulazione del contratto, ecc.). Si è deciso di non rendere configurabili tali valori poiché l'utente non sa minimamente quanto dovrà aspettare e inoltre sono tempi molto variabili cioè è possibile trovare la fila in una determinata fascia oraria oppure non trovarla in un'altra fascia oraria;
- Laddove all'interno di una tratta completa non sia possibile trovare anche un solo sottopercorso (il servizio Directions non restituisce nulla), si elimina semplicemente tale percorso dai possibili percorsi;
- Se si richiede al servizio Directions di Google di calcolare un percorso con i mezzi pubblici, è molto probabile che venga restituito un percorso in cui sia presente anche il mezzo di trasporto "a piedi".

4.1.2.2 Considerazioni tecniche ed implementative

- Più è lungo il percorso calcolato nello step 1, più grande sarà l'intervallo considerato nello step 2 e più punti di interesse verranno presi in considerazione. Con tanti punti di interesse presi in considerazione verranno svolti più calcoli negli step successivi, verranno fatte più richieste al servizio Google Directions con il conseguente rischio di non poter calcolare tutti i possibili percorsi per via del limite imposto da Google di massimo 2500 richieste al giorno. Un possibile sviluppo futuro potrebbe essere quello di filtrare, ove necessario, i punti di interesse considerando solo quelli più rilevanti in modo tale da essere certi di poter soddisfare a pieno tutte le richieste al servizio Google Directions. Un altro possibile sviluppo futuro

potrebbe essere, nel caso in cui venga sforato il limite delle 2500 richieste giornaliere, quello di mostrare comunque tutti i percorsi calcolati fino al raggiungimento della soglia all'utente;

- E' stato deciso di effettuare una richiesta al servizio Directions di Google ogni 400 ms. E' stato scelto questo valore poiché non si è a conoscenza del massimo rate consentito di richieste al secondo per i clienti non business e inoltre un valore più basso porterebbe ad una percentuale molto alta di risposte `OVER_QUERY_LIMIT` mentre un valore più alto porterebbe a volte a ritardi nel reperimento della risposta;
- Durante il calcolo di tutti i possibili percorsi, vengono calcolati prima tutti i percorsi effettuati a piedi, con la macchina e in bicicletta dato che i tempi di percorrenza e i tracciati non sono influenzati dall'orario. Discorso diverso vale per i mezzi pubblici (bus e treno) poiché tali percorsi sono influenzati dall'orario di arrivo dell'utente alla fermata del bus o alla stazione dei treni. Tali percorsi infatti sono gli ultimi che vengono calcolati;
- Dato che vengono calcolati numerosi sottopercorsi, e dato che esiste un limite abbastanza restrittivo riguardo il numero di richieste massime giornaliere al servizio Directions (2500) si è voluto ottimizzare il numero di tali richieste. Durante l'esecuzione dell'algoritmo infatti, se il calcolo di un certo sottopercorso è stato già richiesto al servizio Directions, esso non sarà più richiesto ma verrà preso in considerazione per i futuri calcoli di quel medesimo sottopercorso.

4.1.2.3 Activity Diagram

In *figura 12* viene mostrato l'activity diagram relativo all'insieme di azioni che vengono svolte quando inizia l'algoritmo del calcolo dei percorsi.

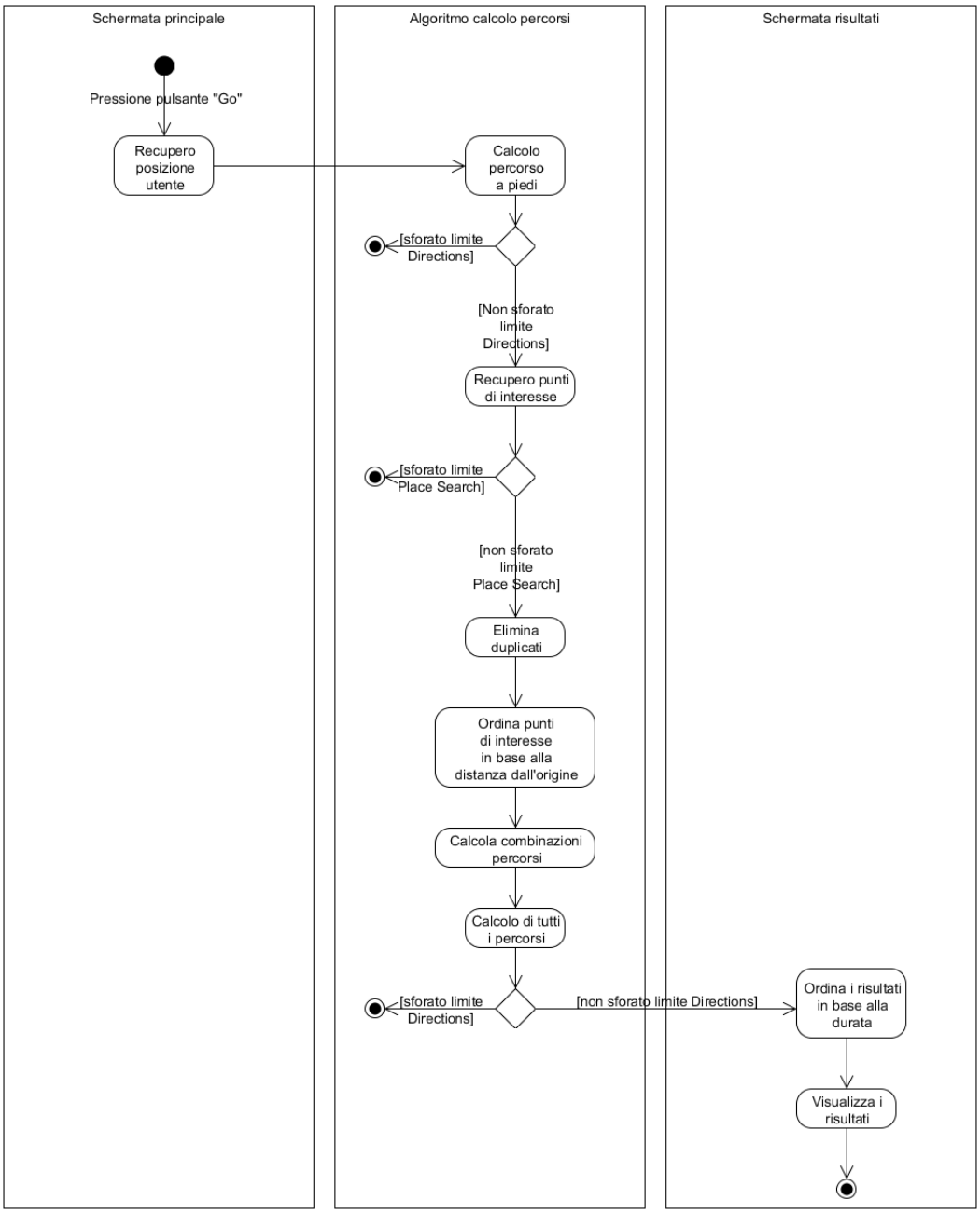

Figura 12: Activity diagram algoritmo

4.2 L’implementazione

Nelle sezioni seguenti verranno analizzate le funzionalità e l’interfaccia dell’applicazione e, infine, verrà fornito un suo esempio di utilizzo.

4.2.1 Funzionalità ed interfaccia

Figura 13: Schermata principale

All'avvio l'applicazione presenta una *Window* in cui è presente una *Tab Bar* con tre item dove il primo item mostra la finestra di ricerca principale, il secondo item i risultati di una ricerca e il terzo item le impostazioni utente.

4.2.1.1 Finestra di ricerca principale

Tale finestra presenta una *Navigation Bar* con un bottone "POI" sulla sinistra, un campo di testo su cui l'utente potrà digitare la destinazione che vuole raggiungere ed una mappa sulla quale vengono visualizzati i punti di interesse desiderati e l'eventuale annotazione che rappresenta la destinazione finale (*figura 13*).

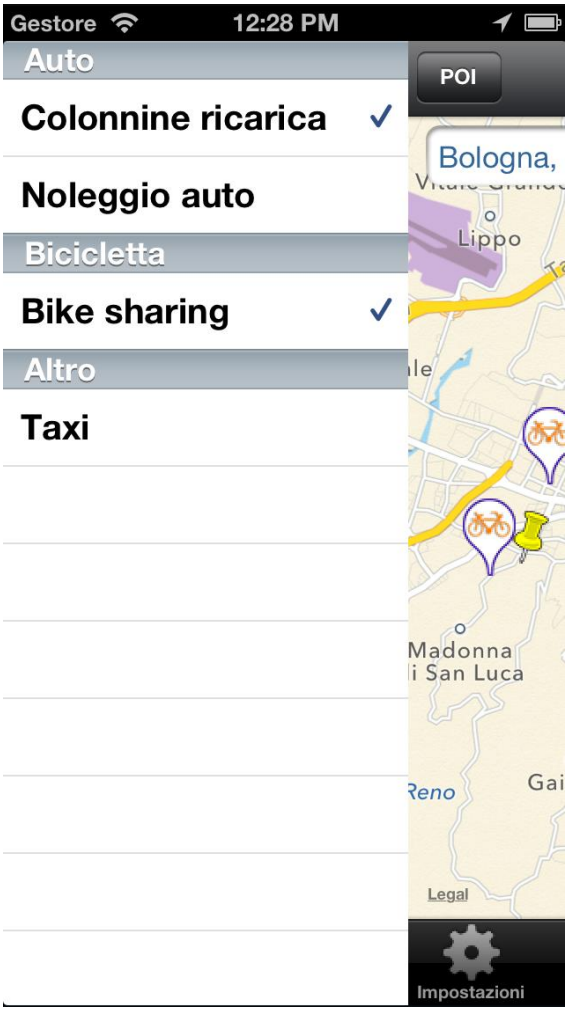

Figura 14: Schermata selezione POI

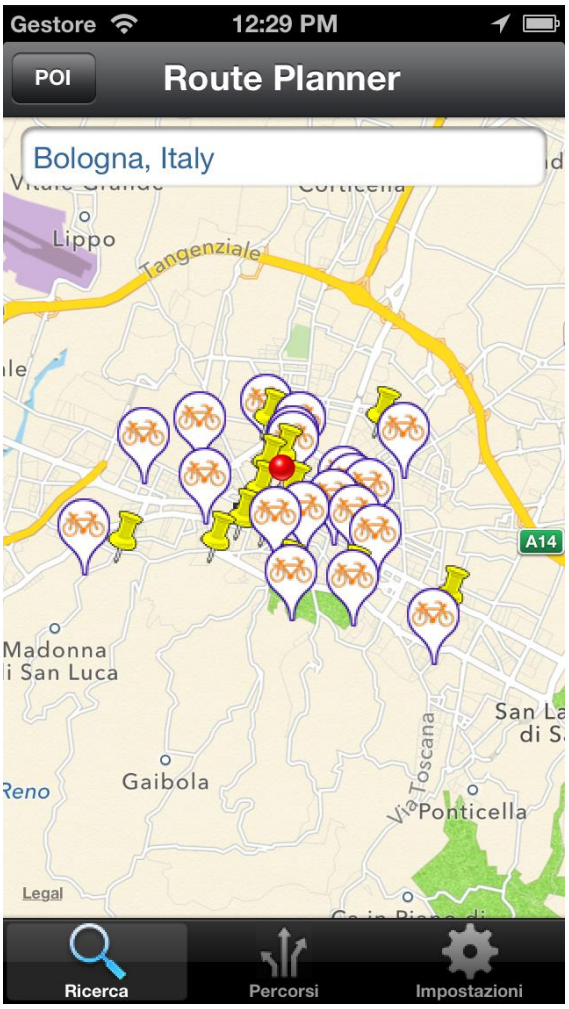

Figura 15: Schermata visualizzazione POI

Vengono ora descritte le funzionalità presenti all'interno di questa *Window*:

Funzionalità	Visualizzazione/nascondimento della tabella sulla sinistra relativa ai punti di interesse da visualizzare/nascondere
Interfaccia	Bottone "POI" presente nella *Navigation Bar*
Input	Pressione del bottone "POI"
Processo	• Se non è ancora visualizzata la tabella sulla sinistra, viene costruita la tabella dei POI ed aggiunta all'interno della *Window* principale attraverso il metodo *add*; • Se è già visualizzata la tabella sulla sinistra, viene distrutta la tabella dei POI attraverso il metodo *remove* della *Window* principale.
Output	• Se non è ancora visualizzata la tabella sulla sinistra, viene effettuata un'animazione che sposta la mappa verso destra e viene visualizzata la tabella dei POI sulla sinistra (*figura 14*); • Se è già visualizzata la tabella sulla sinistra, viene effettuata un'animazione che sposta la mappa verso sinistra nascondendo quindi la tabella dei POI

Funzionalità	Visualizzazione/nascondimento delle locazioni delle colonnine di ricarica dei veicoli elettrici, degli autonoleggio, delle postazioni di bike sharing e delle postazioni dei taxi
Interfaccia	Tabella dove vengono riportati gli item "Colonnine ricarica", "Noleggio auto", "Bike sharing" e "Taxi"
Input	Pressione di un qualsiasi item della tabella dei POI

Processo	• Se l'item non è attivo e viene premuto, vengono recuperati attraverso il servizio Place Search i punti di interesse relativi all'item che è stato premuto e viene chiamato il metodo *addAnnotation* per ogni punto di interesse recuperato; • Se l'item è già attivo, vengono recuperate tutte le annotazioni visualizzate sulla mappa e viene chiamato il metodo *removeAnnotation* solo per le annotazioni relative all'item che è stato premuto.
Output	• Se è stato attivato un certo item, vengono visualizzate sulla mappa le annotazioni relative all'item che è stato premuto (*figura 15*); • Se è stato disattivato un certo item, vengono eliminate dalla mappa le annotazioni relative all'item che è stato premuto.

Funzionalità	Visualizzazione dettagli di un punto di interesse
Interfaccia	Annotazione presente sulla mappa
Input	Pressione sull'icona di una qualsiasi annotazione
Processo	Gestito tutto dal framework Titanium
Output	Viene visualizzata una *View* accanto all'annotazione che riporta il pulsante "Go" e dei dettagli del punto di interesse premuto

Funzionalità	Visualizzazione di una lista di possibili destinazioni
Interfaccia	Il campo di testo presente nella parte alta della mappa
Input	Digitazione o cancellazione di una destinazione
Processo	Dal secondo carattere digitato in poi, ad ogni

	inserimento o cancellazione di un carattere viene richiamato il servizio Place Autocomplete con la stringa che è correntemente presente nel campo di testo
Output	Viene visualizzata una tabella di massimo cinque righe con tutte le possibili destinazioni immediatamente sotto il campo di testo

Funzionalità	Selezione della destinazione desiderata dalla tabella delle possibili destinazioni
Interfaccia	Tabella delle possibili destinazioni
Input	Pressione su una qualsiasi riga della tabella
Processo	Viene creata una nuova annotazione di colore rosso con la destinazione selezionata ed aggiunta alla mappa attraverso il metodo *addAnnotation*
Output	Viene visualizzata l'annotazione sulla mappa

Funzionalità	Calcolo di tutti i possibili percorsi alla destinazione data
Interfaccia	Bottone "Go" presente per una certa annotazione
Input	Pressione del bottone "Go"
Processo	Viene eseguito l'algoritmo spiegato nel sotto capitolo 4.1.2
Output	Viene aperta una nuova *Window* (*figura 16*) associata al secondo item della *Tab Bar* nella quale vengono visualizzati tutti i possibili percorsi

Figura 16: Schermata visualizzazione risultati

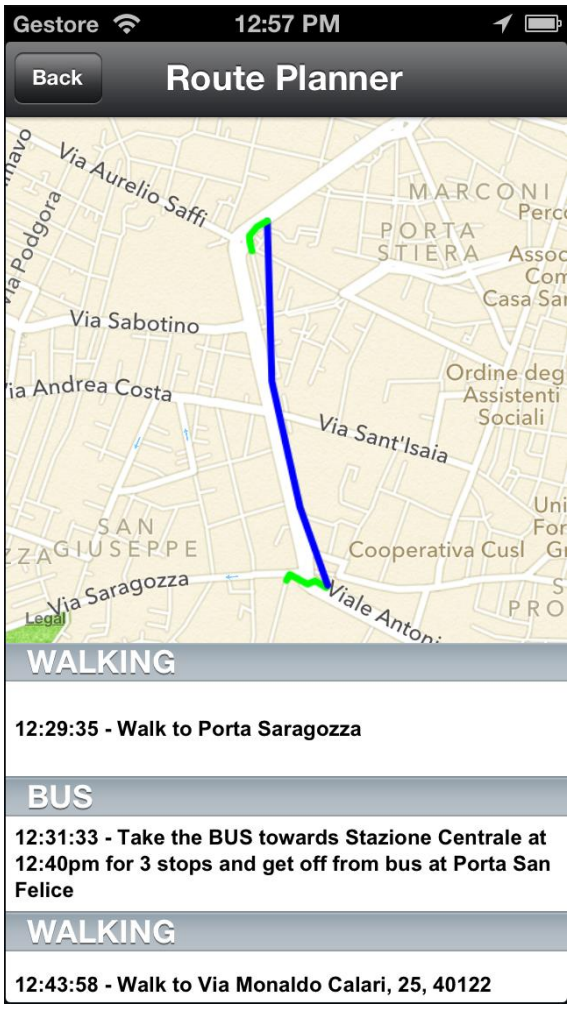

Figura 17: Schermata dettagli percorso

4.2.1.2 Finestra visualizzazione percorsi

In questa seconda *Window* viene mostrata una tabella dove per ogni riga vengono mostrate delle statistiche riguardo un possibile tragitto. In dettaglio sono visualizzate le percentuali di utilizzo in base alla distanza dei quattro mezzi di trasporto (automobile, mezzi pubblici, bicicletta, a piedi), i km totali effettuati e il tempo di percorrenza totale. I risultati sono ordinati in ordine crescente secondo i tempi di percorrenza per cui i risultati migliori (minor tempo di percorrenza)

verranno visualizzati nelle prime posizioni, mentre quelli peggiori nelle ultime posizioni. Si è voluta mostrare la statistica riguardo le percentuali di utilizzo dei mezzi di trasporto proprio per dare un'idea chiara all'utente di quali mezzi di trasporto utilizzerà per raggiungere la destinazione desiderata. Come sviluppo futuro dell'app, per rendere ancora più chiaro il tragitto che l'utente dovrà percorrere, si potrebbe aggiungere un altro parametro riguardante il numero di cambi di mezzi di trasporto che l'utente dovrà effettuare.

Vengono ora descritte le funzionalità presenti all'interno di questa *Window*:

Funzionalità	Visualizzazione dettagli di un certo percorso
Interfaccia	Tabella dei possibili percorsi
Input	Pressione su una qualsiasi riga della tabella
Processo	Vengono manipolate tutte le informazioni che si hanno sul percorso selezionato e viene costruita una nuova *Window*
Output	E' visualizzata la *Window* (*figura 17*) con all'interno una mappa che visualizza il percorso da seguire e una tabella all'interno della quale viene descritto ogni singolo step da compiere. In dettaglio il tragitto sulla mappa è colorato con diversi colori che rappresentano il mezzo di trasporto utilizzato per una specifica tratta: • Verde: la tratta è percorsa a piedi; • Blu: la tratta è percorsa con i mezzi pubblici (eccetto il treno); • Marrone: la tratta è percorsa con il treno; • Arancione: la tratta è percorsa con la macchina; • Rosso: la tratta è percorsa in bicicletta.

Viene ora descritta una funzionalità relativa alla *Window* dei dettagli di un certo percorso:

Funzionalità	Visualizzazione sulla mappa del punto preciso di uno step
Interfaccia	Tabella che descrive in dettaglio ogni singolo step da compiere
Input	Pressione su una qualsiasi riga della tabella
Processo	Vengono recuperate le informazioni di localizzazione dello step selezionato
Output	Viene visualizzata un'annotazione di colore verde nel punto in cui occorre lo step selezionato

4.2.1.3 Finestra impostazioni utente

Figura 14: Schermata impostazioni

La *Window* associata al terzo item della *Tab Bar* (*figura 18*) mostra una tabella in cui è possibile settare un orario massimo di percorrenza.

Vengono ora descritte le funzionalità presenti all'interno di questa *Window*:

Funzionalità	Visualizzazione di un *Picker View*
Interfaccia	L'unica riga presente all'interno della tabella
Input	Pressione sull'unica riga presente nella tabella
Processo	Costruzione del *Picker View* da visualizzare e costruzione di una *View* con i bottoni "Cancel" e "Done"
Output	Visualizzazione del *Picker View* e della *View* che sono stati creati

Funzionalità	Attivazione filtro del massimo numero di ore consentite
Interfaccia	Il *Picker View* e la *View* visualizzate
Input	Selezione di un valore intero da 1 a 10 ore ed infine pressione del pulsante "Done"
Processo	Vengono eliminati tutti i risultati che non rispettano il vincolo del numero massimo di ore consentito
Output	Viene aggiornato il valore presente nella riga della tabella con il nuovo valore selezionato dall'utente

4.2.2 Esempio di calcolo di un percorso

Viene ora mostrato un esempio di calcolo dei percorsi tra il DEI (Ingegneria dell'Energia Elettrica e dell'Informazione) dell'università di Bologna (Viale del Risorgimento, 2) e la sede centrale dell'università di Bologna (Via Zamboni, 33).

Per calcolare tale percorso è necessario stare fisicamente nel punto di origine, quindi, in questo caso è necessario trovarsi esattamente in Viale del Risorgimento, 2.

E’ necessario poi immettere la via di destinazione nella schermata principale. Durante la digitazione della via verranno mostrate in una tabella tutte le possibili destinazioni. Infine l’utente dovrà premere nella riga della tabella che mostra la destinazione dove effettivamente vuole andare. Nell’esempio tale destinazione è “Via Zamboni, 33 Bologna”. Una volta selezionata la destinazione, e premuto sull’annotazione comparsa, la schermata si presenterà come in *figura 19*. A questo punto sarà necessario premere il pulsante “Go” per avviare l’algoritmo di ricerca di tutti i possibili percorsi da dove si trova correntemente l’utente (Viale del Risorgimento, 2) fino a Via Zamboni, 33.

N

Figura 19: Schermata principale con annotazione

Figura 20: Schermata dei risultati

Nel caso specifico l'algoritmo ci mette 44 secondi per trovare 17 possibili percorsi. Una volta finito l'algoritmo di calcolo dei percorsi, verrà effettuato uno switch alla seconda schermata, quella relativa alla visualizzazione dei percorsi, *figura 20*.

A questo punto l'utente ha la possibilità di scegliere uno qualsiasi dei possibili percorsi visualizzati tenendo conto delle percentuali di utilizzo di tutti i mezzi di trasporto (automobile, mezzi pubblici, bicicletta, a piedi). I risultati sono ordinati in ordine crescente secondo i tempi di percorrenza. Selezionando per esempio la prima scelta dalla tabella, si aprirà una nuova schermata (*figura 21*) che visualizzerà più in dettaglio il percorso che deve essere effettuato. Il percorso è colorato con differenti

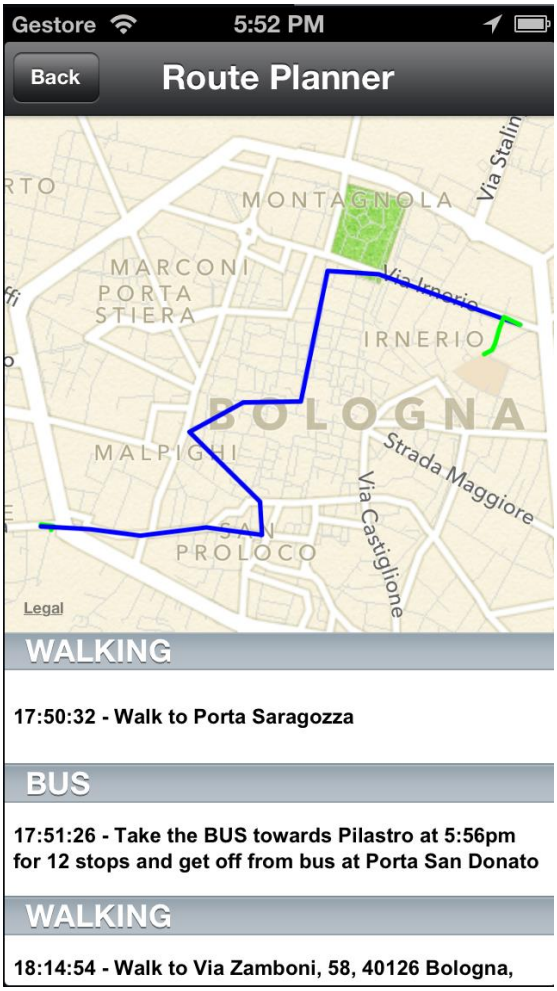

Figura 21: Schermata dettagli primo percorso

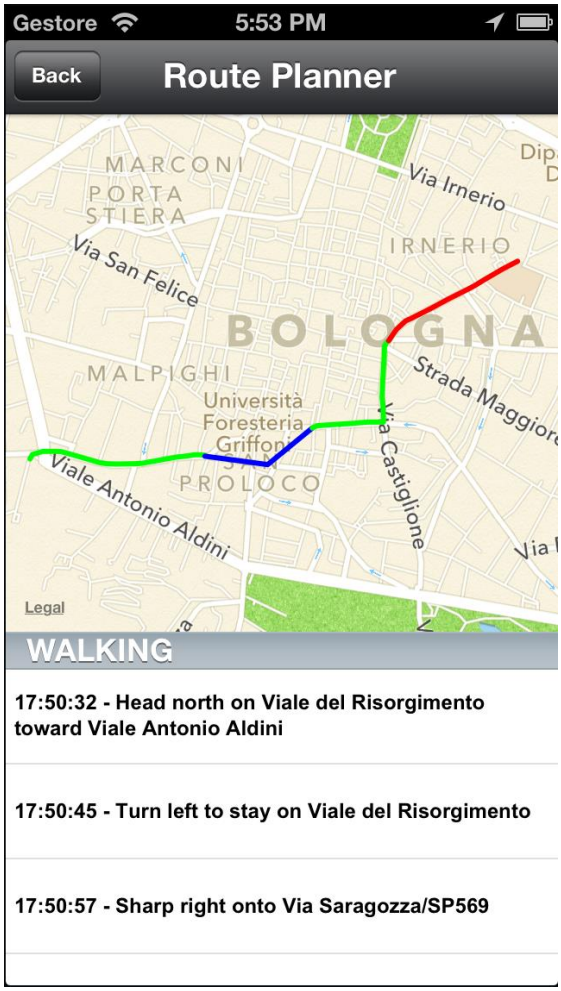

Figura 22: Schermata dettagli secondo percorso

colori per mostrare quale mezzo di trasporto viene utilizzato in una data tratta. In *figura 21* gli unici due mezzi di trasporto che vengono utilizzati sono i mezzi pubblici (blu) e a piedi (verde).

Per maggiori dettagli l'utente può premere su ogni riga dove sono scritte le indicazioni precise e verrà mostrato più in dettaglio, tramite un'annotazione, il punto preciso in cui deve essere effettuato un cambio di direzione. L'utente in questo modo potrà seguire passo passo tutte le indicazioni descritte in maniera semplice ed immediata.

Viene mostrato in *figura 22* un secondo possibile percorso. In tale percorso si può notare come vengano usati tre mezzi di trasporto: mezzi pubblici (blu), bicicletta

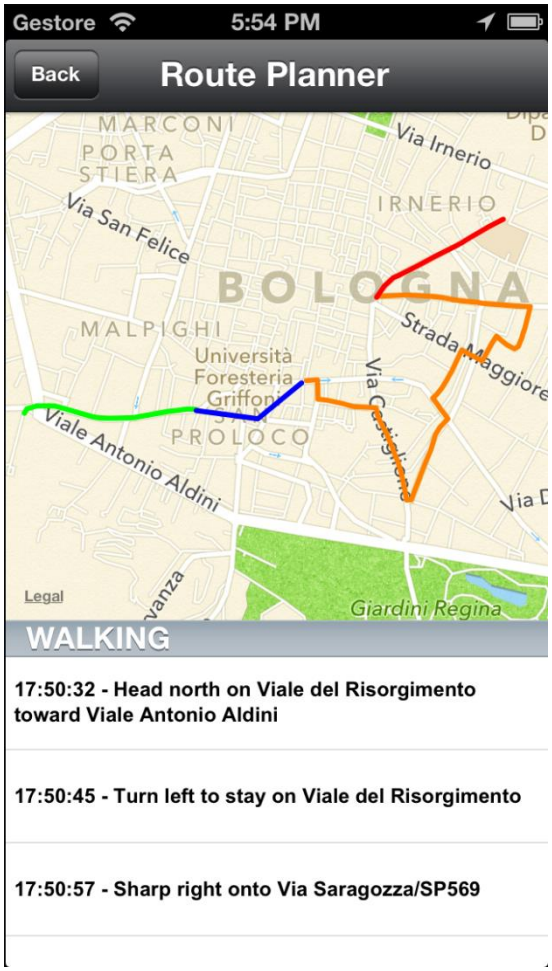

Figura 15: Schermata dettagli terzo percorso

(rosso) e a piedi (verde). E' riportato infine un ultimo possibile percorso (*figura 23*) in cui sono presenti ben quattro mezzi di trasporto: mezzi pubblici (blu), bicicletta (rosso), con l'automobile (arancione) e a piedi (verde).

4.2.2.1 Considerazioni tecniche

Vengono ora analizzati il numero di richieste alle API di Google Maps per avere un'idea di quante volte l'algoritmo può essere avviato tenendo conto dei limiti, abbastanza restrittivi, di richieste massime che si possono fare in 24h ai servizi di Google Maps.

Nell'esempio precedente sono state fatte esattamente 100 richieste al servizio Google Places. Di queste 100 richieste, 80 sono state fatte per ricercare i punti di interesse che l'algoritmo del calcolo dei percorsi deve tenere in considerazione (Nearby Search), 19 sono state fatte per ritornare le possibili destinazioni durante la digitazione di una destinazione da parte dell'utente (Place Autocomplete), 1 è stata fatta per recuperare le coordinate geografiche della destinazione scelta dall'utente (Place Details). In generale il numero di richieste ai servizi Place Autocomplete e Place Details possono variare poiché l'utente potrebbe sbagliarsi durante la digitazione di una destinazione (cancellando per esempio alcune lettere) o potrebbe sbagliarsi anche nel selezionare la sua destinazione finale. Comunque, dato il limite giornaliero di 100000 richieste, è possibile potenzialmente eseguire questo algoritmo per ben $\frac{100000}{100} = 1000$ volte (tenendo in considerazione solo delle richieste al servizio Google Places).

Per quanto riguarda invece il numero di richieste al servizio Directions, sono state fatte esattamente 71 richieste. Di queste 71 richieste, 11 sono state fatte per il calcolo dei percorsi con la macchina, la bicicletta e a piedi e 15 sono state fatte per il calcolo dei percorsi con i mezzi pubblici. Si ricorda che una richiesta di calcolo di percorso con i mezzi pubblici vale come quattro richieste, quindi il numero di richieste con i mezzi pubblici è di $15 * 4 = 60$. Dato il limite giornaliero di

massimo 2500 richieste, potenzialmente è possibile eseguire tale algoritmo solo per $\left\lfloor \frac{2500}{71} \right\rfloor = 35$ volte (tenendo in considerazione solo delle richieste al servizio Directions).

In definitiva, un utente che possiede un iPhone con installata l'applicazione "Route Planner", può calcolare fino a 35 percorsi (35 volte che spinge il pulsante "Go").

Facendo un discorso più in generale, si vuole ora calcolare il numero di terminali (iPhone) massimi che possono, nell'arco di una giornata, utilizzare a pieno l'applicazione. Si suppone che un qualsiasi utente calcoli in una giornata proprio 35 percorsi (di più non si può). Allora il numero di richieste che ogni utente (iPhone) fa al servizio Google Places è di $100 * 35 = 3500$. Quindi il numero massimo di utenti che possono, in una giornata, utilizzare a pieno l'applicazione è $\left\lfloor \frac{100000}{3500} \right\rfloor = 28$. Si tenga presente in quest'ultimo calcolo che le richieste al servizio Google Places richiedono una *key* specifica per l'applicazione. Quindi tutte le richieste che verranno effettuate dai diversi terminali è come se provenissero da un unico dispositivo. Si hanno per cui massimo 100000 richieste indipendentemente dal dispositivo che le genera. Discorso diverso vale per il servizio Directions dato che le richieste non richiedono la *key* dell'applicazione. Quindi potenzialmente è possibile che infiniti terminali utilizzino il servizio Directions, con il solo limite però che ogni dispositivo può generare al massimo 2500 richieste in 24h.

L'ostacolo principale quindi è dato dal limite imposto da Google di 100000 richieste massime giornaliere al servizio Google Places. Se non esistesse tale limite, infiniti terminali avrebbero la possibilità di utilizzare a pieno l'applicazione (vige solo il limite dei 35 percorsi massimi calcolabili per ogni dispositivo in 24h). Per superare il limite giornaliero delle 2500 richieste massime giornaliere al servizio Directions, esistono delle API a pagamento rivolte al mondo Enterprise: Google Maps API for Business. Queste API hanno un costo annuale che parte da 10000 $ ma offrono molti vantaggi in più per le aziende che le utilizzano. Nel caso dell'app che è

stata sviluppata, “Route Planner”, non conviene affatto acquistare tali API dato che si avrebbe solo il misero vantaggio dell’innalzamento del limite giornaliero per il servizio Directions da 2500 a 100000. Il limite giornaliero del numero di richieste al servizio Google Places rimarrebbe invariato. Per superare tale limite è necessario

Funzioni	Maps API	API di Maps for Business
Street View	✓	✓
Servizio web di geocodifica	2500 richieste al giorno	100.000 richieste al giorno
Servizio web di indicazioni stradali	2500 richieste al giorno con 10 waypoint per richiesta	100.000 richieste al giorno con 23 waypoint per richiesta
Servizio web di matrice delle distanze	100 elementi per query 100 elementi per 10 secondi 2500 elementi al giorno	625 elementi per query 1000 elementi per 10 secondi 100.000 elementi al giorno
Servizio web per l'altimetria	2500 richieste al giorno con 25.000 campioni al giorno	100.000 richieste al giorno con 1.000.000 di campioni al giorno
Risoluzione massima dell'API per mappe statiche	640 x 640	2048 x 2048
Scala massima dell'API per mappe statiche	2X	4X
Risoluzione massima dell'API per le immagini di Street View	640 x 640	2048 x 2048
Livello dei dati demografici		✓

Assistenza	Maps API	API di Maps for Business
Risorse per gli sviluppatori dell'API di Google Maps	✓	✓
Accordo sul livello di servizio		✓
Assistenza tecnica		✓
Portale di assistenza e rapporti sull'utilizzo		✓

Casi di utilizzo	Maps API	API di Maps for Business
Gratis e disponibile pubblicamente	✓	✓
Implementazioni interne		✓
Incorporamento in software e applicazioni a pagamento		✓
Servizi di rivendita con Google Maps		✓
Controllo sulle opzioni relative alla pubblicità		✓
Monitoraggio degli asset privati		✓

Figura 16: Differenze Maps API e API di Maps for Business

contattare direttamente Google tramite la Google Api Console (https://code.google.com/apis/console).

In realtà le Google Maps API for Business offrono molto di più offrendo funzioni avanzate e assistenza per le organizzazioni che aggiungono le mappe a siti web o applicazioni mobili a pagamento, a siti web interni o ad applicazioni di

monitoraggio degli asset. In *figura 24* vengono mostrate tutte le differenze tra le Maps API e le API di Maps for Business.

Conclusioni

Durante la stesura dell'elaborato sono state affrontate diverse tematiche attuali di interesse.

Innanzitutto è stato descritto il framework Titanium, grazie al quale è possibile lo sviluppo di applicazioni Cross Platform, fornendone una visione d'insieme ed elencando vantaggi e svantaggi derivanti dal suo utilizzo.

Poi è stata affrontata la tematica degli Open Data che, si presume, diventeranno sempre più importanti dato che spesso essi possiedono un grande potenziale economico, sia per il loro contenuto intrinseco, sia per la loro ampiezza e varietà.

In seguito sono state analizzate le API rilasciate da Google Maps grazie alle quali è stato possibile implementare l'applicazione "Route Planner".

Infine è stata presentata l'applicazione "Route Planner" che è stata sviluppata, insieme ad una descrizione dettagliata dell'algoritmo ad-hoc per il calcolo dei percorsi.

Sviluppi futuri:

Estensione funzionalità:

- Far scegliere all'utente i mezzi preferiti che intende utilizzare;
- Salvare le destinazioni preferite dell'utente;
- Aggiunta di un parametro riguardante il numero di cambi di mezzi di trasporto che l'utente dovrà effettuare nella schermata dove sono presenti i risultati;
- Maggiore personalizzazione per quanto riguarda il vincolo del massimo numero di ore permesse per raggiungere una certa destinazione aggiungendo anche i minuti;
- Adattamento ad iOS 7, nel momento in cui Titanium rilascerà l'aggiornamento che risolve il problema della mancata stampa sulla console.

Miglioramenti dell'algoritmo:

- Considerare solo i punti di interesse più rilevanti in modo tale da essere certi di poter soddisfare a pieno tutte le richieste al servizio Google Directions;
- Mostrare comunque tutti i percorsi calcolati nel caso in cui venga sforato il limite delle 2500 richieste giornaliere;

Bibliografia

[0] Sviluppo di applicazioni mobile con Titanium Appcelerator (1° parte): http://vimeo.com/46222897

[1] Titanium Mobile: Flexibility vs. Performance: http://www.whymca.org/intervento/titanium-mobile-flexibility-vs-performance

[2] Titanium SDK & Titanium Studio: http://docs.appcelerator.com/titanium/latest/

[3] Appcelerator Titanium: http://en.wikipedia.org/wiki/Appcelerator_Titanium

[4] Sviluppare app cross platform per iOS e Android con Appcelerator Titanium: da amore a odio: http://www.marcosiino.it/sviluppo-2/librerie-e-frameworks/sviluppare-app-cross-platform-ios-android-con-appcelerator-titanium-da-amore-odio/

[5] App native contro web app: quali sono meglio?: http://gadget.wired.it/news/applicazioni/2012/04/02/app-native-web-codemotion-89251.html

[6] Mobile web: i 3 migliori framework per creare mobile web apps HTML5: http://www.html5today.it/link/mobile-web-3-migliori-framework-creare-mobile-web-apps-html5

[7] Sencha Touch e Titanium: http://www.techmate.it/?tag=cross-platforms

[8] Introduzione allo sviluppo mobile cross platform con Xamarin: http://www.winfxitalia.com/articoli/xamarin/introduzione-sviluppo-mobile-cross-platform-xamarin.aspx

[9] PhoneGap e Titanium, esiste davvero il write once run anywhere su mobile? http://blogs.aspitalia.com/cradle/post2730/PhoneGap-Titanium-Esiste-Davvero-Write-Once-Run-Anywhere.aspx

[10] Sencha Touch: un framework HTML5 per dispositivi mobili

http://www.melablog.it/post/11991/sencha-touch-un-framework-html5-per-dispositivi-mobili

[11] Sencha touch: applicazioni mobile HTML5 senza sforzo

http://www.html.it/articoli/sencha-touch-applicazioni-mobile-html5-senza-sforzo-1/

[12] Introduzione a jQuery Mobile

http://www.html.it/pag/19526/introduzione-a-jquery-mobile/

[13] Creare applicazioni mobili con JQTouch

http://www.mrwebmaster.it/jquery/creare-applicazioni-mobili-jqtouch_7751.html

[14] MonoTouch

http://blog.html.it/02/07/2009/monotouch/

[15] Cosa sono gli Open Data

http://sardiniaopendata.org/cosa-e-lopen-data/

[16] Open Data

http://www.scribd.com/doc/142068913/Open-Data-Alberto-Pellizzon

[17] Open Data: Come rendere aperti i dati delle pubbliche amministrazioni

http://www.funzionepubblica.gov.it/media/982175/vademecumopendata.pdf

[18] Infografica

http://www.dati.gov.it/content/infografica

[19] Open Data: I nuovi modelli di business

http://amslaurea.unibo.it/4503/1/dallolio_valentina_tesi.pdf

[20] Chi siamo

http://www.tper.it/azienda/chi-siamo

[21] SETA Spa

http://www.setaweb.it/azienda.php?id_azienda=4

[22] Il Gruppo Start

http://www.startromagna.it/chi-siamo/il-gruppo-start/

[23] L'Azienda TEP: azienda della mobiltà

http://www.tep.pr.it/azienda/default.aspx

[24] On line da oggi il sito OpenData del Comune di Bologna

http://www.modena2000.it/2012/04/05/on-line-da-oggi-il-sito-opendata-del-comune-di-bologna/

[25] Google Maps

http://it.wikipedia.org/wiki/Google_Maps

[26] Quick start con le Google Maps API

http://www.hostingtalk.it/quick-start-con-le-google-maps-api_-c000000Cg/

[27] Google Maps JavaScript API v3

https://developers.google.com/maps/documentation/javascript/tutorial

[28] Google Maps API Web Services

https://developers.google.com/maps/documentation/webservices/

[29] The Google Directions API

https://developers.google.com/maps/documentation/directions/

[30] Google Places API

https://developers.google.com/places/documentation/

[31] Place Search

https://developers.google.com/places/documentation/search

[32] Place Actions

https://developers.google.com/places/documentation/actions

[33] Place Details

https://developers.google.com/places/documentation/details

[34] Place Autocomplete

https://developers.google.com/places/documentation/autocomplete

[35] Frequently Asked Questions

https://developers.google.com/maps/documentation/business/faq#pageview

[36] Why Google Maps API for Business?

http://www.google.com/intl/it/enterprise/earthmaps/maps-faq.html

[37] Google Maps API for Business

https://www.google.it/intl/it/enterprise/mapsearth/products/mapsapi.html

[38] Mobilità urbana che fare?

http://titano.sede.enea.it/Stampa/skin2col.php?page=eneaperdettagliofigli&id=47

[39] Trasporto sostenibile e mobilità
http://www.iuses.eu/materiali/ita/MANUALI_PER_RAGAZZI/Manuale_trasporti_ed2.pdf
[40] Mobile Frameworks Comparison Chart
http://www.markus-falk.com/mobile-frameworks-comparison-chart/

Appendice A

Risposte ai servizi di Google Maps

Si vuole ora dare un qualche esempio di risposte in formato JSON alle richieste ai servizi di Google Maps che sono stati utilizzati nell'applicazione "Route Planner": Directions, Place Search, Place Details, Place Actions e Place Autocomplete.

A.1 Directions Responses

Un esempio di richiesta HTTP che calcola il percorso da Chicago, IL a Los Angeles, CA attraverso due tappe intermedie per Joplin, MO e Oklahoma City, OK è la seguente:

```
http://maps.googleapis.com/maps/api/directions/json?origin=Chicag
o,IL&destination=Los+Angeles,CA&waypoints=Joplin,MO|Oklahoma+City
,OK&sensor=false
```

La risposta in JSON è mostrata sotto. Dato che la risposta può essere molto verbosa, gli elementi ripetuti sono stati omessi per chiarezza.

```
{
  "status": "OK",
  "routes": [ {
    "summary": "I-40 W",
    "legs": [ {
      "steps": [ {
        "travel_mode": "DRIVING",
        "start_location": {
          "lat": 41.8507300,
          "lng": -87.6512600
        },
        "end_location": {
          "lat": 41.8525800,
          "lng": -87.6514100
        },
        "polyline": {
          "points": "a~l~Fjk~uOwHJy@P"
        },
```

```
        "duration": {
          "value": 19,
          "text": "1 min"
        },
        "html_instructions": "Head \u003cb\u003enorth\u003c/b\u003e
on \u003cb\u003eS Morgan St\u003c/b\u003e toward \u003cb\u003eW
Cermak Rd\u003c/b\u003e",
        "distance": {
          "value": 207,
          "text": "0.1 mi"
        }
      },
      ...
      ... additional steps of this leg
    ...
    ... additional legs of this route
      "duration": {
        "value": 74384,
        "text": "20 hours 40 mins"
      },
      "distance": {
        "value": 2137146,
        "text": "1,328 mi"
      },
      "start_location": {
        "lat": 35.4675602,
        "lng": -97.5164276
      },
      "end_location": {
        "lat": 34.0522342,
        "lng": -118.2436849
      },
      "start_address": "Oklahoma City, OK, USA",
      "end_address": "Los Angeles, CA, USA"
    } ],
    "copyrights": "Map data ©2010 Google, Sanborn",
    "overview_polyline": {
      "points":
"a~l~Fjk~uOnzh@vlbBtc~@tsE`vnApw{A`dw@~w\\|tNtqf@l{Yd_Fblh@rxo@b}@xx
SfytAblk@xxaBeJxlcBb~t@zbh@jc|Bx}C`rv@rw|@rlhA~dVzeo@vrSnc}Axf]fjz@x
fFbw~@dz{A~d{A|zOxbrBbdUvpo@`cFp~xBc`Hk@nurDznmFfwMbwz@bbl@lq~@loPpx
q@bw_@v|{CbtY~jGqeMb{iF|n\\~mbDzeVh_Wr|Efc\\x`Ij{kE}mAb~uF{cNd}xBjp]
fulBiwJpgg@|kHntyArpb@bijCk_Kv~eGyqTj_|@`uV`k|DcsNdwxAott@r}q@_gc@nu
`CnvHx`k@dse@j|p@zpiAp|gEicy@`omFvaErfo@igQxnlApqGze~AsyRzrjAb__@fty
B}pIlo_BflmA~yQftNboWzoAlzp@mz`@|}_@fda@jakEitAn{fB_a]lexClshBtmqAdm
Y_hLxiZd~XtaBndgC"
    },
    "warnings": [ ],
    "waypoint_order": [ 0, 1 ],
    "bounds": {
```

```
      "southwest": {
        "lat": 34.0523600,
        "lng": -118.2435600
      },
      "northeast": {
        "lat": 41.8781100,
        "lng": -87.6297900
      }
    }
  } ]
}
```

A.2 Place Search Responses

Un esempio di richiesta HTTP che vuole recuperare i ristoranti a Sydney è la seguente:

```
https://maps.googleapis.com/maps/api/place/textsearch/xml?query=r
estaurants+in+Sydney&sensor=true&key=AddYourOwnKeyHere
```

La risposta in formato JSON viene mostrata sotto:

```
{
   "html_attributions" : [
      "Listings by \u003ca
href=\"http://www.yellowpages.com.au/\"\u003eYellow
Pages\u003c/a\u003e"
   ],
   "results" : [
      {
         "formatted_address" : "529 Kent Street, Sydney NSW,
Australia",
         "geometry" : {
            "location" : {
               "lat" : -33.8750460,
               "lng" : 151.2052720
            }
         },
         "icon" :
"http://maps.gstatic.com/mapfiles/place_api/icons/restaurant-
71.png",
         "id" : "827f1ac561d72ec25897df088199315f7cbbc8ed",
         "name" : "Tetsuya's",
         "rating" : 4.30,
```

```
         "reference" :
"CnRmAAAAmmm3dlSVT3E7rIvwQ0lHBA4sayvxWEc4nZaXSSjRtfKRGoYnfr3d5AvQG
k4e0u3oOErXsIJwtd3Wck1Onyw6pCzr8swW4E7dZ6wP4dV6AsXPvodwdVyqHgyGE
         "types" : [ "restaurant", "food", "establishment" ]
      },
      {
         "formatted_address" : "Upper Level, Overseas Passenger
Terminal/5 Hickson Road, The Rocks NSW, Australia",
         "geometry" : {
            "location" : {
               "lat" : -33.8583790,
               "lng" : 151.2100270
            }
         },
         "icon" :
"http://maps.gstatic.com/mapfiles/place_api/icons/cafe-71.png",
         "id" : "f181b872b9bc680c8966df3e5770ae9839115440",
         "name" : "Quay",
         "rating" : 4.10,
         "reference" :
"CnRiAAAADmPDOkn3znv_fX78Ma6X5_t7caEGNdSWnpwMIdDNZkLpVKPnQJXP1ghly
SO-
ixqs28UtDmJaOlCHn18pxpj7UQjRzR4Kmye6Gijoqoox9bpkaCAJatbJGZEIIUwRbT
NIE_L2jGo5BDqiosqU2F5QdBIQbXKrvfQuo6rmu8285j7bDBoUrGrN4r6XQ-
PVm260PFt5kwc3EfY",
         "types" : [ "cafe", "bar", "restaurant", "food",
"establishment" ]
      },
      {
         "formatted_address" : "107 George Street, The Rocks NSW,
Australia",
         "geometry" : {
            "location" : {
               "lat" : -33.8597750,
               "lng" : 151.2085920
            }
         },
         "icon" :
"http://maps.gstatic.com/mapfiles/place_api/icons/restaurant-
71.png",
         "id" : "7beacea28938ae42bcac04faf79a607bf84409e6",
         "name" : "Rockpool",
         "rating" : 4.0,
         "reference" : "CnRlAAAAVK4Ek78r9yHV56I-zbaTxo9YiroCbTlel-
ZRj2i6yGAkLwNMm_flMhCl3j8ZHN-jJyG1TvKqBBnKQS2z4Tceu-
1kZupZ1HSo5JWRBKd7qt2vKgT8VauiEBQL-
zJiKVzSy5rFfilKDLSiLusmdi88ThIQqqj6hKHn5awdj6C4f59ifRoUg67KlbpuGuu
W7S1tAH_EyBl6KE4",
         "types" : [ "restaurant", "food", "establishment" ]
      },
```

```
      {
         "formatted_address" : "483 George Street, Sydney NSW,
Australia",
         "events" : [
            {
              "event_id" : "7lH_gK1GphU",
              "summary" : "Google Maps Developer Meetup: Rockin'
out with the Places API",
              "url" : "https://developers.google.com/places"
            }
          ],
         "geometry" : {
            "location" : {
               "lat" : -33.8731950,
               "lng" : 151.2063380
            }
         },
         "icon" :
"http://maps.gstatic.com/mapfiles/place_api/icons/civic_building-
71.png",
         "id" : "017049cb4e82412aaf0efbde890e82b7f2987c16",
         "name" : "Chinatown Sydney",
         "rating" : 4.0,
         "reference" :
"CnRuAAAAsLNeRQtKD7TEUXWG6gYD7ByOVKjQE61GSyeGZrX-
pOPVps2BaLBlH0zBHlrVU9DKhsuXra075loWmZUCbczKDPdCaP9FVJXB2NsZ1q7188
pqRFik58S9Z1lcWjyVoVqvdUUt9bDMLqxVT4ENmolbgBIQ9Wy0sgDy0BgWyg5kfPMH
CxoUOvmhfKC-lTefXGgnsRqEQwn8M0I",
         "types" : [
            "city_hall",
            "park",
            "restaurant",
            "doctor",
            "train_station",
            "local_government_office",
            "food",
            "health",
            "establishment"
         ]
      }
   ],
   "status" : "OK"
}
```

A.3 Place Details Responses

Un esempio di richiesta HTTP che vuole recuperare più dettagli sul Place identificato dal riferimento "CmRYAAAAciqGsTRX1mXRvuXSH2ErwW-jCINE1aLiwP64MCWDN5vkXvXoQGPKldMfmdGyqWSpm7BEYCgDm-iv7Kc2PF7QA7brMAwBbAcqMr5i1f4PwTpaovIZjysCEZTry8Ez30wpEhCNCXpynextCld2EBsDkRKsGhSLayuRyFsex6JA6NPh9dyupoTH3g" è la seguente:

```
https://maps.googleapis.com/maps/api/place/details/json?reference=
CmRYAAAAciqGsTRX1mXRvuXSH2ErwW-
jCINE1aLiwP64MCWDN5vkXvXoQGPKldMfmdGyqWSpm7BEYCgDm-
iv7Kc2PF7QA7brMAwBbAcqMr5i1f4PwTpaovIZjysCEZTry8Ez30wpEhCNCXpynext
Cld2EBsDkRKsGhSLayuRyFsex6JA6NPh9dyupoTH3g
```

La risposta in formato JSON viene mostrata sotto:

```
{
   "html_attributions" : [],
   "result" : {
      "address_components" : [
         {
            "long_name" : "48",
            "short_name" : "48",
            "types" : [ "street_number" ]
         },
         {
            "long_name" : "Pirrama Road",
            "short_name" : "Pirrama Road",
            "types" : [ "route" ]
         },
         {
            "long_name" : "Pyrmont",
            "short_name" : "Pyrmont",
            "types" : [ "locality", "political" ]
         },
         {
            "long_name" : "NSW",
            "short_name" : "NSW",
            "types" : [ "administrative_area_level_1",
"political" ]
         },
         {
            "long_name" : "AU",
            "short_name" : "AU",
            "types" : [ "country", "political" ]
         },
```

```
         {
            "long_name" : "2009",
            "short_name" : "2009",
            "types" : [ "postal_code" ]
         }
      ],

      "events" : [
        {
          "event_id" : "9lJ_jK1GfhX",
          "start_time" : 1293865200,
          "summary" : "<p>A visit from author John Doe, who will
read from his latest book.</p>
                       <p>A limited number of signed copies will
be available.</p>",
          "url" : "http://www.example.com/john_doe_visit.html"
        }
      ],
      "formatted_address" : "48 Pirrama Road, Pyrmont NSW,
Australia",
      "formatted_phone_number" : "(02) 9374 4000",
      "geometry" : {
         "location" : {
           "lat" : -33.8669710,
           "lng" : 151.1958750
         }
      },
      "icon" :
"http://maps.gstatic.com/mapfiles/place_api/icons/generic_busine
ss-71.png",
      "id" : "4f89212bf76dde31f092cfc14d7506555d85b5c7",
      "international_phone_number" : "+61 2 9374 4000",
      "name" : "Google Sydney",
      "rating" : 4.70,
      "reference" : "CnRsAAAA98C4wD-VFvzGq-
KHVEFhlHuy1TD1W6UYZw7KjuvfVsKMRZkbCVBVDxXFOOCM108n9PuJMJxeAxix3W
B6B16c1p2bY1ZQyOrcu1d9247xQhUmPgYjN37JMo5QBsWipTsnoIZA9yAzA-
0pnxFM6yAcDhIQbU0z05f3xD3m9NQnhEDjvBoUw-BdcocVpXzKFcnMXUpf-
nkyF1w",
      "reviews" : [
         {
            "aspects" : [
               {
                  "rating" : 3,
                  "type" : "quality"
               }
            ],
            "author_name" : "Simon Bengtsson",
            "author_url" :
"https://plus.google.com/104675092887960962573",
            "language" : "en",
```

```
            "rating" : 5,
            "text" : "Just went inside to have a look at
Google. Amazing.",
            "time" : 1338440552869
         },
         {
           "aspects" : [
              {
                 "rating" : 3,
                 "type" : "quality"
              }
             ],
            "author_name" : "Felix Rauch Valenti",
            "author_url" :
"https://plus.google.com/103291556674373289857",
            "language" : "en",
            "rating" : 5,
            "text" : "Best place to work :-)",
            "time" : 1338411244325
         },
         {
           "aspects" : [
              {
                 "rating" : 3,
                 "type" : "quality"
              }
             ],
            "author_name" : "Chris",
            "language" : "en",
            "rating" : 5,
            "text" : "Great place to work, always lots of free
food!",
            "time" : 1330467089039
         }
      ],
      "types" : [ "establishment" ],
      "url" :
"http://maps.google.com/maps/place?cid=10281119596374313554",
      "vicinity" : "48 Pirrama Road, Pyrmont",
      "website" : "http://www.google.com.au/"
   },
   "status" : "OK"
}
```

A.4 Place Actions Responses

Un esempio di richiesta HTTP di tipo POST che vuole aggiungere un Place è la seguente:

```
POST
https://maps.googleapis.com/maps/api/place/add/json?sensor=tru
e_or_false&key=AddYourOwnKeyHere HTTP/1.1
Host: maps.googleapis.com

{
  "location": {
    "lat": -33.8669710,
    "lng": 151.1958750
  },
  "accuracy": 50,
  "name": "Google Shoes!",
  "types": ["shoe_store"],
  "language": "en-AU"
}
```

La risposta in formato JSON viene mostrata sotto:

```
{
  "status": "OK",
  "reference":
"CiQgAAAAeTQS1RtzAyVRVjHcRiIWmWeqcAl3k7bluW7GINLDULESEHozTQhy6O
HJw03ziDvY1uEaFAP_vDRhK-UbWw3Gd7Ulqm3eRjIs",
  "id": "6947fc4007436a71dbda51ef9a58627c8e8858f9"
}
```

Un esempio di richiesta HTTP di tipo POST che vuole cancellare un Place è la seguente:

```
POST
https://maps.googleapis.com/maps/api/place/delete/json?sensor=
true_or_false&key=AddYourOwnKeyHere HTTP/1.1
Host: maps.googleapis.com

{
  "reference":
"CiQgAAAAeTQS1RtzAyVRVjHcRiIWmWeqcAl3k7bluW7GINLDULESEHozTQhy6
OHJw03ziDvY1uEaFAP_vDRhK-UbWw3Gd7Ulqm3eRjIs"
}
```

La risposta in formato JSON viene mostrata sotto:

```
{
  "status": "OK"
}
```

A.5 Place Autocomplete Responses

Un esempio di richiesta HTTP che vuole recuperare tutti i Places geolocalizzati che contengono la stringa “Paris” è la seguente:

```
https://maps.googleapis.com/maps/api/place/autocomplete/json?inpu
t=Paris&types=geocode&sensor=true&key=AddYourOwnKeyHere
```

La risposta in formato JSON viene mostrata sotto:

```
{
  "status": "OK",
  "predictions": [ {
    "description": "Paris, France",
    "id" : "691b237b0322f28988f3ce03e321ff72a12167fd",
    "reference": "CiQYAAAA0Q_JA...kT3ufVLDDvTQsOwZ_tc",
    "terms": [ {
      "value": "Paris",
      "offset": 0
    }, {
      "value": "France",
      "offset": 7
    } ],
    "types": [ "geocode" ],
    "matched_substrings": [ {
      "offset": 0,
      "length": 5
    } ]
  }, {
    "description": "Paris, TX, United States",
    "id" : "518e47f3d7f39277eb3bc895cb84419c2b43b5ac",
    "reference": "CjQjAAAAHnbxZZ...BDR3iIOFdMTxwo1jHg",
    "terms": [ {
      "value": "Paris",
      "offset": 0
    }, {
```

```
      "value": "TX",
      "offset": 7
    }, {
      "value": "United States",
      "offset": 11
    } ],
    "types": [ "geocode" ],
    "matched_substrings": [ {
      "offset": 0,
      "length": 5
    } ]
  }, {
    "description": "Paris, Ontario, Canada",
    "id" : "e7ac9c89d4a590305242b0cb5bf43064027223c9",
    "reference": "CjQhAAAAIv_YWYt...F8KZHY36TwMrbyu_g",
    "terms": [ {
      "value": "Paris",
      "offset": 0
    }, {
      "value": "Ontario",
      "offset": 7
    }, {
      "value": "Canada",
      "offset": 16
    } ],
    "types": [ "geocode" ],
    "matched_substrings": [ {
      "offset": 0,
      "length": 5
    } ]
  }
  ...additional results ...
```

Ringraziamenti

Ringrazio sicuramente tutta la mia mia famiglia e i miei nonni che mi sono sempre stati accanto durante tutto il mio percorso di studi.

Ringrazio anche tutti i miei amici ed in particolar modo i miei più cari amici Lolli, Zanna Bianca, Rossi, il medicone Dallo, Elena, Jimmy e Ксения che mi ha fatto passare quattro mesi molto belli.

Ringrazio infine la Professoressa Sonia Bergamaschi, il Professore Riccardo Martoglia e il dottorando Giovanni Simonini per il sostegno datomi e per aver seguito il mio lavoro pazientemente.

Ringrazio davvero tutti di cuore!!!