ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

CAMPUS DI CESENA
SCUOLA DI SCIENZE

Corso di Laurea in Ingegneria e Scienze Informatiche

PROGETTAZIONE E SVILUPPO DI UN SISTEMA MOBILE BASATO SU ANDROID E SERVIZI REST IN UN PROGETTO DI INTERNET OF THINGS PER IL TRACCIAMENTO E MONITORAGGIO DI IMBARCAZIONI

Elaborata nel corso di: Programmazione di Sistemi Embedded

Tesi di Laurea di:
ELIA BRACCI

Relatore:
Prof. ALESSANDRO RICCI

ANNO ACCADEMICO 2016–2017
SESSIONE II

PAROLE CHIAVE

IoT

Pervasive Computing

Nautica

GPS

Android

RESTful

Alla mia famiglia, a tutte quelle persone che mi sono state vicine, a chi mi ha accompagnato durante questo mio periodo di crescita, a chi non ha mai smesso di credere in me.

Indice

Introduzione

Negli ultimi decenni grazie alle scoperte in ambito scientifico, la tecnologia ha acquisito un ruolo sempre più importante nella società odierna, permettendo così di raggiungere risultati che fino a pochi anni fa erano considerati poco più che fantascienza. Difficilmente si poteva pensare che nel 2017 le luci si sarebbero accese con un battito di mani, gli elettrodomestici si sarebbero azionati da remoto o che ognuno di noi avrebbe posseduto uno smartphone con capacità di calcolo, memoria e connessione dati indubbiamente più vicine ad un Personal Computer che ad un telefono in senso stretto.

Raymond Kurzweil, noto inventore e direttore del reparto Ingegneria a Google, ha dimostrato partendo dall'analisi storica del progresso tecnologico, come l'evoluzione della tecnologia sia caratterizzata da una crescita esponenziale e non lineare, definendola come "Un processo evoluzionistico che nel tempo accelera". Tutto questo è dovuto dal fatto che questi meccanismi "Funzionano per interazione, cioè creano una funzionalità, e poi usano quella funzione per fare il prossimo passo" [1].

Lo sviluppo costante nel tempo di questi processi hanno portato nel 1999, Kevin Ashton, cofondatore e direttore esecutivo di Auto-ID Center (consorzio di ricerca con sede al MIT), durante una presentazione presso Procter & Gamble, ad introdurre il termine "Internet of Things" ovvero "Internet delle cose", meglio conosciuto come IoT. Tale neologismo è riferito all'estensione di Internet al mondo degli oggetti e dei luoghi concreti [2]. L'Internet delle cose è una evoluzione dell'uso della Rete: gli oggetti si rendono riconoscibili, identificabili nel network, acquisendo così intelligenza grazie alla possibilità di poter inviare dati su sé stessi ed accedere ad informazioni aggregate da parte di altri. Quanto appena descritto è reso possibile dal cosiddetto Cloud, un sistema per l'archiviazione, l'elaborazione o la trasmissione di informazioni disponibili on demand, attraverso Internet. Per "cosa" o "oggetto" si può intendere un dispositivo, un'apparecchiatura,

un impianto, un prodotto tangibile, un'opera, una macchina, un'attrezzatura o un qualsiasi altro oggetto connesso e identificabile in rete. Alcuni esempi di IoT nel Mondo d'oggi sono: sveglie che suonano prima in caso di traffico; scarpe da ginnastica che trasmettono tempi, velocità e distanza per gareggiare in tempo reale con altre persone; le scatole delle medicine che avvisano i familiari se si dimenticano di prendere il farmaco; semafori intelligenti che diventano verdi quando rilevano una macchina avvicinarsi e dall'altro lato non sta passando nessuna macchina [3].

Grazie al collegamento alla Rete, tutti gli oggetti possono avere un ruolo attivo, acquisendo una propria identità, con l'obiettivo di far sì che il mondo elettronico tracci una mappa di quello reale. Per esempio, qualsiasi oggetto e luogo munito di etichetta Identificazione a Radio Frequenza (RFID) o Codici QR, comunica informazioni in rete oppure a dispositivi mobili come i telefoni cellulari. Con il passare degli anni i campi di applicabilità dei sistemi IoT sono costantemente aumentati, passando da applicazioni industriali (processi produttivi), alla logistica e all'infomobilità, all'efficienza energetica, all'assistenza remota e alla tutela ambientale, fino alla robotica e alla domotica e persino alla nautica, campo di pertinenza del progetto OneMile.

In questo progetto, un ruolo di fondamentale importanza è svolto dall'uomo, che tramite lo *ubiquitous computing* (computazione ubiqua) [4], interagisce con le macchine permettendo l'elaborazione delle informazioni. Essa è integrata all'interno di oggetti e attività di tutti i giorni che nel nostro contesto è strettamente correlato all'utilizzo dei pedalò. Opposto al paradigma del desktop (letteralmente: «scrivania»), in cui un utente aziona consciamente una singola apparecchiatura per uno scopo specifico, chi utilizza lo *ubiquitous computing*, nel corso di normali attività, aziona simultaneamente diversi sistemi e apparecchiature di calcolo. Qui, l'utilizzatore può anche non essere cosciente del fatto che i sistemi stiano compiendo autonomamente azioni ed operazioni. Lo *ubiquitous computing* viene descritto come calcolo pervasivo o intelligenza ambientale, mentre quando riguarda principalmente gli oggetti coinvolti, è anche detto calcolo fisico o, per l'appunto, Internet of Things.

Un altro elemento chiave di questo progetto è il GPS [5] sigla di Global Positioning System, un sistema per la determinazione delle tre coordinate geocentriche (latitudine, longitudine e altitudine) relative alla posizione di ogni punto posto sulla superficie terrestre. L'utilizzo di questa tecnologia nasce dall'esigenza delle forze armate statunitensi di operare in ogni an-

golo del mondo in tempi rapidi, senza il supporto di stazioni trasmittenti dislocate su tutta la terra. Esso si basa su tre punti cardine: i satelliti, il canale trasmissivo e i ricevitori a terra. Al giorno d'oggi l'utilizzo del GPS è diventato sempre più comune; ricevitori di segnali GPS sono installati sulle nostre auto, sui nostri smartphone e su molti altri oggetti di uso quotidiano.

Con il passare del tempo, non solo gli strumenti a disposizione della società sono aumentati, ma anche le esigenze delle persone stesse. Ed è proprio da qui che nasce OneMile, dal bisogno di Xavier Cortal (proprietario della InterCortal S.L.U.) di tracciare e monitorare le sue imbarcazioni. Come scopo principale vi è il controllo a distanza del lavoro dei suoi operai, unitamente alla verifica del corretto andamento della sua attività. La InterCortal S.L.U. è una società spagnola con sede a Barcellona, che ha stabilimenti balneari e noleggi di mezzi nautici in tutta la Spagna. Come diretta conseguenza della notevole lontananza, è difficile, per il Sig. Cortal, monitorare contemporaneamente le varie situazioni ed essere aggiornato in Real Time su tutto ciò che accade in ciascuna delle sue proprietà.

OneMile ha come fine ultimo l'agevolazione del lavoro suo e dei suoi addetti, permettendo così di offrire anche una qualità di servizio migliore a tutti i suoi clienti.

Capitolo 1

Il progetto OneMile

1.1 Idea e scelte progettuali

Come accennato nell'introduzione, OneMile nasce con uno scopo ben preciso, ottimizzare la gestione e il controllo delle attività della InterCortal S.L.U.. L'attività che si punta a rivoluzionare è il processo di noleggio, rendendo quest'ultimo automatizzato.

Nel momento in cui un cliente decide di affittare un pattino a pedali, effettuerà in primo luogo il pagamento anticipato della tariffa, contestualmente al ricevimento del biglietto attestante l'avvenuto pagamento. A fine giornata, un addetto appositamente designato, passerà in rassegna ogni stabilimento prelevando in primis l'incasso giornaliero e verificando che l'ammontare di quest'ultimo sia consono al numero di biglietti strappati. Chiaramente in questa particolare situazione si esige un controllo puntuale, accurato e possibilmente computerizzato.

L'elevata necessità di personale è il primo ostacolo nella ricerca di soggetti affidabili ed adatti pertanto a svolgere le varie mansioni.

Un'ulteriore problematica alla base del nostro studio, è la dislocazione delle numerose sedi su tutto il territorio spagnolo, più precisamente nel sud della Spagna. Ciò comporta per la InterCortal, una seria difficoltà nell'avere tutto sotto controllo in qualsiasi momento.

La soluzione attuale, adottata da Xavier per far fronte a questo dilemma, è stata quella di selezionare, per ogni stagione balneare, persone fidate che si rechino nelle varie zone accertandone il corretto andamento. I costi di quanto appena descritto, sono ovviamente molto elevati: trasporti, vitto

e alloggio, personale ecc. sono solo alcune delle voci che prese nella loro interezza, risultano ben altro che irrisorie per l'azienda.

Il sig.Cortal stesso ha illustrato i punti cardine sopra descritti, riassumendo gli obiettivi di OneMile come segue:

- monitoraggio remoto Real Time delle sedi e delle attrezzature dislocate sul territorio;
- monitoraggio del corretto svolgimento delle attività lavorative;
- stima degli incassi in tempo reale e verifica giornaliera di questi ultimi.

Il primo passo compiuto è stato quello di verificare se già fosse in commercio, un sistema in grado di soddisfare tali requisiti. Inizialmente non sembrava fosse stato trovato nulla di simile ma procedendo con le ricerche è stato rinvenuto un rilevatore GPS prodotto in Columbia [6].

In breve, il funzionamento del dispositivo colombiano, si basa sull'utilizzo di un rilevatore GPS per la ricezione della posizione e di una sim GSM per la trasmissione dei dati. Questi ultimi saranno in seguito disponibili da remoto, tramite indirizzi e credenziali forniti dall'azienda al momento dell'acquisto. Analizzando il funzionamento della tecnologia trovata, divenne chiaro come i costi di mantenimento e acquisto fossero al di sopra delle aspettative. Ciò era causato dal fatto che ogni imbarcazione avrebbe dovuto installare una copia del sistema rendendo i prezzi degli abbonamenti per le sim insostenibili.

Perciò, escludendo la soluzione precedentemente descritta, si è iniziato a spaziare nel mondo delle comunicazioni WAN in ambito IoT, giungendo a prodotti come Sigfox; Sigfox è il principale servizio di connettività Internet per questi sistemi, basato su principi come il basso consumo, il basso costo e la compatibilità con tutte le tipologie di connessioni wireless. Nonostante questo offra la copertura su tutto il territorio spagnolo, la limitazione del numero di messaggi inviabili nell'arco di una giornata, ha portato ad escluderlo fin da subito ed optare per altre soluzioni. Per il suddetto motivo, si è giunti alla conclusione che, al giorno d'oggi, non è ancora disponibile una tecnologia in grado di soddisfare le esigenze di OneMile.

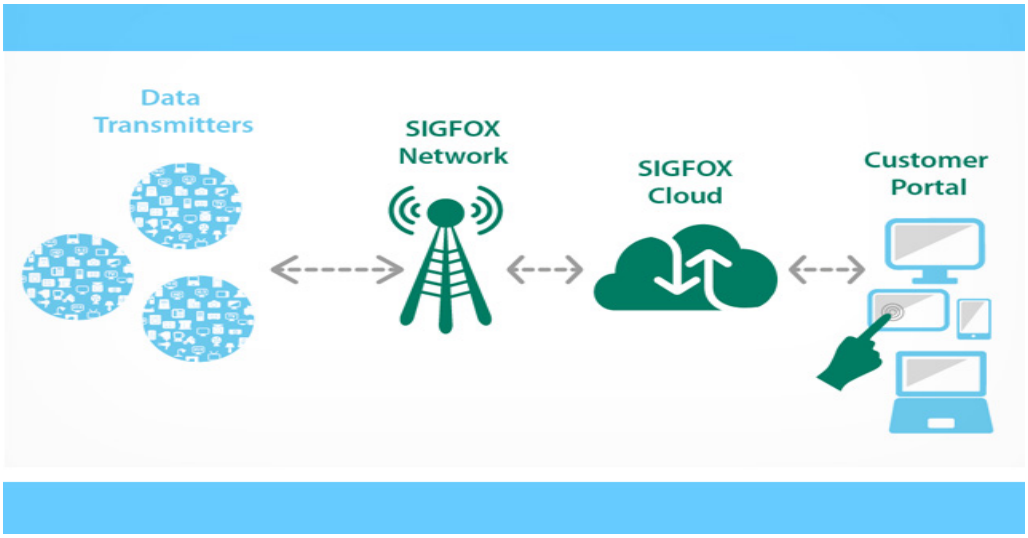


Figura 1.1: Schema funzionamento sistema Sigfox

L'idea è stata quella di progettare un sistema a bassi consumi e basso costo ma con alte possibilità di configurazione, adattamento e funzionalità, il tutto senza tralasciare importanti requisiti quali correttezza, affidabilità, robustezza ed efficienza.

1.2 Requisiti e architettura

Definiti i costi e i tempi di sviluppo del sistema, si è proceduto con l'analisi dei requisiti, suddividendoli in cinque macro-funzioni:

1. reperimento posizione;
2. comunicazione;
3. elaborazione;
4. stoccaggio dati;
5. visualizzazione.

Al termine di numerose sperimentazioni riguardanti le tecnologie da utilizzare, si è optato per suddividere il sistema nel seguente modo:

- un collettore, unità centrale del sistema, con lo scopo di comunicare con le varie imbarcazioni e elaborare e memorizzare i dati nel database;
- i nodi, uno per ogni imbarcazione, dotati di un rilevatore GPS per rilevare la posizione;
- un servizio RESTful, con lo scopo di gestire le richieste HTTP e i dati presenti sul database remoto;
- l'applicazione mobile, per poter visualizzare in tempo reale le informazioni e poterne aggiungere di nuove;
- l'applicazione web, per permettere l'accesso ai dati su tutte le piattaforme.

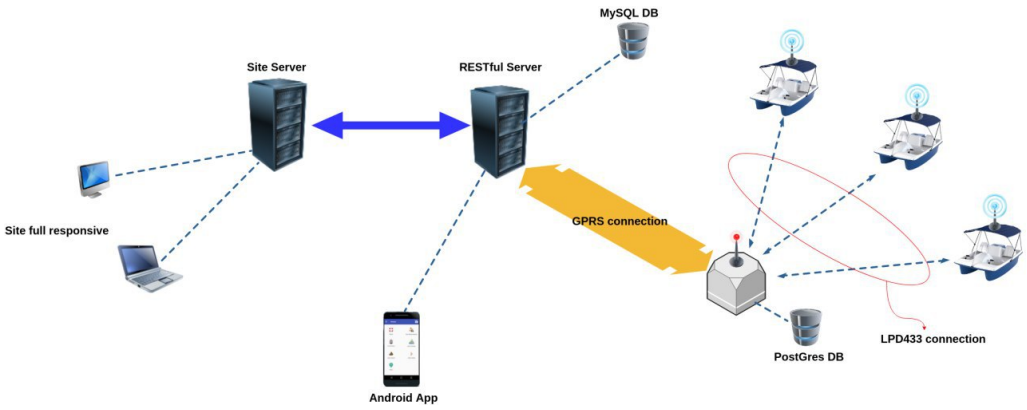


Figura 1.2: Architettura del sistema

Per lo sviluppo del sottosistema collettore, si è optato per l'utilizzo del microprocessore Raspberry Pi3 model B, appositamente configurato e dotato di un RXTX (ricetrasmettitore) a 433Mhz per effettuare la comunicazione con i vari nodi. Questi ultimi, sono composti da un microcontrollore Arduino nano, anch'esso equipaggiato con un ricetrasmettitore a 433Mhz per gestire le richieste del collettore e un ricevitore GPS. Ciascun nodo è dotato di un'antenna supplementare ed è contenuto all'interno di una scatola stagna onde evitare danni causati dall'acqua marina. Il sottosistema RESTful è stato implementato in modo da gestire al meglio le richieste provenienti dai sottosistemi ad esso connessi con l'obiettivo di garantire velocità e

sicurezza nella trasmissione dei dati. Per ciò che concerne l'applicazione mobile, il sistema operativo scelto, seguendo le richieste del committente, è Android. Essa contiene le funzionalità utili alla visualizzazione dei dati in Real Time, all'aggiunta di nuove corse (noleggi) e ulteriori funzioni per agevolare il lavoro dei dipendenti. L'applicazione web, invece, permette a tutti coloro che non possiedono dispositivi Android, di accedere e poter visualizzare, anche se in modo limitato, le informazioni presenti sul database collegato al servizio RESTful.

Al termine delle fasi di sviluppo e progettazione si è proceduto con l'installazione e il collaudo del sistema. L'ultima problematica inerente al progetto OneMile era la scelta della locazione ottimale del dispositivo. Dopo diversi test in mare, l'interno di uno dei due poggiatesta del pedalò si è dimostrata la posizione indubbiamente più adeguata.

Figura 1.3: Installazione del nodo sul pattino a pedali

1.3 Funzionamento complessivo

All'accensione del sottosistema collettore, quest'ultimo sincronizza i dati del suo database locale postgres con i dati del database MySQL collegato al servizio RESTful. Così facendo, è in grado di recuperare le informazioni sui nodi con cui esso dovrà comunicare. Tramite polling, raspberry richiede la posizione a ciascun nodo, trasmettendo tramite il suo modulo a 433Mhz l'ID

dell'imbarcazione e il messaggio contenente l'informazione di cui necessita. Arduino, in caso di ricevimento del messaggio (formato JSON) contenente il suo identificatore, risponde inviando al collettore la sua posizione attuale, definita da latitudine e longitudine. I dati trasmessi vengono successivamente controllati con la verifica del checksum dell'hashmessage. In caso di esito positivo, i dati vengono elaborati, stoccati nel suo database locale e successivamente sincronizzati con quello remoto.

Giunti a questa fase, le informazioni sono pronte per essere visualizzate sulle varie piattaforme messe a disposizione dal progetto OneMile. Il servizio RESTful, in grado di gestire qualsiasi richiesta CRUD che gli viene sottoposta, funge da sorgente per tutti i dati che si vogliono visualizzare.

L'applicazione Android, una volta richieste le informazioni, è in grado di mostrare in diverse schermate le posizioni delle imbarcazioni, le prenotazioni effettuate, i dettagli di ciascuna imbarcazione, il numero di corse ecc. Inoltre, svolge all'interno del sistema un ruolo di fondamentale importanza ovvero la scannerizzazione dei QRCode; questo, presente su ciascun pattino, ha una duplice funzione: identificare ciascuna imbarcazione e registrare le singole corse attraverso la scansione del codice.

In ultimo la webapp consente la visualizzazione da qualsiasi dispositivo delle sole posizioni delle imbarcazioni.

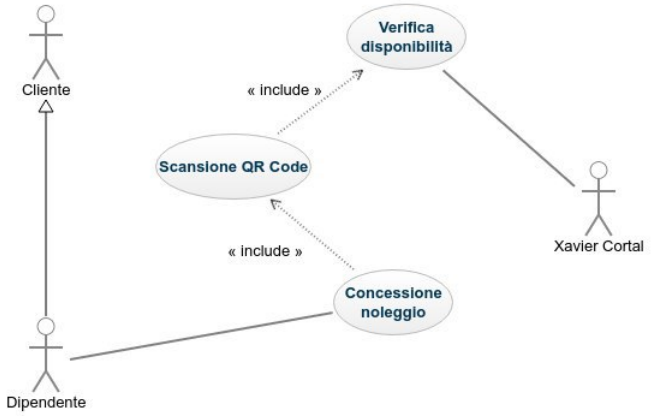


Figura 1.4: Casi d'uso dell'attività di noleggio

Capitolo 2

Programmazione e implementazione prototipale del sottosistema Android

2.1 Struttura e caratteristiche

2.1.1 Sistema operativo Android

Android è un sistema operativo per dispositivi mobili sviluppato da Google e basato sul kernel Linux, tuttavia non è da considerarsi propriamente né un sistema unix-like né una distribuzione GNU/Linux. Ciò è dovuto dal fatto che quasi la totalità delle utilità GNU è sostituita da software in Java, per questo è da ritenersi una distribuzione embedded Linux [7]. Il sistema embedded precedentemente descritto, è progettato principalmente per smartphone e tablet, con alcune interfacce utente specializzate come Andorid TV per i televisori, Android Auto per le automobili, Android Wear per gli orologi da polso e Google Glass per la tecnologia riguardante gli occhiali.

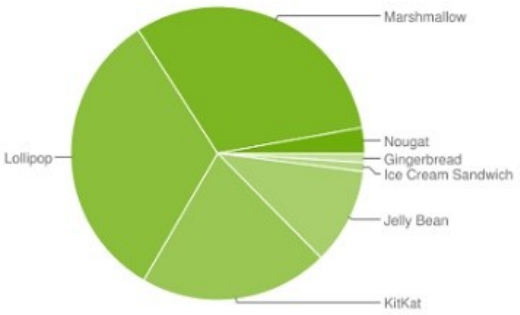
> "Se Google non avesse agito, avremmo dovuto affrontare un futuro spaventoso: un'azienda, un uomo, un dispositivo e un solo operatore sarebbero stati l'unica scelta. Questo non è il futuro che vogliamo. Quindi se credi nella libertà e nell'innovazione, benvenuto in Android."
>
> *Andy Rubin, ex Presidente Android Inc.*

La Android Inc., azienda ideatrice del sistema Android, fu fondata nell'ottobre 2003 da Andy Rubin (cofondatore di Danger), Rich Miner (cofondatore di Danger e di Wildfire Communications), Nick Sears (vicepresidente di T-Mobile) e Chris White (principale autore dell'interfaccia grafica di Web TV). Inizialmente la società operò in segreto, rivelando solo di progettare software per dispositivi mobili. La scarsa disponibilità di denaro ed il budget limitato portarono alla cessione dell'azienda nell'agosto del 2005, acquistata dal colosso di Mountain View, Google. Lo scopo di Google era quello di entrare nel mercato della telefonia mobile, cominciando a sviluppare un sistema operativo basato sul kernel Linux. La prima presentazione ufficiale avvenne nel novembre 2007 all'OHA, un consorzio di aziende del settore Hi Tech che include tra le tante Google, produttori di smartphone come HTC e Samsung, operatori di telefonia mobile come Sprint Nextel e T-Mobile, e produttori di microprocessori come Qualcomm e Texas Instruments Incorporated. Nel 2008 venne lanciato il primo dispositivo equipaggiato con Android; da quella data ci sono stati molti aggiornamenti per migliorarne le prestazioni e per eliminare eventuali problemi di sicurezza delle precedenti versioni. Dalla prima versione rilasciata (Android 1.5), ogni release segue una precisa convenzione alfabetica per i nomi, che in questo caso sono quelli di dolci [8]:

- Android 1.5 – Cupcake (2009)
- Android 1.6 – Donut (2009)
- Android 2.0 / 2.1 – Eclair (2009)
- Android 2.2 – Froyo (2010)
- Android 2.3 – Gingerbread (2010)
- Android 3.0 – Honeycomb (2011)
- Android 4.0 – Ice Cream Sandwich (2011)
- Android 4.1 / 4.2 / 4.3 – Jelly Bean (2012)
- Android 4.4 – KitKat (2013)
- Android 5.0 / 5.1 – Lollipop (2014)

- Android 6.0 – Marshmallow (2015)
- Android 7.0 / 7.1 – Nougat (2016)
- Android 8.0 – Oreo (2017)

Version	Codename	API	Distribution
2.3.3 - 2.3.7	Gingerbread	10	1.0%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	1.0%
4.1.x	Jelly Bean	16	3.7%
4.2.x		17	5.4%
4.3		18	1.5%
4.4	KitKat	19	20.8%
5.0	Lollipop	21	9.4%
5.1		22	23.1%
6.0	Marshmallow	23	31.3%
7.0	Nougat	24	2.4%
7.1		25	0.4%



Data collected during a 7-day period ending on March 6, 2017.
Any versions with less than 0.1% distribution are not shown.

Figura 2.1: Grafico diffusione versione Android maggio 2017

Lo sviluppo di Android prosegue attraverso l'Android Open Source Project, un software libero, ad esclusione di diversi firmware non-liberi inclusi, per i produttori di dispositivi e delle cosiddette "Google Apps" presenti sullo store Google Play. Ad aprile 2017 è il sistema operativo mobile più diffuso al mondo, con una fetta di mercato pari al 62,94% rispetto al totale mondiale, seguito da iOS con solamente il 33,9%. Nello stesso periodo ha superato il marketshare di Windows che, fino ad allora, aveva il più alto marketshare al mondo.

L'innumerevole quantità di dispositivi Android in commercio, il loro prezzo elastico e la loro influenza nel mercato mondiale, sono i motivi

che hanno portato a scegliere Android come sistema operativo mobile per l'applicazione di OneMile.

2.1.2 Requisiti e ambiente di sviluppo

In primis, sono stati definiti i requisiti e i casi d'uso del software, seguendo le linee guida del committente di OneMile.

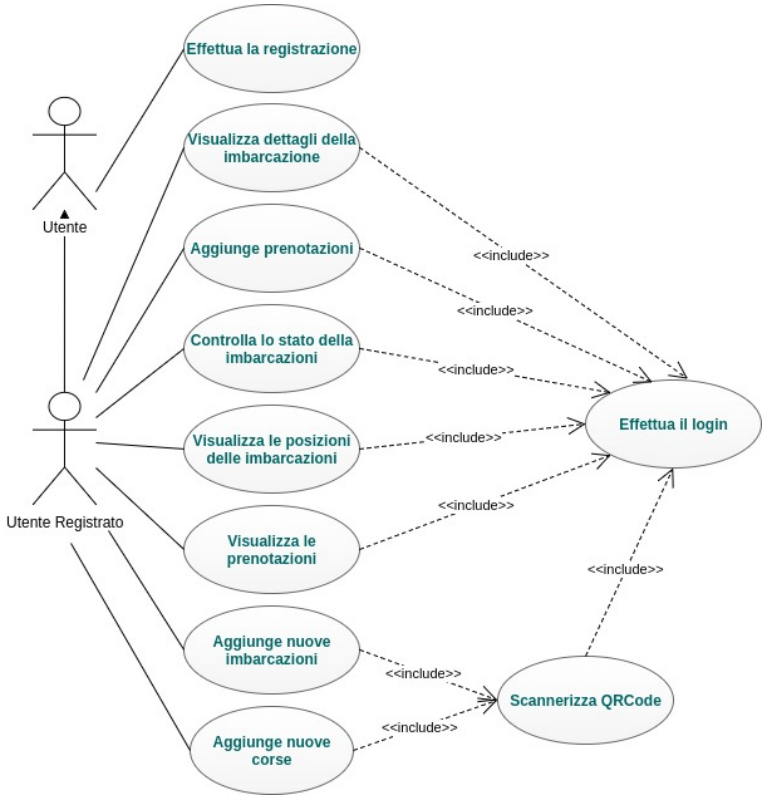


Figura 2.2: Casi d'uso OneMileProject app

La cosiddetta OneMileProject App è stata interamente progettata e implementata attraverso l'utilizzo di Android Studio, l'IDE ufficiale rilasciato da Google per lo sviluppo di applicazioni native Android. Esso permette la creazione di applicazioni per ogni tipo di dispositivo Android, fornendo strumenti come l'editing, il debug, il performance tool, un sistema di build flessibile e un sistema di build/deployment istantaneo a livello mondiale [9].

2.1.3 Architettura

Ogni progetto in Android Studio è composto da uno o più moduli, contenenti file di codice sorgente e file definiti come risorse. I vari tipi di moduli includono:

- i moduli dell'applicazione Android;
- i moduli delle librerie;
- i moduli di Google App Engine (una piattaforma come servizio PaaS di cloud computing per lo sviluppo e l'hosting di applicazioni web gestite dai Google Data Center [10]).

L'impostazione predefinita di questo software, offre una visualizzazione dei file del progetto suddivisi nei vari moduli, fornendo un accesso rapido alle varie risorse contenute in ciascun di essi. L'integrità dei file relativi alla parte di build sono visibili all'interno del modulo Gradle Script, mentre quello dedicato all'applicazione, contiene le seguenti cartelle:

- manifests: contiene il file AndroidManifest.xml;
- java: contiene i file di codice sorgente Java, incluso il codice di prova JUnit;
- res: contiene tutte le risorse di non-codice, come layout XML, stringhe UI e immagini bitmap;

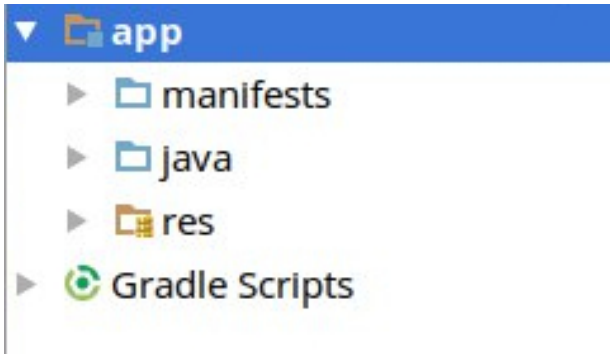


Figura 2.3: Visualizzazione struttura applicazione su Android Studio

Il file AndroidManifest.xml fornisce al sistema Android informazioni essenziali sull'applicazione che esso deve conoscere ancora prima di poter eseguire qualsiasi parte di codice. Più precisamente contiene [11]:

- il nome del package Java, che serve come identificativo univoco per l'applicazione;

```
1 <manifest xmlns:android="http://schemas.android.com/apk/res
    /android" package="com.example.root.onemile">
```

Listing 2.1: Dichiarazione del package Java

- i componenti dell'applicazione, come le attività, i servizi, i broadcast receivers e i diversi contents providers. Inoltre, le classi implementate dichiarando le loro capacità, come gli Intents che possono gestire;

```
1  <activity android:name=".MainActivity">
2      <intent-filter>
3          <action android:name="android.intent.action.MAIN" />
4          <category android:name="android.intent.category.
               LAUNCHER" />
5      </intent-filter>
6  </activity>
7  <activity android:name=".HomeActivity"/>
8  <activity android:name=".qrcodescannerpackage.QRcodeScanner
       "/>
9  <activity android:name=".ForgotPasswordActivity"/>
10 <activity android:name=".RegistrationActivity"/>
11 <activity android:name=".ReportBugActivity"/>
```

Listing 2.2: Dichiarazione delle attività e degli Intent

- i processi host dell'applicazione;

- le autorizzazioni e i permessi che l'applicazione deve avere per accedere alle parti protette dell'API, interagire con le altre applicazioni e permettere agli altri utenti di interagire con essa;

```
1 <uses-permission android:name="android.permission.CAMERA" />
2 <uses-permission android:name="android.permission.
    ACCESS_NETWORK_STATE" />
3 <uses-permission android:name="android.permission.BLUETOOTH
    " />
```

```
4 <uses-permission android:name="android.permission.INTERNET"
      />
5 <uses-permission android:name="android.permission.
      ACCESS_FINE_LOCATION" />
6 <uses-permission android:name="android.permission.
      ACCESS_COARSE_LOCATION" />
7 <uses-feature android:name="android.hardware.location.gps"
      />
8 <uses-permission android:name="com.example.permission.
      MAPS_RECEIVE" />
9 <meta-data android:name="com.google.android.geo.API_KEY"
      android:value="YourGoogleAPIKey" />
```

Listing 2.3: Dichiarazione dei permessi e dichiarazione della GoogleAPIKey

- le classi Instrumentation che forniscono le informazioni di profilo e altre informazioni all'esecuzione dell'applicazione. Queste dichiarazioni sono presenti nel manifest solo durante la fase di sviluppo dell'applicazione e vengono successivamente rimosse prima che essa venga pubblicata;

- le librerie necessarie per il corretto funzionamento del software.

La directory Java contiene al suo interno diversi packages e le Activities dell'applicazione. I packages presenti sono:

- com.example.root.onemile, contenente il codice sorgente Java;

- com.example.root.onemile (androidTest), contenente i test sull'integrità dell'applicazione Android;

- com.example.root.onemile (test), contenente i test JUnit.

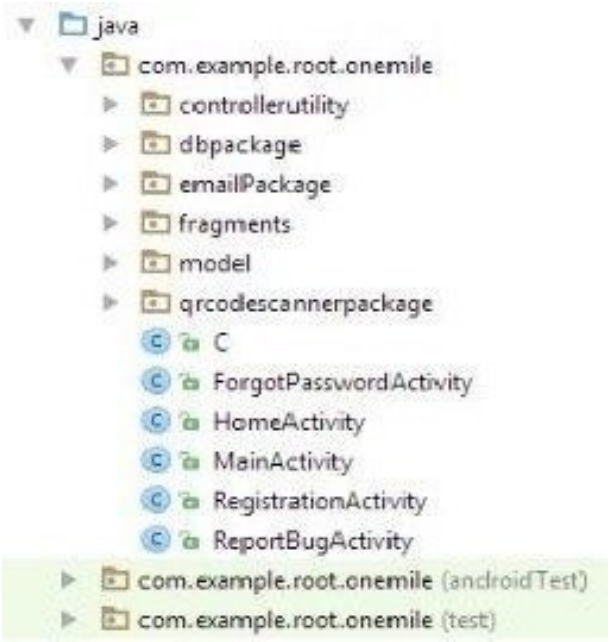


Figura 2.4: Visualizzazione contenuto del package Java

Il primo è suddiviso a sua volta in:

- controllerUtility, questo package contiene varie classi e package per agevolare il lavoro del controller;
- dbPackege, contiene le classi e i package per la connessione al database;
- emailPackage, contenente le classi per l'utilizzo del client Gmail;
- fragments, package contenente i diversi Fragment dell'applicazione;
- model, il model vero e proprio dell'applicazione contenente tutti i bean necessari;
- qrcodescannerpackage, il package relativo all'utilizzo dello scanner dei QRcode;
- le Activities dell'applicazione;
- la classe C contenente le stringhe pubbliche.

Dato l'utilizzo del MCV (Model-View-Controller), un pattern adoperato in programmazione per dividere il codice in blocchi dalle funzionalità ben distinte, il package controllerUtility include al suo interno le funzionalità principali relative al controller. Più precisamente è composto da:

- i custom adapter per customizzare i dati da mostrare nella view;
- le classi relative al mappaggio dei JSON di risposta inviati dal servizio RESTful;
- le classi per le richieste HTTP;
- la classe per la codifica e decodifica sicura dei dati sensibili.

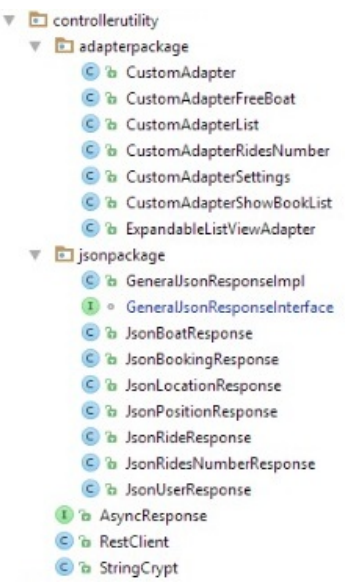


Figura 2.5: Package controllerUtility

La gestione delle richieste HTTP, una delle funzionalità principali dell'applicazione, è interamente affidata alla classe RestClient. Il suo scopo è inviare al server RESTful le richieste di GET e POST tramite la creazione di un client HTTP e il successivo utilizzo dei suoi metodi (get e post). Questi ultimi necessitano come parametri l'URL, l'handler per la gestione della risposta e eventuali parametri della richiesta stessa. Tuttavia, le funzioni utilizzate permettono di eseguire richieste ad URL HTTPS solamente dalla versione di Android Nougat, ovvero Android 7.0. Da questa versione, sono state inserite dai programmatori di Google nuove funzionalità e supporti per le librerie utilizzate. Per far fronte a questa problematica si è proceduto con l'implementazione di un metodo in grado di controllare, al momento della richiesta, l'SDK della versione di Android in uso, riuscendo così a definire l'URL da utilizzare (HTTP o HTTPS).

```
1 private static String getAbsoluteUrl(String relativeUrl) {
2     if(Build.VERSION.SDK_INT < 25){
3         return C.BASE_HTTP_URL + relativeUrl;
```

```
4       }else{
5           return C.BASE_HTTPS_URL + relativeUrl;
6       }
7   }
```

Listing 2.4: Controllo SDK della versione di Android

Gli URL statici delle varie richieste sono memorizzati come variabili statiche all'interno della classe C, in modo da facilitare e velocizzare eventuali modifiche.

```
1 public static final String BASE_HTTPS_URL = "https://www.
    onemileprojectrestful.altervista.org/requests/";
2 public static final String BASE_HTTP_URL = "http://www.
    onemileprojectrestful.altervista.org/requests/";
3 public static final String url_get_all_postions_from_location =
    "get_all_positions_from_location.php";
4 public static final String url_get_all_locations = "
    get_all_locations.php";
```

Listing 2.5: Stringhe classe C

Al fine di ridurre al minimo le problematiche precedentemente descritte relative alla versione di Android utilizzata, si è creato, tramite l'utilizzo del pattern di programmazione Singleton, la classe StringCrypt. Questa permette di codificare e decodificare, in maniera opportuna, le informazioni scambiate tra client e server.

```
1  public class StringCrypt {
2
3      private static StringCrypt instance = null;
4
5      /**
6       *  return a new instance of the StringCrypt class,
7       *  if it has not already been instantiated
8      */
9      public static StringCrypt getInstance(){
10         if(instance == null){
11          instance = new StringCrypt();
12         }
13         return instance;
14     }
15
16     /**
17      * private constructor
```

```
18      */
19      private StringCrypt (){
20
21
22      }
23
24  }
```

Listing 2.6: Singleton pattern

Le risposte inviate dal server, in formato JSON, dovranno essere poi mappate nel client in modo da poter gestire al meglio i dati contenuti. Per far sì che tutto ciò sia possibile, con il supporto della libreria per i JSON di Google (GSON), sono state implementate delle classi, ciascuna per ogni tipologia di richiesta. Gli elementi in comune tra tutte le risposte sono contenuti all'interno della classe GeneralJsonResponseImpl, che a sua volta implementa l'interfaccia GeneralJsonResponseInterface.

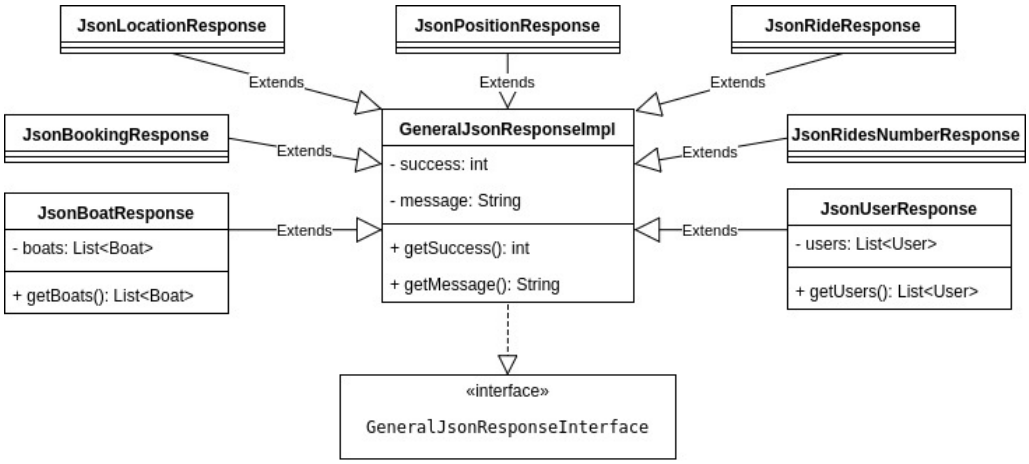


Figura 2.6: Schema JSON mapping

Quando viene avviata una nuova componente dell'applicazione Android e quest'ultima non dispone di altri componenti in esecuzione sul sistema, quest'ultimo avvia un nuovo processo Linux su un singolo thread. Per impostazione predefinita, tutti i componenti della stessa applicazione vengono eseguiti nello stesso processo, chiamato Main Thread. Nella casistica in cui venisse avviato un nuovo componente, ma l'applicazione risulta già in esecuzione, questo viene eseguito all'interno del Main Thread. Tuttavia, si

ha la possibilità di creare processi aggiuntivi per qualsiasi componente, in modo da poterli eseguire separatamente. È altamente sconsigliato eseguire all'interno del thread principale processi che utilizzano la rete internet e per questo le richieste al server sono effettuate tramite l'utilizzo dell'interfaccia AsyncTask, fornita da Android. Così facendo, si riesce ad eseguire correttamente l'applicazione evitando bug e arresti improvvisi. Inoltre, grazie all'esecuzione in background di queste classi, si evita di manipolare o intervenire sul thread principale.

```
1  public class CreateUser extends AsyncTask<User, String, String>
       {
2      //Delegate for the processFinish method
3      private AsyncResponse delegate = null;
4      // Progress Dialog
5      private ProgressDialog pDialog;
6
7      private Context context = null;
8
9      private String ret;
10
11     public CreateUser(Context c, AsyncResponse delegate) {
12         this.context = c;
13         this.delegate = delegate;
14     }
15
16     /**
17      * Before starting background thread Show Progress Dialog
18      */
19     @Override
20     protected void onPreExecute() {
21         /*
22         * Setting the process dialog
23         */
24     }
25
26     /**
27      * Creating user
28      */
29     protected String doInBackground(User... u) {
30
31         /*
32         * Building parameters from the User u
33         */
34
```

```
35          // Creating the Json Http Response Handler to handle the
                JSON response
36          JsonHttpResponseHandler js = new JsonHttpResponseHandler
                () {
37              @Override
38              public void onSuccess(int statusCode, Header[]
                    headers, JSONObject response) {
39                  ret = response.toString();
40              }
41
42              @Override
43              public void onFailure(int statusCode, Header[]
                    headers, String responseString, Throwable
                    throwable) {
44                  super.onFailure(statusCode, headers,
                        responseString, throwable);
45              }
46
47          };
48
49          js.setUseSynchronousMode(true);
50
51          RestClient.post(C.url_create_user, params, js);
52
53          return ret;
54      }
55
56      /**
57       * After completing background task Dismiss the progress
              dialog
58       */
59      protected void onPostExecute(String output) {
60          //call the processFinish Method of the delegate created
61          delegate.processFinish(output);
62          // dismiss the dialog once done
63          pDialog.dismiss();
64      }
65
66  }
```

Listing 2.7: Esempio di AsyncTask per la creazione di un nuovo utente

Le classi che implementano i vari AsyncTask, sono contenute all'interno del dbPackage. Per consentire a classi esterne ad esse di accedere alla stringa in formato JSON ritornata dall'esecuzione della richiesta, si è appo-

sitamente implementata un'interfaccia AsyncResponse contenente il metodo processFinish.

```
1  public interface AsyncResponse {
2
3      void processFinish(String output);
4
5  }
```

Listing 2.8: Interfaccia AsyncResponse

Quest'ultimo viene richiamato al termine dell'esecuzione del task e, le classi esterne, eseguendo l'ovverride su di esso, sono in grado di poter mappare il JSON di risposta e elaborare i dati contenuti.

```
1  public class RegistrationActivity extends AppCompatActivity
       implements AsyncResponse {
2
3      /*
4      * code of Regisration activity
5      */
6
7      @Override
8      public void processFinish(String output) {
9          Gson gson = new Gson();
10
11         GeneralJsonResponseImpl json = gson.fromJson(output,
               GeneralJsonResponseImpl.class);
12
13         if (json.getSuccess() == 0) {
14             /*
15              * error message
16              */
17         } else {
18             /*
19             * success message
20             */
21         }
22     }
23 }
```

Listing 2.9: Override del metodo processFinish

Al fine di aumentare e migliorare le funzionalità offerte dall'applicazione, si è inserito un package (emailPackage) contenente le classi per l'utilizzo di

un server smtp Gmail, per permettere l'invio di email direttamente dall'applicazione. Questa funzionalità è utilizzata all'interno delle Activities ForgotPasswordActivity, che permette per l'appunto di recuperare la password e ReportBug.

All'interno del package model, contenente le classi bean utili all'applicazione, sono stati adottati alcuni accorgimenti per migliorare la velocità e la fluidità dell'applicazione. Un esempio è il metodo refresh di ciascun bean, in grado di settare a null la lista locale di tutti gli oggetti, relativi a quel bean, presenti sul database remoto. Al momento di una prima interazione con l'applicazione, vengono salvati in liste gli elementi e le informazioni utili presenti sul database, evitando di richiederle ad ogni cambio di Activity o Fragment. I dati memorizzati vengono successivamente aggiornati tramite il metodo refresh, nel momento in cui sono inserite nuove informazioni, come ad esempio durante la scannerizzazione di una nuova corsa o l'aggiunta di una nuova imbarcazione.

```
Boat.refreshBoats(null);
```

```
1  private static List<Boat> getBoats(final Context c) {
2      String output = null;
3      try {
4          output = new GetAllBoats(c).execute().get();
5      } catch (InterruptedException | ExecutionException e) {
6          /*
7           * Error message
8           */
9      }
10     Gson gson = new Gson();
11
12     JsonBoatResponse json = gson.fromJson(output,
           JsonBoatResponse.class);
13
14     if (json == null) {
15         /*
16         * Error message
17         */
18     } else {
19         if (json.getSuccess() == 0) {
20             /*
21             * Error message
22             */
```

```
23          } else {
24              boats = json.getBoats();
25              Boat.setCurrentBoat(boats.get(0));
26              return boats;
27          }
28      }
29      return new LinkedList<>();
30  }
31
32  public static List<Boat> getAllBoats(Context c) {
33      return (boats != null) ? boats : getBoats(c);
34  }
35
36  public static List<Boat> getBoats() {
37      return boats;
38  }
39
40  private static void setBoats(List<Boat> boats) {
41      Boat.boats = boats;
42  }
43
44  public static void refreshBoats(List<Boat> boats) {
45      setBoats(boats);
46  }
```

Listing 2.10: Esempio di bean con metodo refresh

Le operazioni sopra descritte vengono eseguite tramite la scannerizzazione di codici QR, univoci per ciascun imbarcazione. Questa procedura è implementata nella classe QRCodeScanner all'interno del package relativo qrCodeScannerPackage. L'operazione di scanning è eseguita grazie al supporto della libreria me.dm7.barcodescanner.zxing. Nelle versioni più recenti di Android è necessario concedere i permessi alle applicazioni non sviluppate da Google, di accedere ai servizi dello smartphone come la fotocamera. Perciò, al momento dell'avvio dell'Activity QRCodeScanner viene verificato il valore dei permessi relativi all'utilizzo della fotocamera e, nel caso in cui non siano stati ancora concessi, viene mostrata una finestra di dialogo che permette di consentire o negare i permessi.

```
1  @Override
2  protected void onCreate(Bundle savedInstanceState) {
3      super.onCreate(savedInstanceState);
4      setRequestedOrientation(ActivityInfo.
           SCREEN_ORIENTATION_PORTRAIT);
```

```
5
6       // check current value of the permissions
7       if (ContextCompat.checkSelfPermission(this, Manifest.
            permission.CAMERA) != PackageManager.PERMISSION_GRANTED)
             {
8           //permissions requests
9           ActivityCompat.requestPermissions(this,
10                  new String[]{Manifest.permission.CAMERA}, 1);
11      }else{
12          //starting QRcode scanner
13          startScan();
14      }
15  }
16
17  @Override
18  public void onRequestPermissionsResult(int requestCode, @NonNull
        String permissions[], @NonNull int[] grantResults) {
19
20      switch (requestCode) {
21          case 1: {
22              if (grantResults.length > 0 && grantResults[0] ==
                    PackageManager.PERMISSION_GRANTED) {
23                  //starting QRcode scanner
24                  startScan();
25              } else {
26                  /*
27                  * Error code of permissions not granted
28                  */
29              }
30              break;
31          }
32      }
33  }
```

Listing 2.11: Controllo permessi fotocamera

La parte relativa alla View dell'applicazione, ovvero i file xml presenti nel package layout della directory res, sono gestiti all'interno delle corrispettive Activities. Onde evitare di creare un numero eccessivo di Activity, sono stati implementati Fragments per ciascuna schermata che non necessita di una gestione autonoma del thread UI. Questi sono contenuti nel package fragments e vengono switchati all'interno della HomeActivity in base alla schermata selezionata dal menù laterale a scomparsa o dalla home di OneMileProject App.

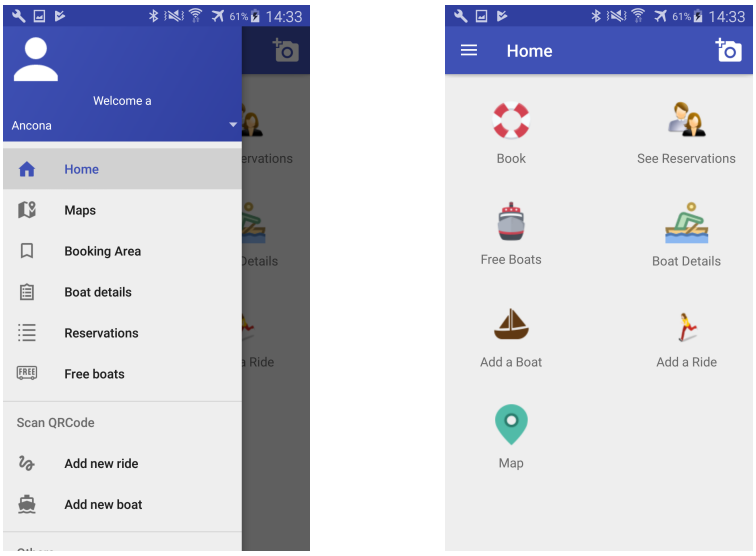


Figura 2.7: Menù e Home di OneMileProject App

La GUI dell'applicazione è stata implementata in modo da riuscire ad esaltare al meglio le potenzialità del cervello umano come "riconoscere e associare", "generalizzare e dedurre". Per far sì che ciò sia possibile, ci si è concentrati su "chi dovrà utilizzare l'applicazione" e "per cosa essa verrà utilizzata". Così facendo, sono stati definiti gli obiettivi principali dell'implementazione grafica:

- facilità di navigazione;
- memorabilità;
- efficacia;
- utilizzo di colori adeguati;
- prevenzione degli errori;
- utilizzo di fonts adeguati.

Il raggiungimento di questi requisiti è stato possibile grazie all'utilizzo di icone semplici, metaforiche e facili da ricordare, all'utilizzo di colori vivaci per le aree piccole e neutri per le aree grandi, all'utilizzo di un contrasto

efficace tra testo e sfondo e all'utilizzo di font leggibili in relazione al tipo e alle caratteristiche del carattere. Inoltre, si è mostrato particolare riguardo alla posizione dei vari elementi nelle diverse schermate, anche se la situazione potrebbe essere ulteriormente migliorata.

2.2 Funzionamento

All'avvio dell'applicazione la prima schermata che appare è quella relativa al login, con la possibilità di effettuarlo, registrarsi oppure richiedere il recupero della password.

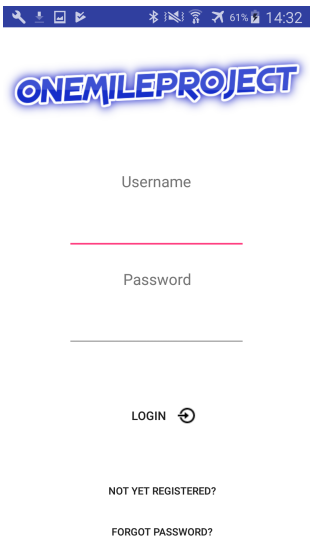


Figura 2.8: Schermata di login

Una volta eseguito con successo il login è possibile usufruire delle funzionalità offerte dall'applicazione come la visualizzazione delle posizioni e la scannerizzazione di QRCode. All'interno della schermata home sono presenti le icone relative ai componenti principali del software, ripetuti all'interno del menù laterale. Quest'ultimo ha il supplemento di quelle funzionalità non presenti nella home come il logout, le impostazioni e la schermata relativa alle info del programma.

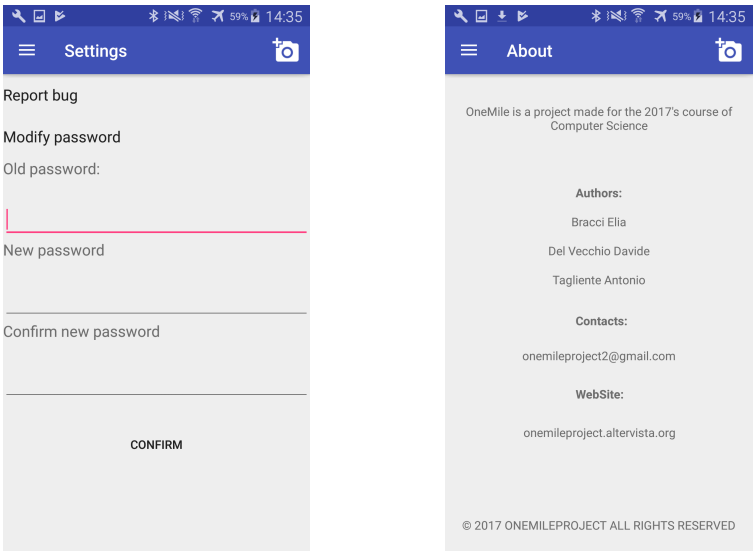


Figura 2.9: Settings e About di OneMileProject App. La modifica della password è mostrata a video tramite un menù a scomparsa

Dal menù è inoltre possibile selezionare la location di riferimento. Nel momento in cui si desidera visualizzare le posizioni sulla mappa premendo sul bottone o sull'icona Map, verranno mostrate le posizioni solamente delle imbarcazioni relative alla location selezionata precedentemente.

Figura 2.10: Visualizzazione della Mappa

Nel caso in cui si desiderasse vedere le posizioni solamente di una determinata imbarcazione, si può procedere o cliccando sull'imbarcazione desiderata nella schermata Free Boats oppure accedendo alla schermata relativa ai dettagli dell'imbarcazione Boat Details e successivamente premendo su Display positions.

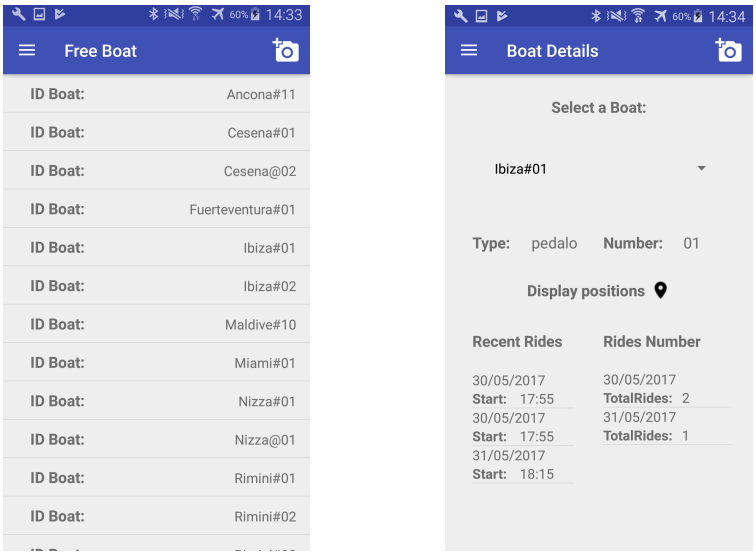


Figura 2.11: Free Boats e Boat Details di OneMileProject App.

La schermata che verrà visualizzata, sarà caratterizzata da un marker di colore diverso, indicante l'ultima posizione, in ordine cronologico, rilevata.

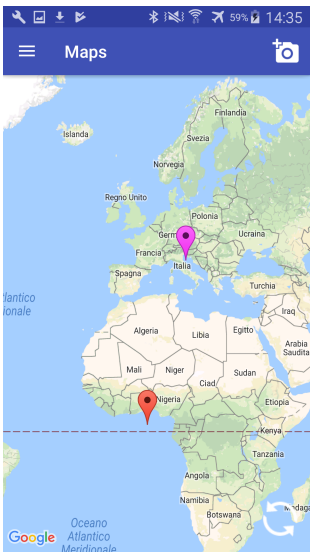


Figura 2.12: Esempio di visualizzazione dei marker sulla mappa

Per le utenze dell'applicazione è possibile effettuare prenotazioni per ciascun imbarcazione e successivamente visualizzarle con l'eventualità di eliminarle.

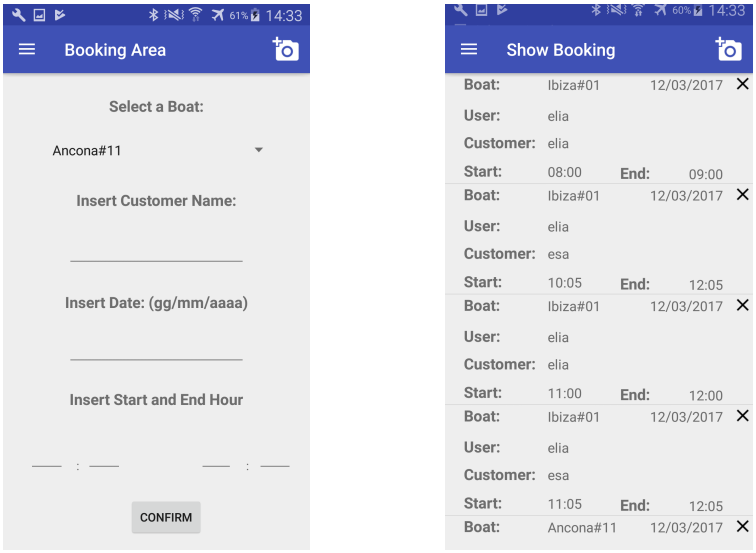


Figura 2.13: Schermata Booking e Show Booking di OneMileProject App

L'ultima funzionalità, ma non di certo in ordine di importanza, è la capacità di scannerizzare i QRcode con un duplice scopo: aggiungere nuove corse oppure aggiungere nuove imbarcazioni. È possibile accedere a questa schermata premendo rispettivamente sui bottoni o sulle icone dedicate. Nel caso in cui si stia utilizzando una versione di Android recente, come spiegato nella sezione precedente, verrà mostrata la schermata con la richiesta dei permessi per accedere alla fotocamera. Al fine di facilitare l'aggiunta di nuove corse, dato che sarà la funzionalità maggiormente utilizzata, si è scelto di inserire nell'angolo superiore destro di ciascuna schermata dell'applicazione un bottone di scelta rapida.

Figura 2.14: Schermata di scannerizzazione

Durante l'esecuzione dell'applicazione è possibile riscontrarsi di fronte a schermate di errore riguardanti, per esempio, la mancata connessione a internet oppure l'inserimento di dati errati nei diversi form.

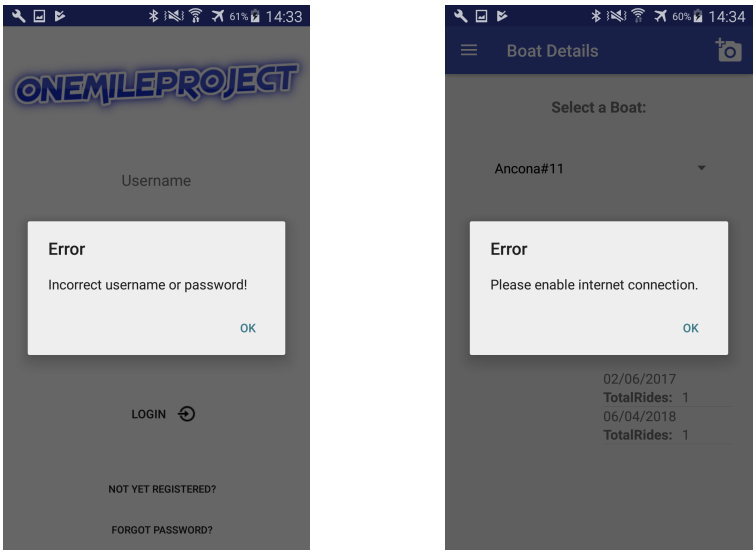


Figura 2.15: Esempi di AlertDialog di errore

Capitolo 3

Programmazione e implementazione prototipale del sottosistema RESTful

3.1 Struttura e caratteristiche

La necessità di memorizzare le informazioni in rete, riguardanti al sistema in uso, ha portato ad optare per la creazione di un servizio RESTful, semplice e veloce, che sia in grado di gestire qualsiasi informazione presente nel database.

L'acronimo REST e la sua espressione "representational state transfer" furono introdotti nel 2000 nella tesi di dottorato di Roy Fielding, uno dei principali autori delle specifiche dell'HTTP. Esso consiste nell'avvalersi di identificatori globali (URI) per accedere ad informazioni e risorse presenti sul WEB [12]. La scelta di utilizzare questa tipologia di sottosistema è dovuta dal fatto che, oltre ad offrire ottime prestazioni in termini di velocità ed efficienza, è strettamente sconsigliato effettuare connessioni dirette ad un database online tramite applicazioni Android. In aggiunta al fatto che al giorno d'oggi non siano ancora stati ufficialmente rilasciati strumenti e/o librerie dai programmatori di Google che permettono di effettuare questa operazione, gli smartphone non sono tutt'ora in grado di offrire una stabilità e qualità di connessione paragonabile a quella di un personal computer.

3.1.1 Protocollo HTTPS

Con lo scopo di migliorare la sicurezza delle connessioni stabilite con il servizio, il sottosistema utilizza il protocollo HTTPS, un protocollo per la comunicazione su Internet che protegge l'integrità e la riservatezza dei dati scambiati tra i computer e i siti [13]. I dati inviati utilizzando HTTPS vengono tutelati mediante il protocollo Transport Layer Security (TLS), che fornisce tre livelli di protezione essenziali:

- crittografia. I dati scambiati vengono criptati per proteggerli dalle intercettazioni. Ciò significa che, mentre l'utente consulta un sito web, nessuno può "ascoltare" le sue conversazioni, tenere traccia delle attività svolte in più pagine o carpire le sue informazioni;

- integrità dei dati. I dati non possono essere modificati o danneggiati durante il trasferimento, intenzionalmente o meno, senza essere rilevati;

- autenticazione. Dimostra che gli utenti comunicano con il sito web previsto. Protegge da attacchi man-in-the-middle e infonde fiducia negli utenti, il che si traduce in altri vantaggi commerciali.

Per poter attivare il protocollo HTTPS in un sito è necessario ottenere un certificato di sicurezza emesso da un'autorità di certificazione (CA), che adotta misure per verificare che il tuo indirizzo web appartenga effettivamente alla tua organizzazione. Il certificato utilizzato dal servizio REST del progetto OneMile è stato rilasciato da COMODO ECC Domain Validation Secure Server CA 2 [14]. Questa certificazione utilizza il protocollo TLS 1.2, AES_128_GCM come algoritmo di cifratura e garantisce uno scambio sicuro di chiavi grazie a ECDHE_ECDSA con X25519.

3.1.2 Architettura

La struttura del sottosistema OneMileProjectRESTful, accessibile tramite l'URL `https://onemileprojectrestful.altervista.org`, è composta da una parte riguardante la VIEW, una contenente tutto ciò che riguarda il corpo vero proprio del servizio REST e il database MySQL ad esso collegato. La View si suddivide in quattro semplici schermate:

- la schermata di Home;

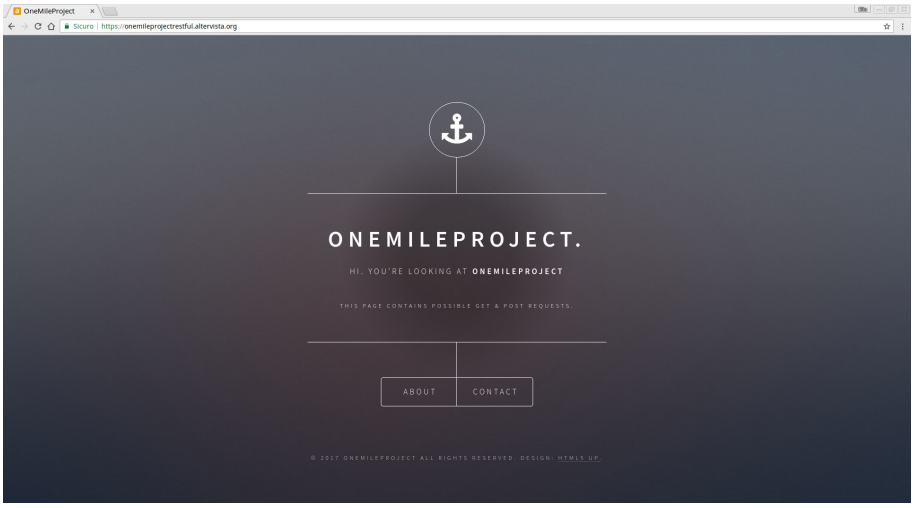


Figura 3.1: Home del sito RESTful

- la schermata About;

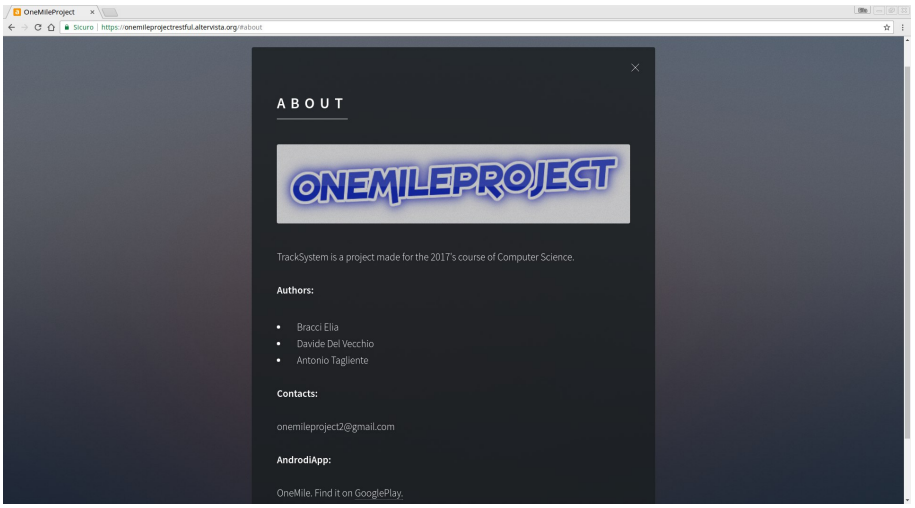


Figura 3.2: Pagina about del sito RESTful

- la schermata Contact;

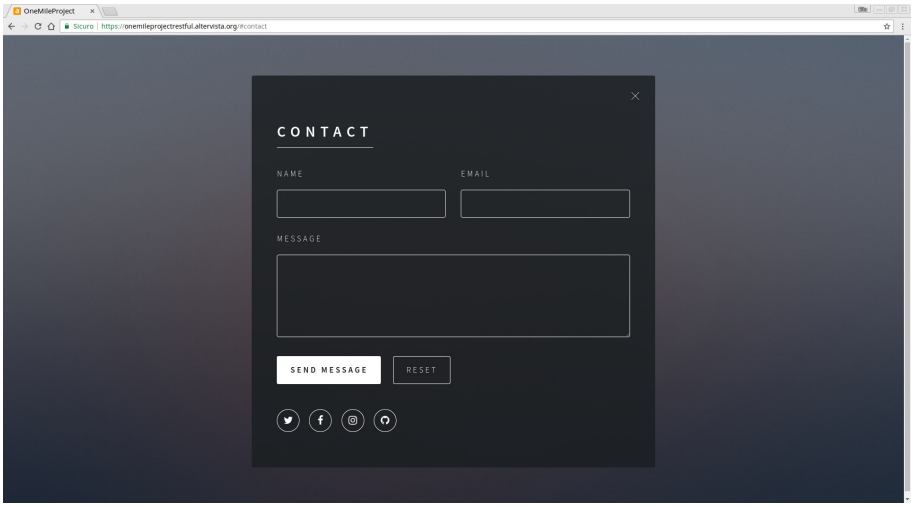


Figura 3.3: Pagina contact del sito RESTful

- la schermata di errore.

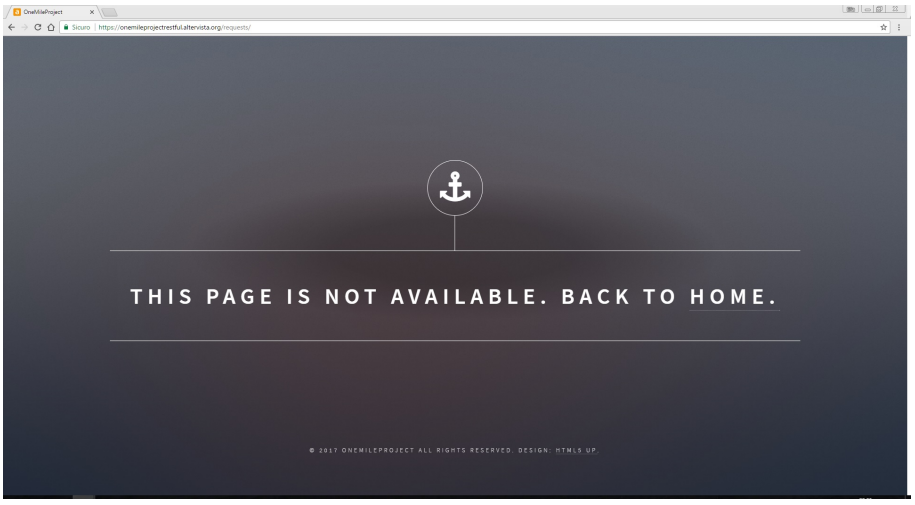


Figura 3.4: Pagina di errore nel tentativo di accesso al folder contenente le richieste gestite dal sito RESTful

La schermata Contact sopra illustrata, permette tramite uno script in PHP, qui sotto riportato, di inviare una email all'indirizzo di posta elettronica del nostro progetto (onemileproject2@gmail.com) utilizzando il metodo 'mail' fornito da Apache, il free web service più diffuso e utilizzato al giorno d'oggi [15].

```
1  <?php
2  /*
3   * Following code will send an email to onemileproject2@gmail.
       com
4   * from the Apache Web Server hosting OneMileProjectRESTful
5   */
6  if(isset($_POST['email']) && isset($_POST['name'])
7     && isset($_POST['message'])){
8
9    $email = 'onemileproject2@gmail.com';
10   $subject = 'Contact on onemileprojectrestful site.';
11   $message =  'From: ' . $_POST['name'] . "\n" . 'Email: ' .
12     $_POST['email'] . "\n" . 'Message: ' . $_POST['message'];
13     mail($email, $subject, $message);
14
15  }
16
17 ?>
```

Listing 3.1: Script per l'invio di email

Le pagine di gestione delle richieste HTTP sono state programmate utilizzando PHP, un linguaggio di scripting concepito per la programmazione di pagine web dinamiche [16]. Ciascuna di queste pagine interagisce ed instaura una connessione con il database del sito, presente in locale sui server del servizio di webhosting utilizzato Altervista. La connessione al database avviene tramite la pagina chiamata 'db_connect.php'. Quest'ultima, per far sì che possa eseguire i suoi metodi, necessita dello script 'db_config.php', contenente tutte le informazioni necessarie per stabilire la connessione con la base di dati, ovvero il nome del server, l'username e la password di accesso e il nome del database con il quale si desidera comunicare. I metodi contenuti al suo interno (della pagina 'db_connect') sono due: connect e close. Il primo è chiamato ogni volta che si voglia effettuare una nuova connessione, mentre il secondo viene chiamato al termine dell'esecuzione di tutte le operazioni eseguite sulla banca dati. Così facendo si è in grado di evitare

connessioni persistenti, risparmiando risorse nel sistema e garantendo una migliore sicurezza.

```php
<?php

/**
 * A class file to connect to database
 */
class DB_CONNECT {

    /**
     * Function to connect with database
     */
    function connect() {
        // import database connection variables
        require_once __DIR__ . '/db_config.php';

        // Connecting to mysql database
        $con = new mysqli(DB_SERVER, DB_USER,
            DB_PASSWORD, DB_DATABASE) or die(mysqli_error());

        // returing connection cursor
        return $con;
    }

    /**
     * Function to close db connection
    */
    function close($con) {
        // closing db connection
        mysqli_close($con);
    }

}

?>
```

Listing 3.2: Classe per la connessione al database

Le altre pagine PHP sono strutturate in modo tale da riuscire a gestire le richieste HTTP provenienti dai diversi client. Le tipologie di richiesta si distinguono in GET e POST; la differenza tra i due metodi è nel passaggio dei parametri al server. Nelle GET requests i parametri sono contenuti nell'URL della richiesta stessa, mentre nelle POST requests i parametri sono contenuti all'interno del body. Le prime vengono utilizzate nel momento in

cui si desidera eseguire operazioni di Read, mentre le seconde per tutte le altre operazioni ovvero Create, Update e Delete. [17].

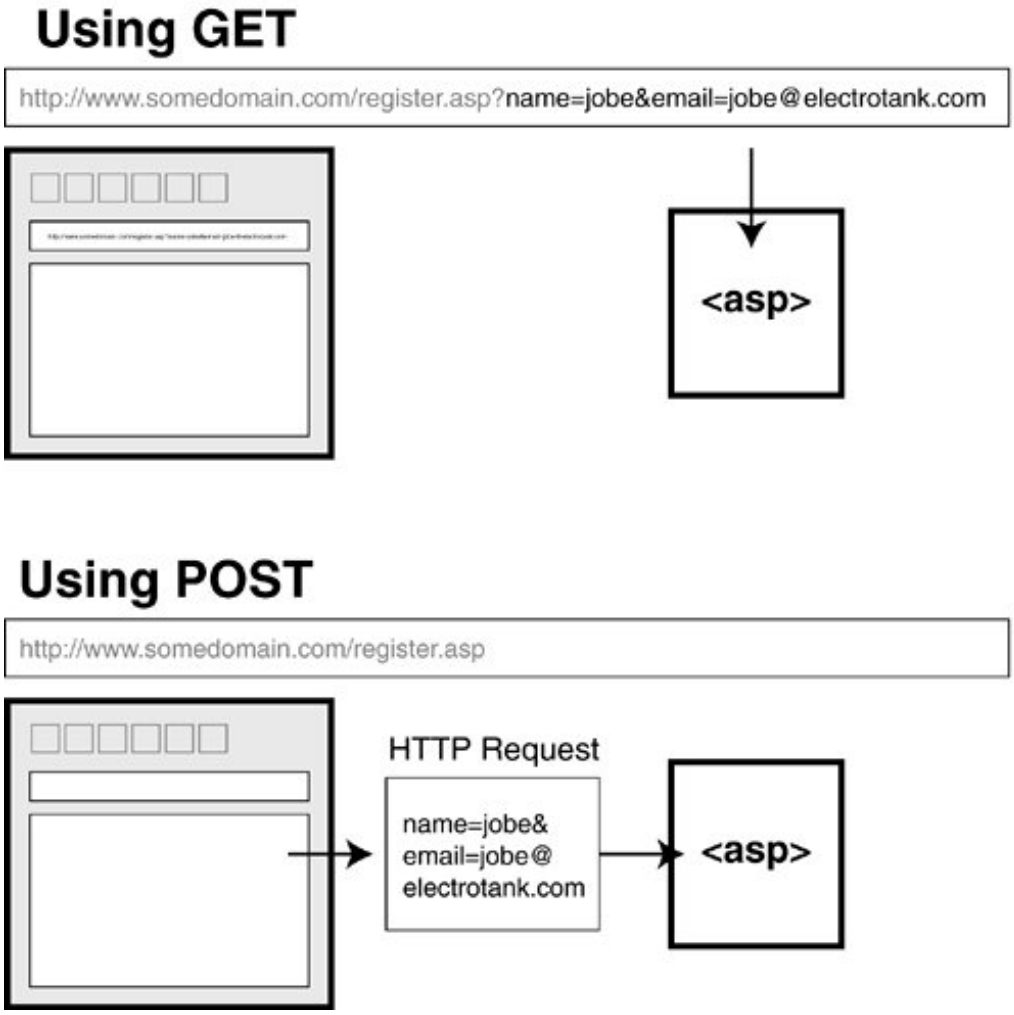


Figura 3.5: GET vs POST

Qui di seguito sono riportati due esempi di classi che gestiscono rispettivamente richieste con metodologia POST e richieste con metodologia GET.

```
1 <?php
```

```
2
3  /*
4   * Following code will create a new boat row
5   * All boat details are read from HTTP Post Request
6   */
7
8  // array for JSON response
9  $response = array();
10
11 //this line add the permessions to accept all http requests
12 header('Access-Control-Allow-Origin: *');
13
14 // check for required fields
15 if(isset($_POST['token']) && $_POST['token'] == "YourToken"){
16     if (isset($_POST['id_boat']) && isset($_POST['location'])
17         && isset($_POST['num']) && isset($_POST['type'])) {
18
19         $id_boat = $_POST['id_boat'];
20         $location = $_POST['location'];
21         $num = $_POST['num'];
22         $type = $_POST['type'];
23
24     // include db connect class
25         require_once __DIR__ . '/db_connect.php';
26
27     // connecting to db
28         $db = new DB_CONNECT();
29
30         $mysqli = $db->connect();
31
32     // mysql inserting a new row
33         $result = mysqli_query($mysqli, "INSERT INTO Boat(
               id_boat, location, num, type) VALUES('$id_boat', '
               $location', '$num', '$type')");
34
35     // check if row inserted or not
36         if ($result) {
37         // successfully inserted into database
38             $response["success"] = 1;
39             $response["message"] = "Boat successfully created.";
40
41         // echoing JSON response
42             echo json_encode($response);
43             $db->close($mysqli);
44         } else {
```

```
45          // failed to insert row
46              $response["success"] = 0;
47              $response["message"] = "Oops! An error occurred.";
48
49              $db->close($mysqli);
50          // echoing JSON response
51              echo json_encode($response);
52          }
53      } else {
54      // required field is missing
55          $response["success"] = 0;
56          $response["message"] = "Required field(s) is missing";
57
58          $db->close($mysqli);
59          // echoing JSON response
60          echo json_encode($response);
61      }
62  }else{
63      //incorrect token
64   $response["success"] = 0;
65      $response["message"] = "Incorrect token";
66
67      // echoing JSON response
68      echo json_encode($response);
69  }
70  ?>
```

Listing 3.3: Esempio di gestione di una richiesta con metodo POST

```
1   <?php
2
3   /*
4    * Following code will list all users
5    */
6
7   // array for JSON response
8   $response = array();
9
10  //this line add the permessions to accept all http requests
11  header('Access-Control-Allow-Origin: *');
12
13  if(isset($_GET['token']) && $_GET['token'] == "YourToken"){
14  // include db connect class
15      require_once __DIR__ . '/db_connect.php';
16
```

```
17 // connecting to db
18     $db = new DB_CONNECT();
19
20     $mysqli = $db->connect();
21
22 // get all products from products table
23     $result = mysqli_query($mysqli, "SELECT *FROM User")
24         or die(mysqli_error());
25
26 // check for empty result
27     if (mysqli_num_rows($result) > 0) {
28     // looping through all results
29     // products node
30         $response["users"] = array();
31
32         while ($row = mysqli_fetch_array($result)) {
33         // temp user array
34             $user = array();
35             $user["username"] = $row["username"];
36             $user["password"] = $row["password"];
37
38         // push single user into final response array
39             array_push($response["users"], $user);
40         }
41     // success
42         $response["success"] = 1;
43         $response["message"] = "Success!";
44         $db->close($mysqli);
45     // echoing JSON response
46         echo json_encode($response);
47     } else {
48     // no users found
49         $response["success"] = 0;
50         $response["message"] = "No users founded";
51
52         $db->close($mysqli);
53      // echo no users JSON
54         echo json_encode($response);
55     }
56 }else{
57  // incorrect token
58     $response["success"] = 0;
59     $response["message"] = "Incorrect token";
60
61     // echo no users JSON
```

```
62      echo json_encode($response);
63  }
64
65  ?>
```

Listing 3.4: Esempio di gestione di una richiesta con metodo GET

Le pagine PHP presenti in questo sottosistema sono contenute nella directory requests, che risulta non accessibile tramite URL diretto.

Come precedentemente accennato, il database utilizzato è di tipo MySQL, il più diffuso RDBMS(Relational database management system) a livello mondiale [18]. Di conseguenza le query sono effettuate utilizzando SQL (Structured Query Language), un linguaggio standardizzato per database basati sul modello relazionale progettato per [19]:

- creare e modificare schemi di database (DDL - Data Definition Language);
- inserire, modificare e gestire dati memorizzati (DML - Data Manipulation Language);
- interrogare i dati memorizzati (DQL - Data Query Language);
- creare e gestire strumenti di controllo ed accesso ai dati (DCL - Data Control Language).

```
1  SELECT * FROM Customers;
```

Listing 3.5: Esempio query SQL

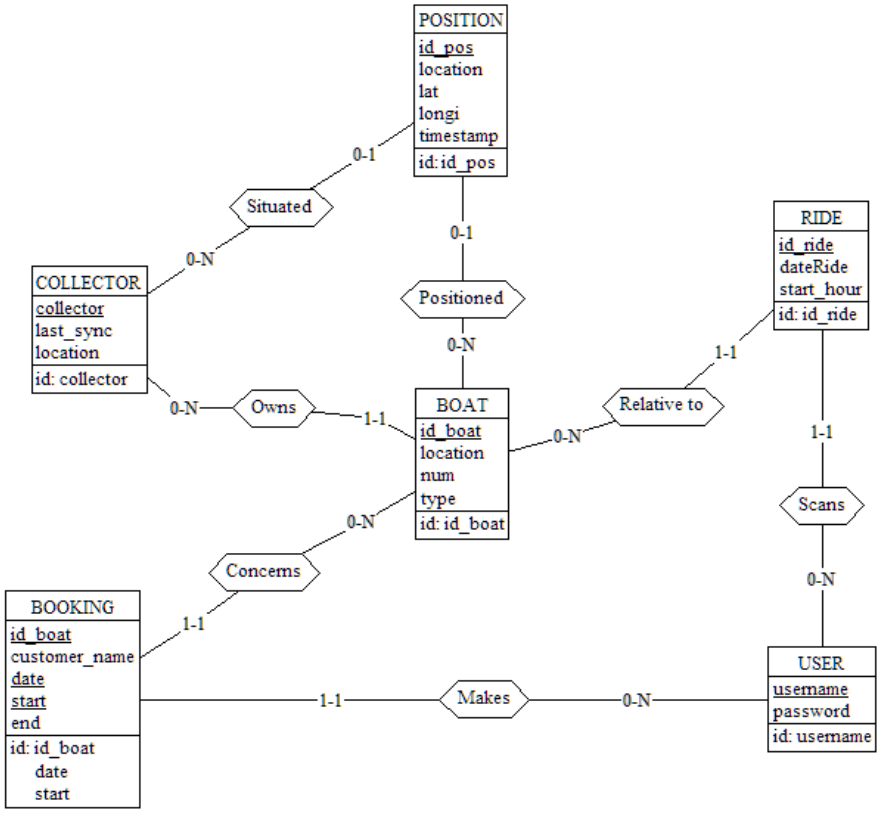


Figura 3.6: Schema Database MySQL di OneMile

3.2 Funzionamento

Il funzionamento del sottosistema RESTful sopra descritto è semplice, sicuro e rapido. Per aumentare ulteriormente la sicurezza e impedire che chiunque possa accedere e/o effettuare richieste al servizio, si è inserita una costante prestabilita, chiamata "token". Il suo valore è conosciuto solamente da coloro che hanno programmato il sistema. Ciascuna richiesta, per far sì che vada a buon fine, deve contenere al suo interno il valore di questa costante permettendo così alla pagina PHP di verificarne il valore e la correttezza. L'esempio qui sotto riportato riguarda una richiesta HTTP di tipologia GET, in cui i parametri come descritto nella sezione precedente, sono passati

all'interno dell'URL dopo il carattere "?" e suddivisi dal carattere "&". Più precisamente si sta richiedendo a OneMileProjectRestful di restituirci l'elenco delle posizioni inerenti alla location "Ibiza":

`https://onemileprojectrestful.altervista.org/requests/get_all_positions_from_location.php?token=YourToken&location=Ibiza`

Al termine dell'esecuzione si possono riscontrare diverse tipologie di risposta:

- richiesta andata a buon fine. In questo caso si otterrà come risposta un JSON contenente l'esito dell'operazione nel campo "success", il messaggio sempre riguardante l'esito "Success!" e l'array delle posizioni richieste;

  ```
  {"positions":[{"id_boat":"Ibiza#01","location":"Ibiza",
  "lat":"44.356701","longi":"13.111281","timestamp":
  "30\/02\/2017 18:15"},{"id_boat":"Ibiza#02",
  "location":"Ibiza","lat":"0", "longi":"0","timestamp":
  "30\/04\/2017 14:15"}], "success":1,"message":"Success!"}
  ```

- richiesta non andata a buon fine. Questa situazione si può verificare in diverse casistiche:

  - token errato. Il servizio restituirà un JSON contenente l'esito negativo della richiesta e il messaggio di errore "Incorrect token";

    ```
    {"success":0,"message":"Incorrect token"}
    ```

  - nessun risultato. Nel caso in cui non esistono elementi presenti nel database che soddisfino le nostre richieste, il JSON di risposta conterrà l'esito negativo della richiesta e, in questo caso, il messaggio "No positions found";

    ```
    {"success":0,"message":"No positions found"}
    ```

  - requisiti mancanti. Se la richiesta effettuata è priva dei requisiti necessari per fare in modo che essa vada a buon termine, il JSON conterrà l'esito negativo dell'operazione e il messaggio "Required field(s) is missing".

    ```
    {"success":0,"message":"Required field(s) is missing"}
    ```

Esempio di richiesta POST per inserire una nuova imbarcazione:

```
https://onemileprojectrestful.altervista.org/requests/create_boat.php
```

```
Body of the request:

token=YourToken&
id_boat=ID_BOAT&
location=LOCATION&
num=BOAT_NUM&
type=BOAT_TYPE
```

Anche in questo caso l'esecuzione della richiesta porta ad avere diverse risposte a seconda dell'esito dell'operazione:

- richiesta andata a buon fine. In questo caso si otterrà come risposta un JSON contenente l'esito dell'operazione nel campo "success" e il messaggio riguardante l'esito positivo "Boat successfully created!";

  ```
  {"success":1,"message":"Boat successfully created."}
  ```

- richiesta non andata a buon fine. Questa situazione si può verificare in diverse casistiche:

  - token errato. Il servizio restituirà un JSON contenente l'esito negativo della richiesta e il messaggio di errore "Incorrect token";

    ```
    {"success":0,"message":"Incorrect token"}
    ```

  - informazione già presente. Nel caso in cui all'interno del database sono già presenti le informazioni che si sta provando ad inserire, il JSON di risposta conterrà l'esito negativo della richiesta e, in questo caso, il messaggio di errore "Oops! An error occurred.";

    ```
    {"success":0,"message":"Oops! An error occurred."}
    ```

  - requisiti mancanti. Se la richiesta effettuata è priva dei requisiti necessari per fare in modo che essa vada a buon termine, il JSON conterrà l'esito negativo dell'operazione e il messaggio "Required field(s) is missing";

```
{"success":0,"message":"Required field(s) is missing"}
```

I metodi di formulazione delle richieste variano in base al client che le effettua, mentre la struttura della richiesta finale che si andrà ad ottenere sarà, indipendentemente dal metodo utilizzato, sempre la stessa come nei casi precedentemente analizzati.

Capitolo 4

Validazione e testing

4.1 Sottosistema Android

4.1.1 Test del codice Java

Per effettuare i test riguardanti il codice dell'applicazione scritto in Java, sono state utilizzate le classi esempio fornite di default da Android Studio. Nello specifico, i test effettuati sono stati di due tipi:

- JUnit test. Utilizzati per il test dei metodi Java implementati;

```
1  @Test
2  public void checkListSorting() throws Exception {
3      List<Booking> dates = new LinkedList<>();
4      dates.add(new Booking("Test#01", "Elia", "Bracci", "
           12/12/2017", "14:10", "15:10"));
5      dates.add(new Booking("Test#01", "Elia", "Bracci", "
           12/12/2017", "14:09", "15:09"));
6      dates.add(new Booking("Test#01", "Elia", "Bracci", "
           10/12/2016", "17:00", "18:00"));
7
8      this.sortBookListByDate(dates);
9      this.sortBookListByHour(dates);
10
11     List<Booking> orderedDates = new LinkedList<>();
12     orderedDates.add(new Booking("Test#01", "Elia", "Bracci
           ", "10/12/2016", "17:00", "18:00"));
13     orderedDates.add(new Booking("Test#01", "Elia", "Bracci
           ", "12/12/2017", "14:09", "15:09"));
```

```
14      orderedDates.add(new Booking("Test#01", "Elia", "Bracci
            ", "12/12/2017", "14:10", "15:10"));
15
16      assertEquals(orderedDates, dates);
17
18  }
```

Listing 4.1: Android esempio JunitTest

- Instrumented Unit Tests. Utilizzati per il test delle unità strumentali dell'applicazione come i package e le Activities.

```
1  @Test
2  public void useAppContext() throws Exception {
3      // Context of the app under test.
4      Context appContext = InstrumentationRegistry.
           getTargetContext();
5
6      assertEquals("com.example.root.onemile", appContext.
           getPackageName());
7  }
```

Listing 4.2: Esempio di test sulle attività strumentali

La complessità dei test effettuati ha riscontrato esito positivo.

4.1.2 Test dell'interfaccia grafica

La parte di testing con maggiore rilevanza è quella relativa alla GUI dell'applicazione. Per verificare che la grafica di OneMileProject fosse adeguata e ben realizzata, come prima cosa è stata installata l'APK dell'applicazione su diversi dispositivi di varie grandezze (in termini di pollici dello schermo). Così facendo si è riuscito a verificare che l'applicazione fosse ben visibile in ciascuno di essi. Questa procedura ha permesso di eseguire test sulla User Experience. L'applicazione è stata adoperata da diversi soggetti con fascia d'età dai 16 ai 50 anni, con lo scopo di rilevare eventuali problematiche e/o miglioramenti da apportare. Le verifiche effettuate sono andate a buon fine e hanno permesso di migliorare di volta in volta OneMileProject App in base ai feedback rilasciati dagli utenti.

4.2 Sottosistema RESTful

4.2.1 Test del server HTTPS

Come accennato nel capitolo riguardante il sottosistema RESTful, il servizio del progetto OneMile utilizza il protocollo di comunicazione HTTPS.

Figura 4.1: Connessione sicura da browser Google Chrome

Per verificarne l'efficacia e la sicurezza, sono state effettuate verifiche sul server tramite servizi web come `https://www.ssllabs.com`. Grazie al riscontro positivo ottenuto, si può affermare che il servizio RESTful è protetto da eventuali attacchi informatici attivi come il man in the middle o passivi come lo sniffing.

Figura 4.2: Test del server HTTPS

4.2.2 Test delle richieste HTTP

Con lo scopo di verificare il corretto funzionamento delle richieste gestite dal server, sono state utilizzate applicazioni web come Postman. Tramite quest'ultimo, si è stato in grado di verificare la conformità di tutte le richieste HTTP presenti nel servizio, ottenendo sempre il risultato preventivato. Qui sotto sono riportati alcuni esempi.

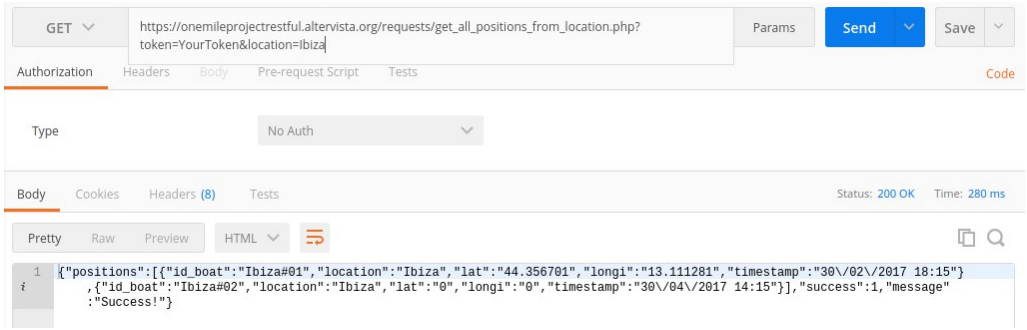


Figura 4.3: Test richiesta HTTP con metodo GET utilizzando Postman

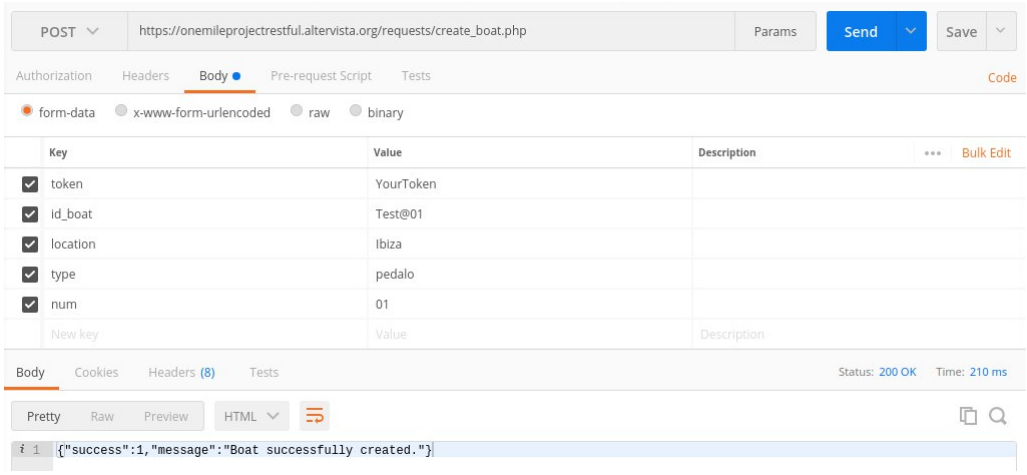


Figura 4.4: Test richiesta HTTP con metodo POST utilizzando Postman

Inoltre, sono stati validati tutti i JSON di risposta inviati dal sottosistema RESTful tramite il supporto offerto da `https://jsonlint.com/`.

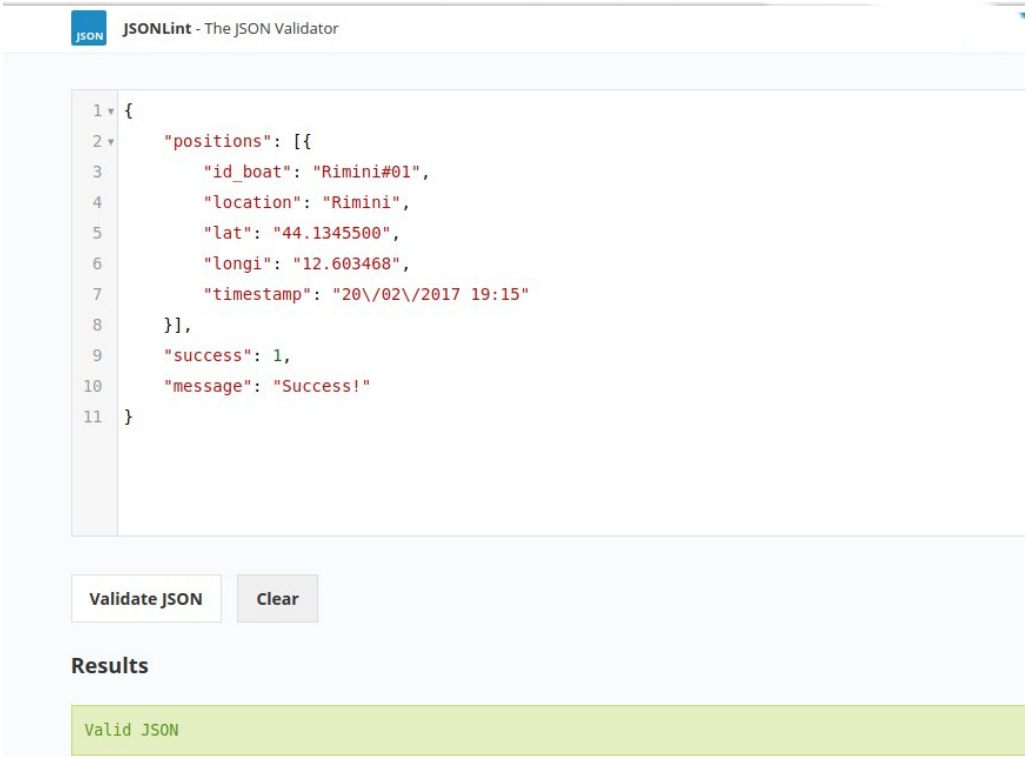


Figura 4.5: Validazione del JSON

Capitolo 5

Conclusione e sviluppi futuri

5.1 Resoconto

I test effettuati ci hanno permesso di confermare il raggiungimento degli obiettivi prefissati per il progetto OneMile. I tempi di sviluppo e progettazione preventivati si sono dimostrati pressoché veritieri, permettendo al nostro gruppo di agire ed operare in maniera elastica. Noi ci riteniamo altamente soddisfatti del nostro operato e del sistema ottenuto al termine di questo primo passo del progetto.

Il cliente è rimasto a sua volta soddisfatto del prodotto da noi presentatogli, dichiarando l'intenzione di installarlo in ciascuno dei pattini in suo possesso. Tuttavia, il Sig.Cortal non ha messo in discussione il fatto che esso sia ancora migliorabile.

OneMile è da considerarsi un notevole passo avanti per il mondo della nautica, soprattutto per il settore relativo al noleggio di pedalò.

5.2 Sviluppi futuri

Il progetto OneMile continuerà a pieno regime il suo sviluppo, con l'aggiunta di nuove funzionalità. Si sta già lavorando all'implementazione di un algoritmo per la rilevazione automatica della costa, in modo da ottenere una visualizzazione dei dati ancora più precisa.

Per ciò che riguarda invece i sottosistemi da me implementati, procederò con un miglioramento del client che effettua le richieste HTTP da OneMi-

leProject App, implementando una variante che sia in grado di inviare richieste a server HTTPS. Così facendo, la versione di Android utilizzata non sarà più un problema da tenere in considerazione. Inoltre, provvederò ad inserire all'interno del sistema RESTful uno script capace di identificare il client richiedente le informazioni e di conseguenza collegarsi al database del cliente correlato. Grazie a questo script si riuscirà ad utilizzare lo stesso servizio RESTful per ciascun acquirente ma prelevando i dati da database differenti.

Infinite sono ancora le possibilità che l'informatica e la tecnologia possono offrire all'uomo, perciò è mia intenzione sfruttarle al meglio per far sì che OneMile arrivi al massimo delle proprie potenzialità.

"Le nostre azioni non sono state ancora
all'altezza della vastità delle sfide esistenti."

Barack Obama

Ringraziamenti

Volevo ringraziare in primis tutti coloro che mi sono stati vicini e mi hanno permesso di raggiungere questo traguardo, il professor Ricci per la sua disponibilità e competenza, Tagliente e Del Vecchio per l'eccellente lavoro svolto, la mia famiglia e per ultimi, ma non meno importanti, Xavier Cortal della InterCortal e Gianluca Bracci del Centro Nautico Adriatico che ci hanno permesso di realizzare questo progetto.

Bibliografia

[1] Rita Esposito. Kurzweil: La tecnologia è un processo esponenziale. http://startegy.it/kurzweil-la-tecnologia-e-un-processo-esponenziale/. [Online; accessed 03/09/2017].

[2] Wikipedia. Internet delle cose. https://it.wikipedia.org/wiki/Internet_delle_cose. [Online; accessed 29/08/2017].

[3] Mauro Bellini. Internet of things, gli ambiti applicativi in italia. https://www.internet4things.it/iot-library/internet-of-things-gli-ambiti-applicativi-in-italia/, 5 novembre 2016. [Online; accessed 02/09/2017].

[4] Wikipedia. Ubiquitos computing. https://it.wikipedia.org/wiki/Ubiquitous_computing. [Online; accessed 29/08/2017].

[5] Treccani. Gps. http://www.treccani.it/enciclopedia/gps/. [Online; accessed 01/09/2017].

[6] GPS Colombiano. Cyrus gps tracker. http://www.gpstrackingtracker.com/GPS/Cyrus. [Online; accessed 02/09/2017].

[7] Wikipedia. Android. https://it.wikipedia.org/wiki/Android. [Online; accessed 28/08/2017].

[8] Android. Android. https://www.android.com/. [Online; accessed 28/08/2017].

[9] Android. Android studio. https://developer.android.com/studio/index.html. [Online; accessed 28/08/2017].

[10] Wikipedia. Google app engine. https://it.wikipedia.org/wiki/Google_App_Engine. [Online; accessed 28/08/2017].

[11] Android. Android manifest. https://developer.android.com/guide/topics/manifest/manifest-intro.html. [Online; accessed 28/08/2017].

[12] Wikipedia. Representational state transfer. https://it.wikipedia.org/wiki/Representational_State_Transfer. [Online; accessed 03/09/2017].

[13] Google. Proteggere il sito con il protocollo https. https://support.google.com/webmasters/answer/6073543?hl=it. [Online; accessed 05/09/2017].

[14] Tbs certificates. Comodo ecc domain validation secure server ca. https://www.tbs-certificates.co.uk/FAQ/en/COMODOECCDomainValidationSecureServerCA.html. [Online; accessed 05/09/2017].

[15] The Apache Software Foundation. Apache. https://www.apache.org/foundation/. [Online; accessed 05/09/2017].

[16] PHP. Manuale php. http://php.net/manual/it/intro-whatis.php. [Online; accessed 02/09/2017].

[17] w3schools. Http methods: Get vs. post. https://www.w3schools.com/tags/ref_httpmethods.asp. [Online; accessed 03/09/2017].

[18] MySQL. Mysql. https://www.mysql.com/it/. [Online; accessed 12/09/2017].

[19] Wikipedia. Structured query language. https://it.wikipedia.org/wiki/Structured_Query_Language. [Online; accessed 12/09/2017].